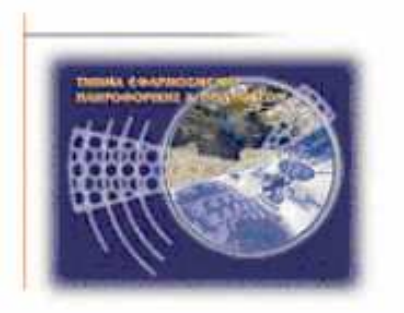
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος:
Η Εργονομική Διαδρομή του Ίππου σε Σκακιέρα NxN.

Σεραφειμίδης Ιωσήφ (ΑΜ: 1803)

Επιβλέπων καθηγητής : Καραγιαννάκης Δημήτριος

SUMMARY

We apply and also investigate the use of the heuristic algorithm proposed by H. C. Warnsdorff in the early 19th century on the problem of finding the knights path and we examine its possible extension for a more efficient solution in the case of "ties". We describe and review the procedure for discovering knight paths on all chessboards NxN with N ≥ 5. Highlight of differences when N is even and when it is odd. Examples of the application of the rule of Warnsdorff are provided on boards of various sizes and of various solutions of the "ties" with the use of C++. Finally, various visual mapping of knight paths are presented along with a results analysis by use of Matlab.

ΣΥΝΟΨΗ

Υλοποιούμε και διερευνούμε την χρήση του ευριστικού αλγορίθμου που προτάθηκε από τον H. C. Warnsdorff στις αρχές του 19ου αιώνα για το πρόβλημα της εύρεσης της διαδρομής του ίππου και εξετάζουμε την δυνατότητα επέκτασης αυτού για την αποδοτικότερη λύση σε περιπτώσεις «ισοπαλίας». Περιγραφή και εξέταση της διαδικασίας εύρεσης των διαδρομών του ίππου σε σκακιέρες NxN με N ≥ 5.Δίνεται μία καταγραφή των διαφορών για N άρτιο και περιττό καθώς και παραδείγματα με την εφαρμογή του κανόνα του Warnsdorff σε σκακιέρες διάφορων διαστάσεων και με διάφορες λύσεις «ισοπαλίας» με την χρήση της C++. Τέλος, δίνεται η οπτική αποτύπωση διαφόρων διαδρομών και ανάλυση αποτελεσμάτων πολλαπλών διαδρομών με χρήση Matlab.

Περιεχόμενα

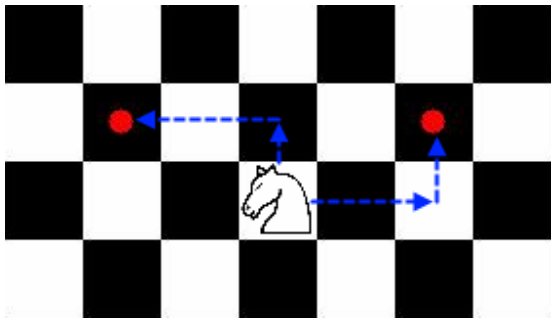
Πίνακες

Εικόνες

1.ΕΙΣΑΓΩΓΗ

Το κλασικό σκάκι ως γνωστόν παίζεται πάνω σε ένα τετράγωνο διάγραμμα, την σκακιέρα, χωριζόμενο σε 8 γραμμές και 8 στήλες. Για τις ανάγκες της παρούσας εργασίας θα δανειστούμε τον έναν ίππο καθώς και την σκακιέρα αλλά όχι μόνο στις 8x8 διαστάσεις της αλλά θα εργαστούμε και στην γενικότερη περίπτωση της NxN σκακιέρας με $N \geq 5$.

Ο ίππος αν και δεν είναι το πιο δυνατό κομμάτι στην σκακιέρα είναι πιθανόν, το πιο ενδιαφέρον λόγο του τρόπου με τον οποίο κινείται. Για να κινηθεί σχηματίζει μια γωνία, σαν ένα κεφαλαίο Γ, και αυτό μπορεί να το κάνει προς κάθε κατεύθυνση κινούμενο είτε δυο θέσεις και άλλη μία σε κάθετη κατεύθυνση προς την πρώτη κίνηση, είτε μία θέση και άλλες δυο πάλη σε κάθετη κατεύθυνση προς την πρώτη κίνηση.



Εικόνα 1 - Κίνηση του ίππου

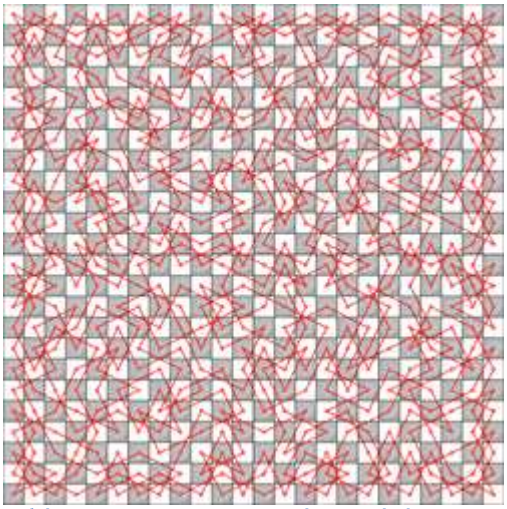
Το πρόβλημα του ίππου είναι το ερώτημα εάν υπάρχει μία διαδρομή που μπορεί να ακολουθήσει ο ίππος περνώντας από όλα τα τετράγωνα της σκακιέρας χωρίς να πατήσει δεύτερη φορά σε ένα από αυτά.

1.1.Σύντομη αναφορά στην ιστορία του προβλήματος

Τα παλιότερα παραδείγματα διαδρομών του ίππου βρίσκονται σε ένα αραβικό χειρόγραφο του 840μ.χ. (*History of Chess (1913) H.J.R.Murray*). Ακολουθούν κάποια Ινδικά χειρόγραφα που χρονολογούνται γύρο στο 900μ.χ. τα οποία περιλαμβάνουν, εκτός της διαδρομής του ίππου και διαδρομές άλλον ειδικευμένων κομματιών όπως αυτό του άρματος και του ελέφαντα. Από τότε μέχρι τώρα εμφανίζονται αρκετές αναφορές σε αυτό το πρόβλημα με τις σύγχρονες μελέτες να αρχίζουν με την επανανακάλυψη του στις αρχές του 18ου αιώνα (*http://www.mayhematics.com/t/history/1a.htm G. P. Jelliss*).

Η πρώτη μαθηματική ανάλυση του προβλήματος έγινε το 1759 από τον παραγωγικότερο μαθηματικό της εποχής του, τον Leonhard Euler στην Ακαδημία Επιστημών στο Βερολίνο, αλλά δεν δημοσιεύτηκε μέχρι το1766. Η δημοσίευση αυτή είναι το σημείο εκκίνησης πολλών μετέπειτα αναλύσεων του θέματος όπως αυτή του **Vandermonde** ο οποίος στο έργο του **Remarques sur des problèmes de situation** το 1771 μελέτησε και αυτός την βόλτα του ίππου.

Στην διάρκεια της ιστορίας του προβλήματος, πολλοί προσπάθησαν να επινοήσουν κάποιο απλό κανόνα της κίνησης του ιππότη που θα οδηγήσει στην ολοκλήρωση της περιοδείας του στην σκακιέρα. Οι περισσότεροι, ωστόσο κανόνες αυτοί είναι μόνο οδηγοί μιας λύσης και αν εφαρμοστούν αυστηρά δεν οδηγούν σε εντελώς σωστές διαδρομές, ενώ εκείνοι που το κάνουν τείνουν να είναι δύσκολοι να εφαρμοστούν και λειτουργούν μόνο σε πολύ περιορισμένες περιπτώσεις. Η ποιο γνωστή ίσως μέθοδος είναι αυτή του Warnsdorff την οποία πραγματεύεται η παρούσα εργασία.



Εικόνα 2 - Κλειστή διαδρομή ίππου σε 24 × 24 σκακιέρα. Επίλύση με χρήση νευρωνικού δίκτυου

Οι ποιο σύγχρονες μελέτες αναδεικνύουν το πόσο ενδιαφέρον είναι το πρόβλημα του ίππου αφού έχει υπολογιστεί ότι στην 8x8 σκακιέρα υπάρχουν 13,267,364,410,532 (Martin Loebbing; Ingo Wegener (1996). "The Number of Knight's Tours Equals 13,267,364,410,532 — Counting with Binary Decision Diagrams") διατεταγμένες κλειστές διαδρομές (υπό την έννοια ότι αν έχουν αντίθετους προσανατολισμούς ή αν είναι στροφές ή ανακλάσεις μετρώνται ξεχωριστά) ενώ οι ανοιχτές είναι της τάξης 4*10^51 ή για την ακρίβεια 3926356053343005839641342729308535057127083875101072 ξεχωριστές διαδρομές οι οποίες βεβαίως δεν έχουν υπολογιστεί επιδή αυτό το εγχείρημα προϋποθέτει ανυπολόγιστη υπολογιστική ισχύ και χρόνο περισσότερο από αυτόν που έχουν στην διάθεση τους τα άστρα μέχρι να σβήσει και το τελευταίο από αυτά.

1.2.Κίνητρο εργασίας

Η γεωμετρία της κίνησης του ίππου είναι πολύ ενδιαφέρουσα και φαίνεται φυσικό να θέλουμε να λύσουμε το πρόβλημα της εύρεσης της διαδρομής του ίππου. Το πρόβλημα αυτό έχει αναβιώσει στις μέρες μας με την αξιοποίηση ηλεκτρονικών υπολογιστών και την χρήση άπληστων ή ευριστικών και αναδρομικών ή μη αλγορίθμων για την μελέτη και επίλυσή του. Χρειάζεται καλή γνώση μιας γλώσσας προγραμματισμού και ικανότητας επίλυσης προβλημάτων με αλγορίθμους για την υλοποίηση ενός τέτοιου αλγορίθμου, άρα προσφέρεται για την αξιολόγηση γνώσεων σε αυτούς τους τομείς.

1.3. Σκοπός εργασίας

Στην παρούσα εργασία θα χρησιμοποιήσουμε τον ευριστικό αλγόριθμο του Warnsdorff, χωρίς αναδρομές μέσα στις διαδρομές, για να μπορέσουμε να συγκεντρώσουμε τις διαδρομές που βρίσκει απευθείας και να προσπαθήσουμε να βγάλουμε κάποια συμπεράσματα από αυτές εφαρμόζοντας διάφορες μεθόδους επίλυσης προβλημάτων που αναδύουν. Επίσης θα επισημάνουμε κάποια ενδιαφέροντα χαρακτηριστικά των αποτελεσμάτων τα οποία με περεταίρω διεργασία μπορεί να οδηγήσουν σε μια ποιο ολοκληρωμένη κατανόηση του προβλήματος του ίππου.

1.4. Δομή της εργασίας

Η δομή της εργασίας είναι η εξής:

- Στο πρώτο εισαγωγικό κεφάλαιο γίνεται μια σύντομη ιστορική αναδρομή στο αντικείμενο της εργασίας και μια αρχική περιγραφή, αναφέροντας τους σκοπούς και τους στόχους της.
- Στο κεφάλαιο 2 περιγράφονται οι έννοιες και στοιχεία που χρησιμοποιούνται.
- Στο κεφάλαιο 3 δίνονται κάποιες εξηγήσεις για τους περιορισμούς στην υλοποίηση του αλγορίθμου που εφαρμόζονται στην εργασία αυτή για πρακτικούς λόγους.
- Στο κεφάλαιο 4 περιγράφεται ο αλγόριθμος της εφαρμογής του κανόνα του Warnsdorff και η λογική με την οποία πάρθηκαν κάποιες αποφάσεις για την υλοποίηση του.
- Στο κεφάλαιο 5 υπάρχει μία μικρή περιγραφή για τα προγράμματα που χρησιμοποιήθηκαν.
- Στο κεφάλαιο 6 περιγράφεται ο κώδικας της υλοποίησης στην γλώσσα C++ ανά συνάρτηση.
- Στο κεφάλαιο 7 περιγράφονται οι συναρτήσεις στο Matlab που χρησιμοποιήθηκαν για την γραφική απεικόνιση των διαδρομών και άλλων στατιστικών δεδομένων.
- Στο κεφάλαιο 8 αναλύονται τα αποτελέσματα της εφαρμογής και επισημαίνονται κάποιες ενδιαφέρουσες παρατηρήσεις από αυτά.
- Στο κεφάλαιο 9 προτείνονται κάποιες επεκτάσεις του αλγορίθμου και περιγράφεται πολύ σύντομα πως θα μπορούσε να επεκταθεί σε σκακιέρες μεγαλύτερου μεγέθους.
- Το κεφάλαιο 10 είναι μια περίληψη της εργασίας και τον κυριότερων χαρακτηριστικών της σε μορφή δημοσίευσης.
- Τέλος στο παράστημα δίνεται όλος ο κώδικας της εργασίας και περεταίρω εικόνες και γραφήματα.

2. Έννοιες

2.1. Ονομασία θέσεων

Όταν θα θέλουμε να αναφερθούμε σε ένα τετράγωνο της σκακιέρας δεν θα το κάνουμε με κάποιου είδους σκακιστικής γραφής (π.χ.f3-d2, g5) αλλά ξεκινώντας με ένα μαύρο τετράγωνο πάνω αριστερά, μετράμε γραμμές και στήλες για να βρούμε την θέση, π.χ. για μια θέση στην 3η γραμμή και 4η στήλη, γράφουμε (3, 4).

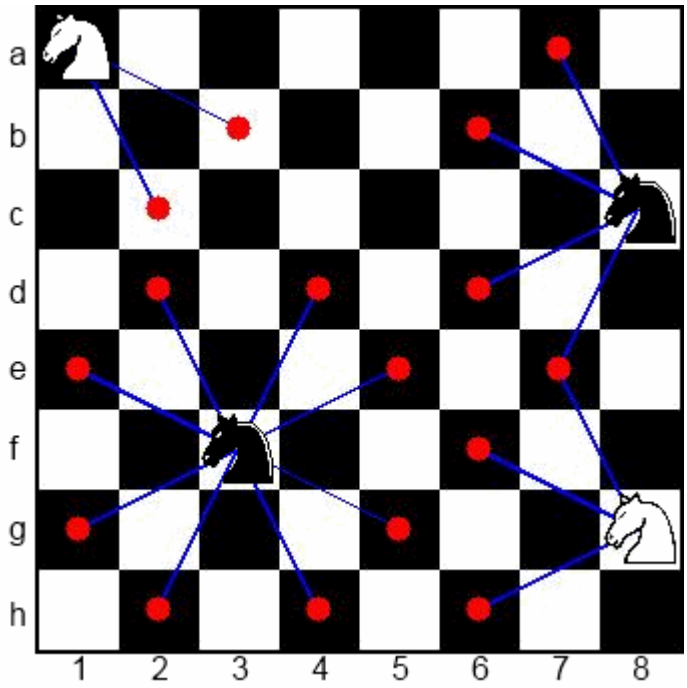
2.2. Παρακείμενες θέσεις

Θα λέμε ότι δύο θέσεις είναι παρακείμενες ή γειτονικές εάν μπορούμε να πάμε από την μία στην άλλη με μία κίνηση του ίππου. Γενικότερα:

Για κάθε σκακιέρα NxN δύο θέσεις $p_1(\alpha,\beta)$ και $p_2(\gamma,\delta)$ θα είναι παρακείμενες εάν και μόνο εάν $1\leq\alpha,\beta,\gamma,\delta\leq N$ και $(\alpha, \beta) \in \{(\gamma+2, \delta+1), (\gamma+2, \delta-1), (\gamma-2, \delta+1), (\gamma-2, \delta-1), (\gamma+1, \delta+2), (\gamma+1, \delta-2), (\gamma-1, \delta+2), (\gamma-1, \delta-2)\}$.

Πίνακας 1 – Παρακείμενες θέσεις

Το πόσες παρακείμενες θέσεις έχει μια δεδομένη θέση εξαρτάτε από το που αυτή βρίσκετε μέσα στην σκακιέρα. Οι θέσεις στα άκρα ή κοντά στα άκρα έχουν λιγότερες παρακείμενες, με τις γωνίες να έχουν μόνο 2 και τα τετράγωνα κοντά στο κέντρο να έχουν 8.



Εικόνα 3 - Κινήσεις ίππων από διάφορες θέσεις στην σκακιέρα

2.3. Ελεύθερη θέση και βαθμός θέσης

Θα λέμε ότι μία θέση είναι ελεύθερη εάν δεν την έχουμε επισκεφτεί προηγουμένως και θα λέμε βαθμό μιας θέσης τον αριθμό των ελεύθερων παρακείμενων

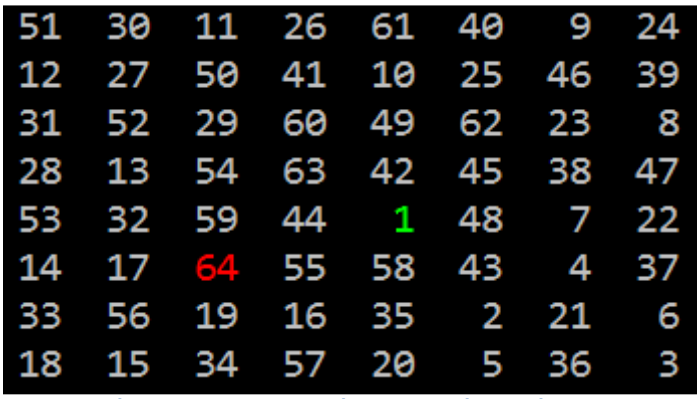
θέσεων σε αυτή. Επίσης ονομάζουμε περίπτωση ισοβαθμίας ή ισοπαλίας την κατάσταση κατά την οποία μία θέση έχει δύο η παραπάνω παρακείμενες με τον ίδιο μικρότερο βαθμό.

2.4. Διαδρομή του ίππου

Η διαδρομή του ίππου είναι μία αλληλουχία κινήσεων κατά την οποία επισκεπτόμαστε τα τετράγωνα της σκακιέρας κινούμενοι σύμφωνα με τους κανόνες της κίνησης του ίππου και χωρίς να επισκεφτούμε ένα τετράγωνο δύο φορές. Εάν επισκεφτούμε ένα τετράγωνο δεύτερη φορά και δεν υπάρχει άλλο γειτονικό τετράγωνο ελεύθερο τότε τελειώνει η συγκεκριμένη διαδρομή του ίππου.

Υπάρχουν τριών ειδών διαδρομών του ίππου:

- Η ανολοκλήρωτη. Η διαδρομή η οποία έχει τελειώσει πριν επισκεφτούμε όλα τα τετράγωνα.
- Η ανοιχτή. Μία ολοκληρωμένη διαδρομή της οποίας η πρώτη και η τελευταία θέση δεν είναι παρακείμενες.
- Η κλειστή ή κυκλική. Μία ολοκληρωμένη διαδρομή της οποίας η πρώτη και η τελευταία θέση είναι παρακείμενες.



Εικόνα 4 - Παράδειγμα αποτύπωσης μιας κλειστής διαδρομής του ίππου

Ένας λειτουργικός τρόπος αποτύπωσης μιας διαδρομής είναι με χρήση αριθμών οι οποίοι αντιπροσωπεύουν την σειρά με την οποία επισκεπτόμαστε τα τετράγωνα. Στην εικόνα 4 βλέπουμε την αποτύπωση μιας διαδρομή σε μια σκακιέρα 8x8 με εκκίνηση στο τετράγωνο (5,5) και τέλος στο τετράγωνο (6,3).

2.5. Συμμετρία της σκακιέρας

1	2	3	4	4	3	2	1
5	6	7	8	8	7	6	5
9	10	11	12	12	11	10	9
13	14	15	16	16	15	14	13
13	14	15	16	16	15	14	13
9	10	11	12	12	11	10	9
5	6	7	8	8	7	6	5
1	2	3	4	4	3	2	1

Πίνακας 2 - Συμμετρία της σκακιέρας

Σε κάποιες υλοποιήσεις του αλγορίθμου θα εκμεταλλευτούμε την συμμετρία της σκακιέρας προς όφελος μας για να μην υπολογίζουμε τις διαδρομές από όλα τα τετράγωνα, αλλά περιορίζουμε τον αλγόριθμο σε αυτές που ξεκινούν από το πρώτο τέταρτο της σκακιέρας και έπειτα ανάγουμε την λύση στις υπόλοιπες. Αυτό γίνεται ιδιαιτέρα εύκολα στην περίπτωση μας, γιατί εξετάζουμε κυρίως ζυγές σκακιέρες στο κλασσικό τετράγωνο σχήμα.

2.6. Ευριστικοί αλγόριθμοι

Ο όρος χρησιμοποιείται για τους αλγόριθμους που βρίσκουν μια λύση από το πεδίο λύσεων ενός προβλήματος χωρίς να εγγυώνται ότι αυτή είναι η καλύτερη, ως εκ τούτου μπορούν να χαρακτηριστούν και ως κατά προσέγγιση λύσεις. Συνήθως ένας τέτοιος αλγόριθμος βρίσκει μια ικανοποιητική λύση πολύ ποιο γρήγορα από ότι θα έκανε να βρει την ιδανική λύση. Κάποιες φορές βέβαια μπορούν να είναι ακριβείς, δηλαδή βρίσκουν την καλύτερη λύση, αλλά εξακολουθούμε να αναφερόμαστε σε αυτούς τους αλγόριθμους ως ευριστικούς μέχρι να αποδειχτεί ότι πράγματι δίνουν την καλύτερη λύση. Χρησιμοποιούνται για την εύρεση λύσεων σε άλυτα ή πολύ δύσκολα προβλήματα, όπως προβλήματα χρονοπρογραμματισμού εργασιών, χρωματισμού χάρτη κ.α.

2.7. Πολυπλοκότητα αλγορίθμου

Η έννοια της πολυπλοκότητας ενός αλγόριθμου αντιπροσωπεύει το κόστος χρήσης του αλγόριθμου για την επίλυση ενός προβλήματος, άρα είναι θέμα απόδοσης του αλγορίθμου με βάση την χρήση υπολογιστικών πόρων που απαιτούνται για την επίλυση ενός προβλήματος. Οι υπολογιστικοί πόροι - ανάλογα και με την φύση του αλγόριθμου - μπορεί να είναι η CPU, η μνήμη ή ακόμη και πιθανοί δικτυακοί πόροι (πχ bandwidth) που χρησιμοποιούνται. Ουσιαστικά όμως, στα περισσότερα προβλήματα αυτό που εξετάζεται είναι η χρήση της CPU, δηλαδή πόσος χρόνος απαιτείται για την εκτέλεση του αλγόριθμου. Το πρόβλημα χαρακτηρίζεται από τα δεδομένα εισόδου ενώ η συνάρτηση της πολυπλοκότητας f(n) εκφράζει την απαίτηση του αλγόριθμου σε χρόνο εκτέλεσης σε σχέση με το μέγεθος των δεδομένων εισόδου N . Επειδή είναι ιδιαίτερα δύσκολο ή και δυνατό σε πάρα πολλές περιπτώσεις να βρεθεί μία ακριβής συνάρτηση, συνήθως σ' αυτήν την περίπτωση ενδιαφέρει η εύρεση της τιμής της f(n) στις εξής περιπτώσεις:

- Καλύτερη Περίπτωση, δηλ. η ελάχιστη τιμή της f(n)
- Χειρότερη Περίπτωση, δηλ. η μέγιστη τιμή της f(n)
- Μέση Περίπτωση, δηλ. η αναμενόμενη τιμή της f(n)

με πλέον χρήσιμη την χειρότερη περίπτωση.

2.8. Χαμιλτόνια διαδρομή και κύκλωμα

Η διαδρομή του ίππου είναι, ουσιαστικά, μία Χαμιλτόνια διαδρομή. Εάν αντικαταστήσουμε τα τετράγωνα με κόμβους και ενώσουμε τους κόμβους οι οποίοι είναι παρακείμενοι τότε δημιουργούμε έναν γράφο. Εάν ο γράφος αυτός έχει μια Χαμιλτόνια διαδρομή τότε η σκακιέρα την οποία αντιπροσωπεύει έχει μια ολοκληρωμένη διαδρομή του ίππου.

Στην θεωρία των γράφων μία Χαμιλτόνια διαδρομή είναι μια πορεία σε ένα μη-κατευθυνόμενο γράφο που επισκέπτεται κάθε κορυφή ακριβώς μία φορά. Μια Χαμιλτόνια διαδρομή είναι ένα κύκλωμα όταν η πρώτη κορυφή και η τελευταία είναι απευθείας ενωμένες. Μπορεί να υπάρχουν πολλές Χαμιλτόνιες διαδρομές για ένα γράφημα, και συχνά το ζήτημα είναι να βρούμε την συντομότερη από αυτές. Αναφερόμαστε στα Χαμιλτόνια προβλήματα γράφων και ως προβλήματα του περιπλανώμενου πωλητή. Κάθε πλήρες γράφημα (n> 2) έχει ένα κύκλωμα Χάμιλτον.

Οι διαδρομές και κυκλώματα Hamilton πήραν το όνομα τους από τον William Rowan Hamilton ο οποίος εφηύρε το παιχνίδι Icosian, επίσης γνωστό και ως παζλ του Hamilton, το οποίο προϋποθέτει την εύρεση ενός κυκλώματος Hamilton, σε ένα γράφημα - αναπαράσταση ενός δωδεκάεδρου. Ο Hamilton έλυσε αυτό το πρόβλημα χρησιμοποιώντας Icosian Calculus, μια αλγεβρική δομή βασισμένη στις ρίζες της μονάδος με πολλές ομοιότητες με τα κουαρτένια, επίσης μια εφεύρεση του Hamilton.

3. Όρια στις διαστάσεις των σκακιέρων και σκακιέρες με περιττό αριθμό τετραγώνων

Η υλοποίηση του κανόνα του Warnsdorff στην παρούσα εργασία είναι ικανή να δώσει αποτελέσματα για οποιαδήποτε NxN σκακιέρα αλλά θεωρούμε σκόπιμο - και απαραίτητο λόγο περιορισμών του συστήματος - να βάλουμε κάποια όρια στις διαστάσεις των σκακιέρων που θα μελετήσουμε. Επίσης θα επισημάνουμε την διαφορά στα αποτελέσματα μεταξύ σκακιέρων με μονό και ζυγό αριθμό τετραγώνων και γιατί θα επικεντρωθούμε κυρίως στα αποτελέσματα των δεύτερων.

3.1. Σκακιέρες NxN από 5x5 μέχρι 300x300

Για τις σκακιέρες 1x1 και 2x2 είναι προφανές ότι δεν υπάρχει διαδρομή του ίππου γιατί δεν υπάρχει χώρος για τον ίππο να κινηθεί.

Η σκακιέρα 3x3 είναι ενδιαφέρουσα περίπτωση γιατί εάν τοποθετήσουμε τον ίππο σε κάποια θέση εκτός της κεντρικής και εφαρμόσουμε τον κανόνα του Warnsdorff δημιουργείτε μία κυκλική διαδρομή που περνάει από όλα τα τετράγωνα εκτός του κεντρικού. Αυτή η ημιτελής διαδρομή είναι και μοναδική για την σκακιέρα αυτή αφού από όποιο εξωτερικό τετράγωνο και να ξεκινήσουμε ο ίππος θα διαγράψει την ίδια πορεία. Από το κεντρικό τετράγωνο ο ίππος δεν μπορεί να κάνει καμία κίνηση.

Η σκακιέρα 4x4 δεν έχει ολοκληρωμένες διαδρομές γιατί πάντα περισσεύουν 2 ή παραπάνω τετράγωνα που δεν είναι γειτονικά και δεν μπορούν να γίνουν μέρος της διαδρομής. Στην βιβλιογραφία έχει αποδειχτεί ότι όλες οι σκακιέρες 4xN έχουν ανοιχτές ολοκληρωμένες διαδρομές εκτός από την 4x4.

Από το μέγεθος 5x5 και πάνω υπάρχουν ολοκληρωμένες διαδρομές τις οποίες μπορούμε να δημιουργήσουμε και να επεξεργαστούμε. Ιστορικά αξίζει να αναφέρουμε ότι πρώτος ο Euler αποτύπωσε διαδρομές σε σκακιέρες 5x5.

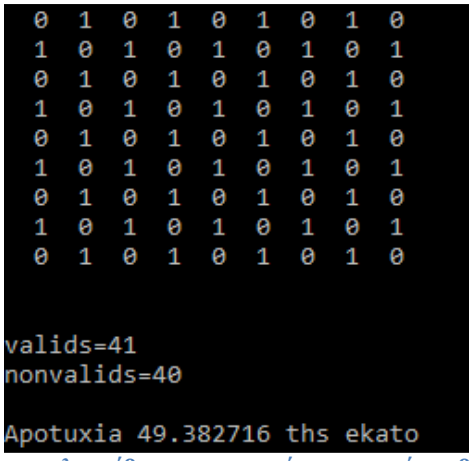
Το όριο στις διαστάσεις της μεγαλύτερης σκακιέρας στα 300x300 μπήκε για δύο λόγους. Ο ένας είναι οι τεχνικοί περιορισμοί του συστήματος και του περιβάλλοντος στο οποίο δημιουργήθηκε και έτρεξε ο κώδικας του προγράμματος και ο άλλος είναι ο χρόνος που χρειάζεται για να ολοκληρωθούν οι υπολογισμοί σε σχετικά μεγάλες σκακιέρες με N>150 για πολλές αφετηρίες και πολλές σκακιέρες.

Οι περιορισμοί του συστήματος προέρχονται από το όριο της μνήμης που παρέχει το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε (Microsoft Visual C++) το οποίο είναι περίπου 1.000.000 bytes. Ο περιορισμός αυτός θα μπορούσε να ξεπεραστεί αλλά δεν κρίθηκε αναγκαίο γιατί τα αποτελέσματα είναι αρκετά για να βγουν τα συμπεράσματα που θέλουμε αλλά και γιατί ο χρόνος που χρειάζεται για να βγουν αυτά τα συμπεράσματα ανεβαίνει εκθετικά όσο μεγαλώνουν οι διαστάσεις των σκακιέρων.

3.2. Μονός αριθμός τετραγώνων

Για να έχει μία NxN σκακιέρα περιττό αριθμό τετραγώνων θα πρέπει το N να είναι περιττό. Σε μια τέτοια σκακιέρα ένα από τα δύο χρώματα θα έχει ένα τετράγωνο παραπάνω από το άλλο. Εάν τα μαύρα τετράγωνα είναι τα περισσότερα τότε εάν αρχίσουμε από μαύρο τετράγωνο μπορεί να βρεθεί ολοκληρωμένη διαδρομή του ίππου ενώ εάν αρχίσουμε από άσπρο τότε δεν θα μπορέσει να περάσει ο ίππος από όλες τις θέσεις γιατί στο τέλος θα μείνουν δύο μαύρα τετράγωνα και ο ίππος κινείτε μόνο από τετράγωνο του ενός χρώματος σε τετράγωνο του άλλου.

Για να μειωθεί η πολυπλοκότητα των μετέπειτα υπολογισμών και για να μην παίρνουμε συνέχεια ιδικές περιπτώσεις από δω και πέρα δεν θα λαβαίνουμε υπόψη μας τις μονές σκακιέρες. Όταν θα λέμε ότι υπολογίζουμε τις διαδρομές για τις σκακιέρες από 6 έως 100 θα είναι μόνο για τα ζυγά νούμερα.



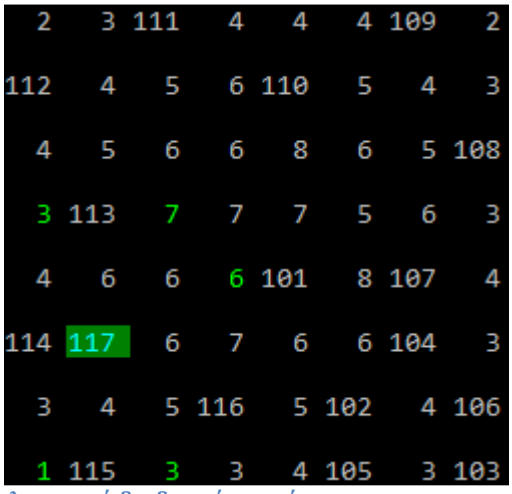
Εικόνα 5 - Μέγιστη επιτυχία του αλγορίθμου σε σκακιέρα με μονό αριθμό τετραγώνων

Από τα παραπάνω βγαίνει επίσης το συμπέρασμα ότι ακόμα και αν υπάρχουν ολοκληρωμένες διαδρομές, δεν υπάρχει καμία κλειστή διαδρομή διότι κάποια στιγμή, λόγο του μονού αριθμού τετραγώνων, θα υπάρχουν δυο τετράγωνα ίδιου χρώματος που θα πρέπει να περάσει ο ίππος συνεχόμενα για να ολοκληρωθεί η διαδρομή, πράγμα που είναι άτοπο.

4. Περιγραφή αλγορίθμου εφαρμογής του κανόνα του Warnsdorff

4.1. Ο κανόνας του Warnsdorff

Το 1823 ο H. C. Warnsdorff πρότεινε στο "Des Rösselsprungs einfachste und allgemeinste Lösung" (Η διαδρομή του ιππότη, μια απλή και γενική λύση) Schmalkalden (1823) έναν ευριστικό αλγόριθμο για την εύρεση της διαδρομής του ίππου. Ο αλγόριθμος είναι ο εξής: «Από την παρούσα θέση κινούμαστε στην παρακείμενη της με τον μικρότερο βαθμό». Όταν επισκεπτόμαστε κάποια θέση αυτή δεν υπολογίζεται πια ως πιθανός προορισμός για τον ίππο. Από αυτό είναι εμφανές ότι μετά από μία κίνηση ο βαθμός όλων των ελεύθερων παρακείμενων θέσεων του προορισμού μειώνετε κατά ένα.



Εικόνα 6 - Στιγμιότυπο υπολογισμού διαδρομής του ίππου.

Ο στόχος του αλγορίθμου είναι να αποφεύγονται τα αδιέξοδα. Να μην φτάνει δηλαδή ο ίππος σε κάποιο τετράγωνο το οποίο δεν έχει ελεύθερες παρακείμενες θέσεις χωρίς πρώτα να έχει περάσει από όλα τα τετράγωνα της σκακιέρας ακριβώς μία φορά. Ο ίππος κινείτε στην παρακείμενη ελεύθερη θέση με τον μικρότερο βαθμό για να αποφευχθεί η δημιουργία μεμονωμένων θέσεων στις οποίες ο ίππος δεν θα έχει τρόπο να φτάσει.

Όπως θα δείξουμε παρακάτω, η απλή και διαισθητική αυτή η λογική λειτουργεί αρκετά καλά, αλλά αφήνει στην κρίση αυτού που εφαρμόζει τον αλγόριθμο την λύση των ισοβαθμιών. Θα δούμε πως αυτή η φαινομενική λεπτομέρεια, προσδιορίζει αν θα ολοκληρωθεί με επιτυχία μια διαδρομή του ίππου, ή ακόμα και αν αυτή θα είναι κλειστή ή ανοιχτή.

Εξαιτίας της απλότητας του, ο κανόνας του Warnsdorff μπορεί να υλοποιηθεί αλγοριθμικά με διάφορους τρόπους. Παρακάτω περιγράφονται κάποιοι τρόποι υλοποίησης και παρατίθενται λόγοι για την επιλογή αυτών που θα χρησιμοποιηθούν σε αυτή την εργασία.

4.2. Προετοιμασία της σκακιέρας

Το πιο σημαντικό στοιχείο του αλγορίθμου, αν και δεν είναι άμεσα αντιληπτό, είναι ο υπολογισμός του βαθμού των θέσεων. Ο ευθείς τρόπος να συμπεράνουμε τον βαθμό μιας θέσης είναι να υπολογίσουμε ποιες άλλες θέσεις είναι παρακείμενες σε αυτή και πόσες από αυτές είναι ελεύθερες. Σε σκακιέρα NxN ο ακόλουθος ψευδοκώδικας υπολογίζει τον βαθμό μιας θέσης:

Η $p_0(\alpha,\beta)$ είναι μια θέση της οποίας θέλουμε να βρούμε τον βαθμό.
Θέτουμε την αρχική τιμή του βαθμού της θέσης p_0 ίση με 0.
Για κάθε p_ι (χ, ψ) για το οποίο ισχύει:

$1 \leq \chi, \psi \leq N$

και

$(\chi, \psi) \in \{(\alpha+2, \beta+1), (\alpha+2, \beta-1), (\alpha-2, \beta+1), (\alpha-2, \beta-1), (\alpha+1, \beta+2), (\alpha+1, \beta-2), (\alpha-1, \beta+2), (\alpha-1, \beta-2)\}$

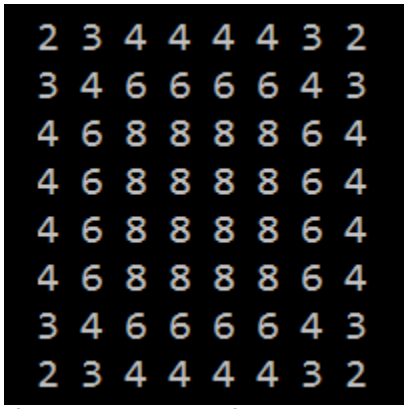
αυξάνουμε στον βαθμό της p_0 κατά 1.

Πίνακας 3 - Υπολογισμός βαθμού θέσης

Ο παραπάνω αλγόριθμος πρέπει να χρησιμοποιείτε πολλαπλές φορές σε κάθε περίπτωση που χρειάζεται να βρούμε την μικρότερου βαθμού θέση από αυτές που «απειλεί» ο ίππος. Εάν όμως σκεφτούμε ότι ο χώρος στον οποίο εργαζόμαστε είναι πάντα μια NxN σκακιέρα και ο τρόπος με τον οποίο υπολογίζονται οι παρακείμενες θέσεις είναι πάντα ίδιος , γίνετε φανερό ότι μπορούμε να προετοιμάσουμε τον βαθμό κάθε θέσης πριν καν τοποθετήσουμε τον ίππο στην σκακιέρα. Ο ίππος κινείτε κάθε φορά 1 και μετά 2 ή 2 και μετά 1 τετράγωνο προς οποιαδήποτε κατεύθυνση. Με μία απλή παρατήρηση της σκακιέρας συμπεραίνουμε τα παρακάτω για τις θέσεις της:

- Όποιες θέσεις απέχουν 2 ή περισσότερα τετράγωνα από τις άκρες της σκακιέρας έχουν 8 γειτονικές θέσεις.
- Όποιες απέχουν 1 τετράγωνο από μία άκρη και 2 ή περισσότερα τετράγωνα από τις άλλες άκρες της σκακιέρας έχουν 6 γειτονικές θέσεις.
- Όποιες είναι στην μία άκρη και απέχουν 2 ή περισσότερα τετράγωνα από τις άλλες άκρες της σκακιέρας έχουν 4 γειτονικές θέσεις.
- Όποιες απέχουν ακριβώς 1 τετράγωνο από δύο άκρες της σκακιέρας έχουν 4 γειτονικές θέσεις.
- Όποιες είναι στην μία άκρη και απέχουν ακριβώς 1 τετράγωνο από μια άλλη άκρη της σκακιέρας έχουν 3 γειτονικές θέσεις.
- Όποιες είναι στις γωνίες της σκακιέρας έχουν 2 γειτονικές θέσεις.

Εάν γράψουμε σε μορφή πίνακα τους βαθμούς των θέσεων που δημιουργούνται από τους παραπάνω κανόνες τότε παίρνουμε τον πίνακα της εικόνας 7. Με την δημιουργία ενός τέτοιου πίνακα δεν χρειάζεται να υπολογίζουμε κάθε φορά τον βαθμό των θέσεων, αλλά πρέπει να προσέχουμε να μειώνουμε τον βαθμό μιας θέσης όταν παρακείμενή της προστίθεται στην διαδρομή του ίππου.



Εικόνα 7 - Οι βαθμοί όλων των θέσεων σε μια σκακιέρα 8*8 πριν αρχίσει η διαδικασία για την εύρεση μιας διαδρομής του ίππου

4.3. Υπολογισμός επόμενης κίνησης

Το που θα κινηθεί ο ίππος ορίζεται, σύμφωνα με τον κανόνα του Warnsdorff, από τον βαθμό των θέσεων που απειλεί. Η θέση με τον μικρότερο βαθμό γίνεται ο προορισμός του ίππου. Σε κάποιες περιπτώσεις δύο η περισσότερες θέσεις μοιράζονται τον ίδιο μικρότερο βαθμό αλλά ο κανόνας του Warnsdorff δεν αναφέρεται σε τέτοιες καταστάσεις και έτσι χρειάζεται να προσθέσουμε έναν δικό μας κανόνα για να δώσει τη λύση. Στην βιβλιογραφία έχουν προταθεί διάφοροι τρόποι επίλυσης και παρακάτω θα περιγράψουμε κάποιους από αυτούς.

4.3.1. Βαθύς υπολογισμός βαθμού

Σε περίπτωση ισοπαλίας παίρνουμε τις θέσεις που ισοβαθμούν, μετράμε πόσες γειτονικές θέσεις είναι διαθέσιμες για την κάθε μία και διαλέγουμε αυτήν που έχει τις λιγότερες. Αυτό γίνετε όσες φορές χρειάζεται μέχρι να σπάσει η ισοπαλία. Αυτή η διαδικασία ακολουθεί την ίδια λογική με τον κανόνα του Warnsdorff στο ότι προσπαθεί να περάσει τον ίππο πρώτα από τις θέσεις που είναι εύκολο να απομονωθούν. Οι υπολογισμοί που γίνονται με αυτήν την μέθοδο επιβαρύνουν τον υπολογισμό της διαδρομής του ίππου εκθετικά, γιατί μπορεί να χρειαστεί να φτάσει πολύ βαθιά για να βρεθεί λύση.

4.3.2. Τυχαία επιλογή

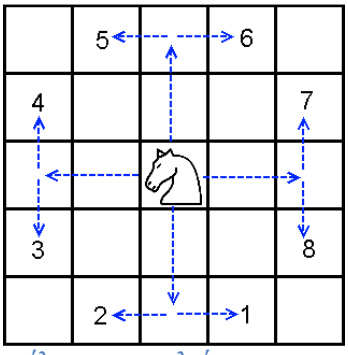
Η μόνη «οδηγία» που φέρεται να έχει δώσει ο Warnsdorff για την επίλυση των ισοβαθμιών είναι η τυχαία επιλογή μεταξύ των πιθανών θέσεων. Αυτή η τακτική δεν εγγυάται πάντα κάποια ολοκληρωμένη διαδρομή ίππου αλλά είναι πιθανό να υπάρχουν για όλες τις σκακιέρες και όλες τις αφετηρίες εάν επιλεχθούν, τυχαία, οι σωστές θέσεις.

4.3.3. Εξωτερική θέση

Από τις θέσεις που ισοβαθμούν επιλέγουμε αυτήν που βρίσκετε ποιο μακριά από το κέντρο, πράγμα το οποίο συνήθως σημαίνει την θέση που έχει τους γείτονες με τον μικρότερο βαθμό, αφού όπως έχουμε πει οι θέσεις κοντά στις άκρες της σκακιέρας έχουν λιγότερους γείτονες από τις άλλες. Αυτή η λογική αρχικά φαίνεται να είναι η καλύτερη αλλά με την εφαρμογή της βρίσκουμε εύκολα διαδρομές που καταλήγουν σε αδιέξοδο.

4.3.4. Προκαθορισμένη σειρά επίλυσης ισοπαλιών

Οι περισσότερες κινήσεις που μπορεί να κάνει ο ίππος είναι 8. Αριθμούμε αυτές τις κινήσεις με νούμερα από το ένα μέχρι το οχτώ. Όταν δύο ή παραπάνω θέσεις ισοβαθμούν τότε επιλέγουμε ως επικρατούσα αυτήν που έρχεται πρώτη στην αρίθμηση. Εάν π.χ. ισοβαθμούν οι 3-6-7 τότε παίρνουμε την 3 σαν επόμενη θέση. Αυτός είναι ο τρόπος επίλυσης ισοπαλιών στην υλοποίηση του αλγορίθμου σε αυτήν την εργασία. Η αρίθμηση ξεκινάει από την θέση «δύο κάτω, μία δεξιά» του ίππου και οι υπόλοιπες αριθμούνται σύμφωνα με την φορά του ρολογιού.



Εικόνα 8 - Προκαθορισμένη σειρά επίλυσης ισοπαλιών.

5. Εργαλεία

5.1.C++

Η γλώσσα προγραμματισμού C ορίστηκε αρχικά στο κλασσικό σύγγραμμα των Kernigham και Ritchie "The C Programming Language", και ήταν το πρότυπο που χρησιμοποιούσαν όλοι οι προγραμματιστές στη C. Το πρότυπο ANSI για τη C τελικά εγκρίθηκε τον Δεκέμβριο του 1989 και έγινε το επίσημο πρότυπο για τον προγραμματισμό στη C. Το πρότυπο ANSI εισήγαγε αρκετά νέα στοιχεία, που δεν υπήρχαν στην αρχική έκδοση των Kernigham και Ritchie, και άλλαξε κάποια άλλα, έτσι ώστε τα δύο πρότυπα δεν είναι τελείως συμβατά.

Η **C++** είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Θεωρείται μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου. Είναι μια μεταγλωττιζόμενη γλώσσα πολλαπλών παραδειγμάτων, με τύπους. Υποστηρίζει δομημένο, αντικειμενοστραφή και γενικό προγραμματισμό. Η γλώσσα αναπτύχθηκε από τον Bjarne Stroustrup το 1979 στα εργαστήρια Bell της AT&T, ως βελτίωση της ήδη υπάρχουσας γλώσσας προγραμματισμού C, και αρχικά ονομάστηκε "C with Classes", δηλαδή C με Κλάσεις. Μετονομάστηκε σε C++ το 1983. Οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα κ.α. Επειδή η αντικειμενοστραφής τεχνολογία ήταν καινούργια και όλες οι αντικειμενοστραφείς υλοποιήσεις που υπήρχαν ήταν αρκετά αργές και μη αποδοτικές, ένας δευτερεύων σκοπός της C++ ήταν να διατηρήσει την αποδοτικότητα της C.

Η C++ μπορεί να θεωρηθεί μια διαδικαστική γλώσσα με κάποιες επιπλέον δομές, μερικές από τις οποίες προστέθηκαν για αντικειμενοστραφή προγραμματισμό, ενώ άλλες για την βελτίωση του συντακτικού της γλώσσας. Ένα καλογραμμένο πρόγραμμα πρέπει να έχει στοιχεία τόσο αντικειμενοστραφή όσο και κλασσικού διαδικαστικού προγραμματισμού. Η C++ είναι ουσιαστικά μια επεκτάσιμη γλώσσα αφού μπορούμε να ορίσουμε νέους τύπους με τέτοιο τρόπο ώστε να λειτουργούν σαν τους προκαθορισμένους τύπους, που είναι τμήμα της γλώσσας. Η C++ σχεδιάστηκε για την ανάπτυξη μεγάλων προγραμμάτων.

```
#include <iostream>
using namespace std;

int main ()
{
  cout << "Hello World!";
  return 0;
}
```

Πίνακας 4 - Παράδειγμα προγράμματος C++

Στο βιβλίο του "The Design and Evolution of C++ (1994), ο Bjarne Stroustrup περιγράφει κάποιους κανόνες που χρησιμοποιεί για το σχεδιασμό της C++:

- η C++ είναι σχεδιασμένη ως μια γενικής χρήσης γλώσσα με στατικούς τύπους, που είναι όσο αποτελεσματική και φορητή, όσο η C
- η C++ είναι σχεδιασμένη να υποστηρίζει άμεσα και σφαιρικά πολλά είδη προγραμματισμού (δομημένος προγραμματισμός, αντικειμενοστραφής, γενικός προγραμματισμός)
- η C++ είναι σχεδιασμένη να δίνει επιλογές στον προγραμματιστή, ακόμα κι αν του επιτρέπει να επιλέξει λανθασμένα

- η C++ είναι σχεδιασμένη να είναι όσο το δυνατόν συμβατή με τη C, ώστε να διευκολύνει τη μετάβαση από τη C
- η C++ αποφεύγει χαρακτηριστικά που αναφέρονται σε συγκεκριμένες πλατφόρμες ή δεν είναι γενικής χρήσης
- η C++ δεν έχει κόστος για χαρακτηριστικά της γλώσσας που δεν χρησιμοποιούνται
- η C++ είναι σχεδιασμένη να λειτουργεί χωρίς κάποιο εξελιγμένο προγραμματιστικό περιβάλλον

Η C++ παρέχει περισσότερους από 30 τελεστές, που καλύπτουν τη βασική αριθμητική, το χειρισμό bit, αναφορά δεικτών, συγκρίσεις, λογικές πράξεις κ.α. Σχεδόν όλοι οι τελεστές μπορούν να υπερφορτωθούν για τύπους ορισμένους από το χρήστη, με λίγες εξαιρέσεις όπως πρόσβαση μέλους (. και .*). Το πλούσιο σύνολο από τελεστές που μπορούν να υπερφορτωθούν είναι βασικό για τη χρήση της C++ ως γλώσσα ειδικού πεδίου (domain specific language). Οι υπερφορτώσιμοι τελεστές είναι ακόμα βασικό μέρος πολλών προχωρημένων τεχνικών προγραμματισμού της C++, όπως οι έξυπνοι δείκτες. Η υπερφόρτωση ενός τελεστή δεν αλλάζει την προτεραιότητα των υπολογισμών όπου χρησιμοποιείται, ούτε τον αριθμό των τελεστέων που χρησιμοποιεί ο τελεστής (αν και οποιοσδήποτε τελεστέος μπορεί απλά να αγνοείται).

+	addition
-	subtraction
*	multiplication
/	division
%	Modulo
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
&&	AND
||	OR
&	Bitwise AND
|	Bitwise Inclusive OR
^	Bitwise Exclusive OR
~	Unary complement (bit inversion)
<<	Shift Left
>>	Shift Right

Πίνακας 5 - Βασικοί τελεστές της C++

Κατά τον προγραμματισμό με την C++, αποθηκεύονται οι μεταβλητές στη μνήμη του υπολογιστή με διαφορετικό τρόπο για κάθε τύπο. Αυτό γίνετε διότι δεν πρόκειται να καταλάβει το ίδιο ποσό της μνήμης για να αποθηκεύσει έναν απλό αριθμό από ότι για ένα γράμμα ή ένα μεγάλο αριθμό. Η μνήμη στους υπολογιστές είναι οργανωμένη σε bytes. Ένα byte είναι το ελάχιστο ποσό της μνήμης που μπορεί να διαχειριστεί η C++. Ένα byte μπορεί να αποθηκεύσει ένα σχετικά μικρό ποσό των δεδομένων: έναν απλό χαρακτήρα ή ένα μικρό ακέραιο (συνήθως έναν ακέραιο αριθμό μεταξύ 0 και 255). Επιπλέον, ο υπολογιστής μπορεί να χειριστεί πιο σύνθετους τύπους δεδομένων που προέρχονται από συνένωση πολλών byte, όπως πολύ μεγάλους ή μη ακέραιους αριθμούς.

Name	Description	Size*	Range*
`char`	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
`short int(short)`	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long int (long)`	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`bool`	Boolean value. It can take one of two values: true or false.	1byte	`true` or `false`
`float`	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
`double`	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`long double`	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`wchar_t`	Wide character.	2 *or* 4 bytes	1 wide character

Πίνακας 6 - Βασικοί τύποι μεταβλητών στην C++

5.2.MATLAB

Το Matlab (matrix laboratory) είναι ένα αριθμητικό υπολογιστικό περιβάλλον και γλώσσα προγραμματισμού τέταρτης γενιάς. Αναπτύχθηκε από την εταιρεία MathWorks και επιτρέπει χειρισμούς πινάκων, οπτική αναπαράσταση συναρτήσεων και δεδομένων, την εφαρμογή αλγορίθμων, δημιουργία φορμών διεπαφής με το χρήστη, και έχει δυνατότητες διασύνδεσης με προγράμματα που είναι γραμμένα σε άλλες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των C, C + +, Java, και Fortran.

Αν και το Matlab προορίζεται κυρίως για αριθμητικούς υπολογισμούς, υπάρχει μια προαιρετική εργαλειοθήκη (toolbox) που χρησιμοποιεί μια συμβολική μηχανή – την MuPAD - δίνοντας στους υπολογιστές την δυνατότητα για αλγεβρικούς υπολογισμούς. Ένα επιπλέον πακέτο, το Simulink, προσθέτει γραφική αναπαράσταση εργαλείων σε πολλούς επιστημονικούς τομείς και ένα μοντέλο προσομοίωσης για δυναμικά και ενσωματωμένα συστήματα.

Το Matlab είναι χτισμένο γύρω από τη γλώσσα Matlab, και η χρήση του περιλαμβάνει κυρίως πληκτρολόγηση του κώδικα Matlab στο παράθυρο εντολών (ως ένα διαδραστικό μαθηματικό πλαίσιο), ή για την εκτέλεση αρχείων κειμένου που περιέχουν συναρτήσεις και κώδικα Matlab.

```
>> x = 17
x =
 17
>> x = 'hat'
x =
hat
>> y = x + 0
y =
     104      97     116
>> x = [3*4, pi/2]
```

```
x =
   12.0000    1.5708
>> y = 3*sin(x)
y =
   -1.6097    3.0000
```

Πίνακας 7 - Εντολές και τα αποτελέσματα τους στο παράθυρο εντολών του Matalb

Όπως υποδηλώνει και το όνομά του (εργαστήριο πινάκων – Matrix Laboratory - Matlab), το Matlab μπορεί να δημιουργήσει και να επεξεργαστεί συστοιχίες μιας (διανύσματα), δυο (πίνακες), ή περισσότερων διαστάσεων. Στην Matlab, αναφερόμαστε σε ένα διάνυσμα ως ένα μονοδιάστατο (1 × N ή N × 1) πίνακα. Ένας πίνακας αναφέρεται γενικά σε ένα δισδιάστατο πίνακα, δηλαδή ένα M × N πίνακα όπου M και N είναι μεγαλύτερο από 1. Οι πίνακες με περισσότερες από δύο διαστάσεις αναφέρονται ως πολυδιάστατοι πίνακες. Οι πίνακες αυτοί αποτελούν θεμελιώδες στοιχείο του Matlab και πολλές βασικές λειτουργίες του υποστηρίζουν εγγενώς την επεξεργασία τους απευθείας χωρίς τη χρήση μαθηματικών πράξεων ή συναρτήσεων. Επομένως, η γλώσσα του Matlab είναι ένα παράδειγμα γλώσσας προγραμματισμού πινάκων.

```
>> ari = 1:5
ari =
 1 2 3 4 5
>> array = 1:2:9
array =
 1 3 5 7 9
>> array = 1:3:9
array =
 1 4 7
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
 16  3  2 13
  5 10 11  8
  9  6  7 12
  4 15 14  1
>> A(2,3)
ans =
 11
>> A(2:4,3:4)
ans =
 11 8
 7 12
 14 1
>> eye(3)
ans =
 1 0 0
 0 1 0
 0 0 1
>> zeros(2,3)
ans =
 0 0 0
 0 0 0
>> ones(2,3)
```

```
ans =
 1 1 1
 1 1 1
```

Πίνακας 8 - Δημιουργία πινάκων

6. Υλοποίηση του κανόνα του Warnsdorff με την γλώσσα προγραμματισμού C++

Ακολουθεί η περιγραφή του κώδικα που υλοποιεί τον κανόνα του Warnsdorff. Για την μελέτη του αλγορίθμου γράφτηκαν διάφορες συναρτήσεις με την καθεμία να προσφέρει διαφορετικά αποτελέσματα για την σφαιρικότερη μελέτη του. Παρακάτω θα τις περιγράψουμε ξεχωριστά. Το ανώτατο όριο 300x300 στο μέγεθος των σκακιέρων μπήκε για λόγους περιορισμών στο σύστημα που ήταν διαθέσιμο για τους υπολογισμούς των διαδρομών. Δεν υπάρχει λόγος να μας κάνει να πιστεύουμε ότι η υλοποίηση δεν δουλεύει για μεγαλύτερες σκακιέρες.

6.1. Includes and declarations

```
#include "stdafx.h"
#include <fstream>
#include <time.h>
#include <windows.h>
#include <direct.h>
#include <string>
```

Πίνακας 9 - Βιβλιοθήκες που χρησιμοποιεί το πρόγραμμα.

Οι βιβλιοθήκες της C++ είναι συλλογές από λειτουργίες, σταθερές, κλάσεις, αντικείμενα και πρότυπα που εκτείνουν την γλώσσα και παρέχουν τις βασικές λειτουργίες για να εκτελεστούν διάφορες εργασίες, όπως οδηγίες για να αλληλεπιδράσει με το λειτουργικό σύστημα, επεξεργασία δεδομένων και άλλους χρήσιμους αλγορίθμους. Τα διάφορα στοιχεία που παρέχονται από τις βιβλιοθήκες χωρίζονται σε διάφορες επιγραφές (headers) που πρέπει να περιλαμβάνονται στον κώδικα, προκειμένου να έχουμε πρόσβαση στα στοιχεία τους.

Οι επιγραφές που χρησιμοποιούμε σε αυτό το πρόγραμμα είναι οι εξής:

- #include "stdafx.h": περιέχει ένα σύνολο από επιγραφές οι οποίες φορτώνονται στην αρχή του προγράμματος και έτσι εξοικονομούμε χρόνο κατά την διάρκεια των υπολογισμών.
- #include <fstream>: επιτρέπει την εγγραφή και το διάβασμα αρχείων.
- #include <time.h>: δίνει πρόσβαση στο ρολόι του συστήματος.
- #include <windows.h>: δίνει πρόσβαση στις λειτουργίες του συστήματος των Windows.
- #include <direct.h>: περιέχει διάφορες συναρτήσεις που έχουν να κάνουν με δημιουργία και την πρόσβαση σε φακέλους του μέσου αποθήκευσης.
- #include <string>: δίνει πρόσβαση σε συναρτήσεις που χρησιμοποιούνται για την επεξεργασία συμβολοσειρών.

```
void problemKnight01_1(int, int, int); //μία αφετηρία, μία σκακιέρα
void problemKnight01_2(int, int, int); //debug-demonstration
void problemKnight01_3(int, int, int); // μία αφετηρία, μία σκακιέρα, δεδομένη
λύση με τυχαία αναδιάταξη
void problemKnight02_1(int); // όλες οι αφετηρίες, μία σκακιέρα
void problemKnight02_2(int); // όλες οι αφετηρίες, μία σκακιέρα, συγκεκριμένη
αναδιάταξη
void problemKnight02_3(int); //m σκακιέρες, όλες οι αφετηρίες, ένα τέταρτο,
συγκεκριμένη αναδιάταξη, σώσιμο έγκυρων και άκυρων θέσεων
void problemKnight03_1(int); // όλες οι αφετηρίες, μία σκακιέρα, όλες οι λύσεις
void problemKnight03_2(int); // m σκακιέρες, όλες οι αφετηρίες, όλες οι λύσεις,
σώσιμο αποτελεσμάτων
void problemKnight03_3(int); // m σκακιέρες, όλες οι αφετηρίες, όλες οι λύσεις
```

```
void problemKnight03_4(int); // m σκακιέρες, όλες οι αφετηρίες, τυχαία διάταξη
void problemKnight04(int, int*, int); // όλες οι αφετηρίες, μία σκακιέρα, σειριακή αναδιάταξη από την order_sequence(), ένα τέταρτο
void order_sequence(int, int, int); // σειριακή αναδιάταξη για το problemKnight04()
void board_setup(int, int*); //αρχικοποίηση πίνακα με όλες τις πιθανές κινήσεις από κάθε θέση
void move_check(int, int, int, int*, int*, int*, int*, int*); //εύρεση επόμενης κίνησης
void board_respec(); //επαναϋπολογισμός πιθανών κινήσεων από κάθε θέση μετά από κίνηση του ίππου
```

Πίνακας 10 - Δηλώσεις συναρτήσεων

Στην C++ χρειάζεται να δηλώσουμε τις συναρτήσεις, έξω από την βασική συνάρτηση, μαζί με τον τύπο τους και τον τύπο των ορισμάτων που χρησιμοποιεί η κάθε μία.

6.2. Συνάρτηση problemKnight01_1

Αυτή είναι η βασική συνάρτηση που περιέχει την υλοποίηση του αλγορίθμου. Θα την αναλύσουμε ολόκληρη γιατί πάνω σε αυτήν στηρίζονται οι υπόλοιπες, για τις οποίες θα επισημάνουμε μόνο τις διαφορές τους. Υπολογίζει την διαδρομή του ίππου για μια σκακιέρα και από την θέση που θα της δώσουμε. Επίσης γράφει σε αρχείο, με όνομα που της δίνουμε, την διαδρομή.

```
void problemKnight01_1(int m,int startx, int starty) {
```

Πίνακας 11 - Ορίσματα της συνάρτησης

- m = μέγεθος σκακιέρας
- startx = γραμμή θέσης εκκίνησης
- starty = στήλη θέσης εκκίνησης.

```
int n=m;
int x,y,xmax,ymax,xtemp,ytemp,moves,check=0,invalidMove;
int   pin[300][300],   xmove[8]={2,2,1,-1,-2,-2,-1,1},   ymove[8]={1,-1,-2,-2,-1,1,2,2}, order[8]={1,2,3,4,5,6,7,8};
char fName[30] = "F:\output.txt";
x = --startx; y = --starty;
pin[x][y]=100;
```

Πίνακας 12 - Δήλωση μεταβλητών και αρχικοποίηση τιμών

- n: μέγεθος σκακιέρας
- x,y,xmax,ymax,xtemp,ytemp: μεταβλητές που περιέχουν την θέση του ίππου σε διάφορα στάδια του αλγορίθμου.
- Check=0: αριθμός ανολοκλήρωτων.
- invalidMove: πόσες θέσεις γύρω από αυτή που βρίσκετε ο ίππος είναι άκυρες είτε διότι είναι έξω από την σκακιέρα είτε διότι είναι κατειλημμένες.
- pin[300][300]: πίνακας ο οποίος περιέχει την σκακιέρα.
- xmove[8]={2,2,1,-1,-2,-2,-1,1}, ymove[8]={1,-1,-2,-2,-1,1,2,2}:τα περιεχόμενα των πινάκων αυτών ανά ζευγάρι για κάθε θέση {(xmove[0],ymove[0]), (xmove[1],ymove[1]), κτλ.} μας δίνουν την κίνηση του ίπου.
- order[8]={1,2,3,4,5,6,7,8}: πίνακας ο οποίος δείχνει την σειρά με την οποία λύνονται οι ισοπαλίες.

- fName[30] = "F:\output.txt": πίνακας χαρακτήρων με το όνομα του αρχείου που θα αποθηκευθεί η διαδρομή του ίππου.
- x = --startx, y = --starty: αρχικοποίηση των x και y από τα δεδομένα που έστειλε η main.
- pin[x][y]=100: στην θέση από την οποία ξεκινάει ο ίππος δίνουμε το νούμερο 100.

```
board_setup(n, *pin);
```

Πίνακας 13 - Κλήση συνάρτησης board_setup()

Εδώ η συνάρτηση board_setup() παίρνει ως ορίσματα το μέγεθος της σκακιέρας και τον δείκτη που δείχνει στον πίνακα pin και υπολογίζει τους βαθμούς των θέσεων της σκακιέρας πριν αρχίσει ο υπολογισμός της διαδρομής. Λεπτομέρειες για το πώς ακριβώς λειτουργεί δίνονται παρακάτω.

```
for(int i=0;i<m*n-1;i++) {
	xtemp=0;
	ytemp=0;
	xmax=0;
	ymax=0;
	invalidMove=0;
	moves=9;
```

Πίνακας 14 - Επανάληψη που διατρέχει όλες τις θέσεις της σκακιέρας και αρχικοποιήσεις για κάθε επανάληψη

Η επανάληψη αυτή τρέχει τόσες φορές όσα είναι και τα τετράγωνα της σκακιέρας. Εδώ μέσα γίνονται οι απαραίτητοι υπολογισμοί για να βρεθεί η επόμενη θέση που θα πάει ο ίππος. Εάν η διαδρομή είναι ολοκληρωμένη τότε η επανάληψη θα τελειώσει κανονικά, εάν ο ίππος βρεθεί σε κάποια θέση από την οποία δεν μπορεί να πάει σε κάποιο τετράγωνο το οποίο δεν έχει επισκεφτεί τότε η επανάληψη τελειώνει πρόωρα. Οι μεταβλητές xtemp, ytemp, xmax, ymax και invalidMove μηδενίζονται για να αρχίσει η εύρεση της επόμενης θέσης του ίππου χωρίς να επηρεαστεί από την προηγούμενη. Η moves γίνετε 9 γιατί οι περισσότερες προσκείμενες θέσεις που μπορεί να έχει διαθέσιμές ένα τετράγωνο είναι 8 και παρακάτω στον κώδικα η μεταβλητή αυτή χρησιμοποιείτε για να κρίνουμε εάν μια θέση είναι έγκυρη για να πάει ο ίππος εάν το πλήθος των θέσεων που είναι προσκείμενες σε αυτή είναι μικρότερο από τον αριθμό της moves.

```
for(int e=0;e<8;e++) {
	xtemp=x+xmove[order[e]-1];
	ytemp=y+ymove[order[e]-1];
	move_check(n, xtemp, ytemp, &xmax, &ymax, &moves, &invalidMove,*pin);
}
```

Πίνακας 15 - Εξέταση όλων των θέσεων που είναι προσκείμενες στην θέση που βρίσκετε ο ίππος και εύρεση επόμενης η τερματισμός της επανάληψης

Εδώ έχουμε μία επανάληψη που τρέχει πάντα ακριβώς οχτώ φορές, μια φορά για κάθε κίνηση που μπορεί να κάνει ο ίππος. Οι μεταβλητές xtemp και ytemp περιέχουν την θέση που θα περάσει στην συνάρτηση move_check() για να συγκριθεί με την καλύτερη θέση που έχει βρεθεί μέχρι εκείνη την στιγμή. Ο πίνακας order[] περιέχει την σειρά με την οποία λύνονται οι ισοπαλίες. Αλλάζοντας την σειρά στον order[] αλάζει ο τρόπος που λύνονται οι ισοπαλίες και κατά συνέπια αλλάζουν και οι διαδρομές που

βρίσκουμε. Η σύγκριση μεταξύ των παρακείμενων της θέση του ίππου γίνεται στην συνάρτηση move_check() που θα περιγραφεί παρακάτω.

```
if(invalidMove==8)
	break;
x=xmax;
y=ymax;
pin[x][y]=101+i;
```

Πίνακας 16 - Έξοδος από την επανάληψη

Εάν η invalidMove γίνει 8 σημαίνει ότι ενώ υπάρχουν ελεύθερες θέσεις στην σκακιέρα – εάν δεν υπήρχαν θα είχε τελειώσει η εξωτερική επανάληψη – δεν υπάρχουν παρακείμενες θέσεις για να συνεχίσει ο ίππος την διαδρομή και έτσι τελειώνει πρόωρα η εξωτερική επανάληψη. Τα xmax και ymax δείχνουν την θέση στην οποία θα κινηθεί ο ίππος και καταχωρούνται στα x και y αντίστοιχα τα οποία μπαίνουν στο πίνακα pin[x][y] για να αλλάξει τον νούμερο του πίνακα σε αυτή τη θέση. Μέχρι να «πατήσει» ο ίππος σε μία θέση, αυτή περιέχει τον αριθμό των ελεύθερων παρακείμενων θέσεων της, ενώ μετά περιέχει έναν αριθμό που δείχνει την σειρά κατά την οποία ο ίππος πέρασε από εκεί συν 100 για να μην μπερδεύονται οι πρώτες 8 θέσεις της διαδρομής με τις παρακείμενες των υπολοίπων θέσεων.

```
for(int e=0;e<8;e++) {
	xtemp=x+xmove[e];
	ytemp=y+ymove[e];
	board_respec(n,xtemp, ytemp, *pin);
}
```

Πίνακας 17 - Επαναϋπολογισμός θέσεων σκακιέρας

Άλλη μία επανάληψη που τρέχει πάντα 8 φορές, η οποία αυτή τη φορά περνάει στην συνάρτηση board_respec() τις παρακείμενες της θέσης που βρήκε ο αλγόριθμος ως επόμενη για τον ίππο, έτσι ώστε να μειώσει τα νούμερα του πίνακα pin[][] που υποδεικνύουν τον βαθμό τους.

```
for(int d=0;d<n;d++) {
	for(int f=0;f<m;f++) {
		pin[d][f]=pin[d][f]-99;
		if(pin[d][f]<0) {
			check++;
			 pin[d][f]=0;
		}
		printf("%3d ",pin[d][f]);
	}
	printf("\n");
}
```

Πίνακας 18 - Εκτύπωση διαδρομής στο command window

Μετά τον υπολογισμό της διαδρομής ο πίνακας pin[][] περιέχει κάποια νούμερα πάνω από το 100 και, εάν η διαδρομή δεν είναι ολοκληρωμένη, κάποια νούμερα μονοψήφια. Εδώ έχουμε δυο επαναλήψεις, μια εξωτερική και μία εμφωλευμένη, οι οποίες διατρέχουν τον pin και αφαιρούν από όλα τα νούμερα 99. Έτσι η διαδρομή του ίππου αντιπροσωπεύεται από νούμερα με το 1 να είναι η αρχική θέση του ίππου και να φτάνει στο NxN ή, εάν είναι ανολοκλήρωτη, να φτάνει μέχρι εκεί που ο ίππος βρέθηκε σε αδιέξοδο με τα υπόλοιπα νούμερα να είναι αρνητικά. Μετά, αφού έχει κάνει τα αρνητικά νούμερα 0, εκτυπώνει τον πίνακα στο command window όπως θα φαίνονταν πάνω σε μία σκακιέρα.

```
printf("\nBoard size = %dx%d\nStart square %d,%d\nEnd square %d,%d\nUnvisited
squares = %d",m,n,startx+1,starty+1,xmax+1, ymax+1,check);
if ((abs(startx-xmax) == 1 && abs(starty-ymax) == 2) || (abs(startx-xmax) == 2
&& abs(starty-ymax) == 1)) {
	printf("\nPath is closed.");
}
else {
	printf("\nPath is open.");
}
```

Πίνακας 19 - Εκτύπωση στατιστικών αποτελεσμάτων

Στο τέλος της συνάρτησης εκτυπώνονται κάποια στατιστικά στοιχεία για την διαδρομή.

- Μέγεθος σκακιέρας.
- Αρχική θέση ίππου.
- Τελική θέση ίππου.
- Πλήθος θέσεων που δεν τα επισπεύτηκε ο ίππος.
- Μήνυμα για το εάν η διαδρομή είναι κλειστή ή ανοιχτή αφού υπολογιστεί εάν το πρώτο και το τελευταίο τετράγωνο βρίσκονται σε παρακείμενες θέσεις.

```
std::ofstream os(fName);
	for(int d=0;d<n;d++) {
		for(int f=0;f<m;f++) {
			os << pin[d][f]<<" ";
		}
		os <<"\n";
	}
}
```

Πίνακας 20 - Αποθήκευση διαδρομής σε αρχείο κειμένου

Για περεταίρω επεξεργασία της διαδρομής την αποθηκεύουμε σε ένα αρχείο κειμένου, εδώ έχει όνομα «F:\output.txt», με την ίδια μορφή που τυπώνεται στο command window όπως φαίνεται ποιο πάνω στην συνάρτηση.

Παρακάτω ακολουθεί περιγραφή των υπολοίπων συναρτήσεων. Λόγο της ομοιότητας τους με την problemKnight01_1 θα περιγράφονται μόνο οι διαφορές και οι πρόσθετες λειτουργίες τους.

6.3. Συνάρτηση problemKnight01_2

Η συνάρτηση αυτή χρησιμοποιείτε για την οπτικοποίηση των βημάτων του υπολογισμού της διαδρομής του ίππου. Καλούνται κάποια αρχεία του συστήματος για να μπορέσουμε να χρωματίσουμε συγκεκριμένα στοιχεία μέσα στο command window.

```
HANDLE h = GetStdHandle ( STD_OUTPUT_HANDLE );
WORD wOldColorAttrs;
CONSOLE_SCREEN_BUFFER_INFO csbiInfo;
GetConsoleScreenBufferInfo(h, &csbiInfo);
wOldColorAttrs = csbiInfo.wAttributes;
```

Πίνακας 21 - Κλήση και αρχικοποίηση αρχείων συστήματος

Μετά τον υπολογισμό κάθε βήματος στην διαδρομή καθαρίζεται το command window και τυπώνεται το εκάστοτε στιγμιότυπο της διαδρομής. Η τρέχουσα θέση του

ίππου αντιπροσωπεύεται από το νούμερο που δείχνει την σειρά της θέσης αυτής στην διαδρομή χρωματισμένο γαλάζιο σε πράσινο πλαίσιο. Οι παρακείμενες θέσεις έχουν πράσινο χρώμα.

```
for(d=0;d<n;d++) {
	for(f=0;f<m;f++) {
		if (((abs(x-d) == 1 && abs(y-f) == 2) || (abs(x-d) == 2 && abs(y-
f) == 1)) && pin[d][f]<9) {
			SetConsoleTextAttribute	(	h,	FOREGROUND_GREEN	|
FOREGROUND_INTENSITY );
			printf("%3d ",pin[d][f]);
			SetConsoleTextAttribute ( h, wOldColorAttrs);
		}
		else if (x == d  && y == f) {
			SetConsoleTextAttribute	(	h,	0x00023	|
FOREGROUND_INTENSITY );
			printf("%3d ",pin[d][f]);
			SetConsoleTextAttribute ( h, wOldColorAttrs);
		}
		else {
			printf("%3d ",pin[d][f]);
		}
	}
	printf("\n\n");
}
```

Πίνακας 22 - Χρωματισμός στιγμιότυπου

Στο τέλος του υπολογισμού τυπώνεται η διαδρομή του ίππου, με το 1 που δηλώνει την αρχική θέση με πράσινο χρώμα και το NxN νούμερο που δηλώνει την τελευταία θέση με μπλε χρώμα. Εάν υπάρχουν θέσεις που δεν έχει επισκεφτεί ο ίππος τότε αυτές οι θέσεις γίνονται μηδέν και χρωματίζονται κόκκινες.

```
system( "cls" );
for(d=0;d<n;d++) {
	for(f=0;f<m;f++) {
		pin[d][f]=pin[d][f]-100;
		if(pin[d][f]<0) {
			++check;
			pin[d][f]=0;
		}
		if (pin[d][f] == 1) {
			SetConsoleTextAttribute	(	h,	FOREGROUND_GREEN	|
FOREGROUND_INTENSITY );
			printf("%3d ",pin[d][f]);
			SetConsoleTextAttribute ( h, wOldColorAttrs);
		}
		else if(pin[d][f] == 0) {
			SetConsoleTextAttribute	(	h,	FOREGROUND_RED	|
FOREGROUND_INTENSITY );
			printf("%3d ",pin[d][f]);
			SetConsoleTextAttribute ( h, wOldColorAttrs);
		}
		else if(pin[d][f] == m*n) {
			SetConsoleTextAttribute	(	h,	FOREGROUND_BLUE	|
FOREGROUND_INTENSITY );
			printf("%3d ",pin[d][f]);
			SetConsoleTextAttribute ( h, wOldColorAttrs);
		}
		else {
			printf("%3d ",pin[d][f]);  //debug line
```

```
                }
        }
        printf("\n");  //debug line
}
```

Πίνακας 23 - Επισήμανση αρχικής και τελικής θέσης

6.4. Συνάρτηση problemKnight01_3

Αυτή η συνάρτηση είναι η πρώτη που χρησιμοποιεί και άλλες σειρές επιλογής σε περίπτωση ισοπαλιών εκτός της προεπιλεγόμενης. Η συγκεκριμένη δημιουργεί μια καινούργια τυχαία σειρά κάθε φορά που μία διαδρομή καταλήγει σε αδιέξοδο.

```
time_t srand ( time(NULL) );
ch=0;
reo=0;
do {
        if (ch>0) {
                for(int b=0;b<8;b++) {
                        order[b]=0;
                }
                for(int b=0;b<8;b++) {
                        do {
                                ok=0;
                                r=rand() % 8 + 1;
                                for(int c=0;c<8;c++)
                                        if (order[c]==r)
                                                ok=1;
                        } while (ok==1);
                        order[b]=r;
                }
                for(int b=0;b<8;b++) {
                        printf("%d" ,order[b]);
                }
                printf("\n");
                reo++;
        }
```

Πίνακας 24 - Δημιουργία τυχαίας σειράς

Ο πίνακας order[] περιέχει την σειρά που λύνονται οι ισοπαλίες. Για να υπολογιστεί μία τυχαία, πρώτα μηδενίζουμε τον πίνακα και μετά τοποθετούμε σε κάθε θέση του ένα τυχαίο νούμερο από 1 έως το 8. Σε κάθε θέση μετά την πρώτη τσεκάρουμε εάν το νούμερο που θέλουμε να τοποθετήσουμε υπάρχει σε κάποια προηγούμενη θέση, εάν υπάρχει τότε παίρνουμε έναν άλλο τυχαίο αριθμό και επαναλαμβάνουμε την διαδικασία μέχρι να βρούμε κάποιον που δεν έχει χρησιμοποιηθεί ήδη.

```
if (check==0) {
                        ch=10;
                }
                else {
                        ch++;
                }
        } while (ch<10);
        if(check==0) {
                valids++;
        }
        else {
                nonvalids++;
```

```
	}
```

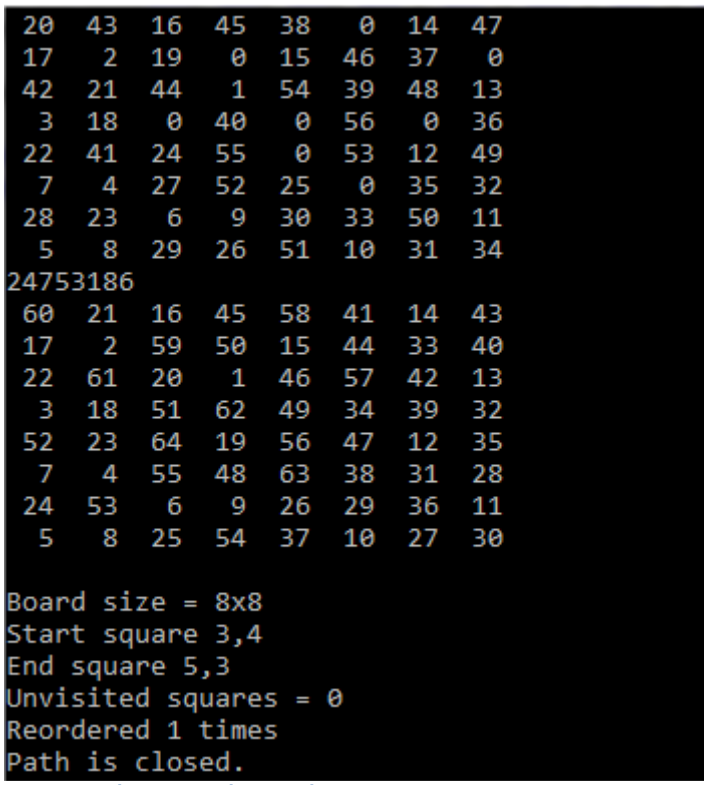
Πίνακας 25 - Επανάληψη εύρεσης διαδρομής με καινούργια τυχαία σειρά

Η μεταβλητή check μηδενίζεται στην αρχή της συνάρτησης και εάν παραμείνει μηδέν τότε έχει βρεθεί μία ολοκληρωμένη διαδρομή του ίππου και κάνει την ch ίση με 10 για να βγει το πρόγραμμα από την "do…while" επανάληψη. Εάν δεν είναι μηδέν τότε ανεβάζει την ch κατά ένα και επαναλαμβάνει την εύρεση διαδρομής. Αυτό γίνεται το πολύ δέκα φορές στην συγκεκριμένη περίπτωση αλλά μπορούμε να το κάνουμε όσες φορές θέλουμε. Εάν βρεθεί ολοκληρωμένη διαδρομή τότε η valids ανεβαίνει κατά ένα αλλιώς ανεβαίνει η nonvalids, για να έχουμε στο τέλος στατιστικά στοιχεία για τις διαδρομές που βρέθηκαν.

```
printf("\nBoard size = %dx%d\nStart square %d,%d\nEnd square %d,%d\nUnvisited
squares = %d\nReordered %d times",m,n,startx,starty,xmax+1, ymax+1,check,reo);
	if ((abs(startx-1-xmax) == 1 && abs(starty-1-ymax) == 2) || (abs(startx-
1-xmax) == 2 && abs(starty-1-ymax) == 1)) {
		printf("\nPath is closed.");
	}
	else {
		printf("\nPath is open.");
	}
}
```

Πίνακας 26 - Εκτύπωση δεδομένων

Στο τέλος τυπώνεται στο command window το μέγεθος της σκακιέρας, η αρχική και τελική θέση, πόσα τετράγωνα δεν επισπεύτηκε ο ίππος, πόσες φορές υπολογίστηκε άλλη σειρά επίλυσης ισοπαλιών και το εάν η διαδρομή είναι κλειστή η όχι.



Εικόνα 9 - Παράδειγμα τυχαίας αλλαγής σειράς

6.5. Συνάρτηση problemKnight02_1

Εδώ με μία επανάληψη for ψάχνουμε διαδρομές από όλες τις θέσεις μιας σκακιέρας και στο τέλος τυπώνει τον αριθμό των επιτυχών και ανεπιτυχών διαδρομών και το ποσοστό τους επί της εκατό.

```
for(i=0;i<m*n-1;i++) {
	xtemp=0;
	ytemp=0;
	xmax=0;
	ymax=0;
	invalidMove=0;
	moves=9;
	for(int e=0;e<8;e++) {
		xtemp=x+xmove[order[e]-1];
		ytemp=y+ymove[order[e]-1];
		move_check(n, xtemp, ytemp, &xmax, &ymax, &moves, &invalidMove,
*pin);
	}
	if(invalidMove==8)
		break;
	x=xmax;
	y=ymax;
	pin[x][y]=101+i;
	for(int e=0;e<8;e++) {
		xtemp=x+xmove[e];
		ytemp=y+ymove[e];
		board_respec(n,xtemp, ytemp, *pin);
	}
}
```

Πίνακας 27 - Επανάληψη για πολλαπλές σκακιέρες

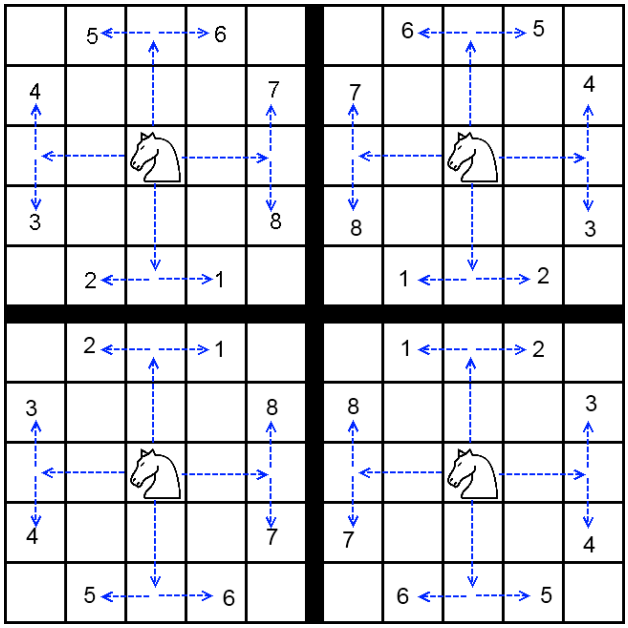
6.6. Συνάρτηση problemKnight02_2

Η συνάρτηση problemKnight02_2 μας επιτρέπει να της δώσουμε συγκεκριμένες σειρές οι οποίες θα χρησιμοποιηθούν η μία μετά την άλλη σε περίπτωση που δεν βρεθεί ολοκληρωμένη διαδρομή. Επίσης εδώ χρησιμοποιούμε την συμμετρία στης σκακιέρας προς όφελος και υπολογίζουμε μόνο τις διαδρομές που ξεκινάνε από το πρώτο τέταρτο της σκακιέρας.

```
for(xrep=0;xrep<m/2;xrep++) {
	for(yrep=0;yrep<n/2;yrep++) {
```

Πίνακας 28 - Διατρέχουμε μόνο το ένα τέταρτο της σκακιέρας

Η μέθοδος που επιλέγουμε την επόμενη θέση σε περίπτωση ισοπαλιών δεν επηρεάζεται από το ότι εξετάζουμε μόνο το ένα τέταρτο της σκακιέρας γιατί και η κάθε σειρά έχει την συμμετρική της, όπως φαίνεται από το παρακάτω σχήμα.



Εικόνα 10 - Συμμετρίες σειράς λύσεις ισοπαλιών

```
order[2][8]={1,2,3,4,5,6,7,8, 6,7,5,2,8,1,3,4};

for(int e=0;e<8;e++) {
	xtemp=x+xmove[order[z][e]-1];
	ytemp=y+ymove[order[z][e]-1];
	move_check(n, xtemp, ytemp, &xmax, &ymax, &moves, &invalidMove,*pin);
}
```

Πίνακας 29 - Εκτύπωση δεδομένων

Ο δισδιάστατος πίνακας order[][] περιέχει τις σειρές και η μεταβλητή z προσδιορίζει ποια από αυτές χρησιμοποιείτε. Η z αρχίζει μηδέν και αυξάνεται κατά ένα κάθε φορά που μία διαδρομή καταλήγει σε αδιέξοδο. Εάν δεν βρεθεί ολοκληρωμένη διαδρομή με τις παρεχόμενες σειρές τότε ο αλγόριθμος τα παρατάει και πάει να βρει διαδρομή για την επόμενη θέση της σκακιέρας.

6.7. Συνάρτηση problemKnight02_3

Για να μπορούμε να επεξεργαστούμε καλύτερα το σύνολο των αποτελεσμάτων από την αναζήτηση διαδρομών σε πολλαπλές σκακιέρες αποθηκεύουμε τα αποτελέσματα σε ένα αρχείο κειμένου. Γράφουμε, με την σειρά - από την μικρότερη σκακιέρα μέχρι την μεγαλύτερη, 0 εάν υπάρχει διαδρομή και 1 εάν δεν υπάρχει. Επίσης σε αυτή την συνάρτηση χρησιμοποιούμε την συμμετρία της σκακιέρας και υπολογίζουμε το ένα τέταρτο της κάθε μίας για υπολογισμό των διαδρομών.

Ένα παράδειγμα αποτελέσματος αυτής της συνάρτησης:
Έστω ότι θέλουμε να βρούμε τις διαδρομές που μπορούν να δημιουργηθούν για τις σκακιέρες 6x6 και 8x8 με σειρά επίλυσης ισοπαλιών 46531827. Το πρόγραμμα θα υπολογίσει με την σειρά τις διαδρομές και θα δώσει 0 εάν υπάρχει διαδρομή και 1 εάν δεν υπάρχει.

- για την 6x6 0 0 0
  0 0 0
  0 0 0
- για την 8x8 0 0 0 0
  0 0 0 0

0 1 0 0
0 0 0 0

και μετά θα γράψει τα αποτελέσματα στο αρχείο 'board46531827.txt' ως εξής: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

```
char fNum[15];
bool mat[300][300];
dName = "F:\\c_stats\\";
fName = "board";
exName = ".txt";
orders=1;
sprintf_s(fNum, "45631827");
fstr = dName + fName + fNum + exName;
```

Πίνακας 30 - Προετοιμασία μεταβλητών για την αποθήκευση αποτελεσμάτων

Στην fstr δημιουργείτε το όνομα του αρχείου στο οποίο αποθηκεύονται τα αποτελέσματα από τις dName, fName, exName και fNum. Στην fNum βάζουμε σε μορφή κειμένου την σειρά επίλυσης που χρησιμοποιούμε για να μπορούμε να δούμε από το όνομα του αρχείου τι ποιάς σειράς αποτελέσματα περιέχει.

```
std::ofstream os(fstr, std::ios_base::app);
for(int d=0;d<board/2;d++) {
	for(int f=0;f<board/2;f++) {
		os << mat[d][f]<<" ";
	}
}
```

Πίνακας 31 - Αποθήκευση αποτελεσμάτων

Τα αποτελέσματα κάθε σκακιέρας αποθηκεύονται στον πίνακα mat[][] και μετά γράφονται, προσθετικά, στο αρχείο κειμένου που έχουμε ορίσει.

6.8. Συνάρτηση problemKnight03_1

Εάν βρεθεί μία ανολοκλήρωτη διαδρομή σε κάποια θέση τότε υπολογίζεται μία τυχαία καινούργια σειρά επίλυσης ισοπαλιών και εφαρμόζεται στην ίδια θέση. Αυτό επαναλαμβάνεται μέχρι να βρεθεί μία ολοκληρωμένη διαδρομή είτε μέχρι η προσπάθεια να γίνει έναν ορισμένο αριθμό φορών, 10 στην συγκεκριμένη περίπτωση.

Η συνάρτηση problemKnight03_1 προσπαθεί να βρει τις καλύτερες διαδρομές και για σκακιέρες με μονό αριθμό τετραγώνων για τα τετράγωνα με το ελάσσων χρώμα στην σκακιέρα. Για παράδειγμα, στην σκακιέρα 9x9 με τα μαύρα να είναι 41 και τα άσπρα 40 βρίσκει διαδρομές με αφετηρία τα άσπρα οι οποίες διανύουν 80 θέσεις, δεν περνάνε δηλαδή από ένα μαύρο.

```
for(xrep=0;xrep<m;xrep++) {
	for(yrep=0;yrep<n;yrep++) {
		ch=0;
		do {
			.
			.
			.
			.
			.
			if (check==0) {
				ch=10;
			}
			else if (check==1 && m%2==1) {
```

```
                        ch=10;
                }
                else {
                        ch++;
                }
        } while (ch<10);
```

Πίνακας 32 - Επαναϋπολογισμός διαδρομής

6.9. Συνάρτηση problemKnight03_2

Η συνάρτηση problemKnight03_2 υπολογίζει τις διαδρομές των σκακιέρων που του δίνουμε και δημιουργεί έναν φάκελο για κάθε σκακιέρα και αποθηκεύει μέσα σε αυτόν τον φάκελο την κάθε διαδρομή σε δικό της αρχείο κειμένου.

```
char fNum[10];
dName = "E:\\outputs\\board";
fName = "\\output";
exName = ".txt";
_mkdir("E:\\outputs");
sprintf_s(fNum, "%d", board);
dstr = dName + fNum;
_mkdir( dstr.c_str() );
```

Πίνακας 33 - Προετοιμασία μεταβλητών και δημιουργία διαδρομής αρχείου

Τα αρχεία με τα αποτελέσματα βρίσκονται μέσα στο φάκελο της κάθε σκακιέρας και όλοι οι φάκελοι βρίσκονται μέσα στον φάκελο outputs ο οποίος εάν δεν υπάρχει τον δημιουργεί η συνάρτηση _mkdir().

```
sprintf_s(fNum, "%d", ++p);
fstr = dstr + fName + fNum + exName;
std::ofstream os(fstr);
for(int d=0;d<board;d++) {
        for(int f=0;f<board;f++) {
                os << pin[d][f]<<" ";
        }
        os <<"\n";
}
```

Πίνακας 34 - Αποθήκευση αποτελεσμάτων

6.10. Συνάρτηση problemKnight03_3

Εδώ ορίζουμε ένα εύρος διαστάσεων σκακιέρων και μια σειρά επίλυσης ισοπαλιών και μετά η συνάρτηση υπολογίζει τις διαδρομές για κάθε σκακιέρα. Εάν κάποια διαδρομή είναι ανολοκλήρωτη τότε υπολογίζεται μια τυχαία διαδρομή και γίνεται αυτό μέχρι να βρεθεί μία ολοκληρωμένη ή μέχρι να φτάσει η επανάληψη αυτή έναν συγκεκριμένο αριθμό.

```
float stats[300][4];
time_t srand ( time(NULL) );
time_t start,end;
double dif;
totalReorders=0;
totalNonvalids=0;
```

```
start = clock();
```

Πίνακας 35 - Αρχικοποίηση μεταβλητών και ξεκίνημα ρολογιού

Ο πίνακας stats[][] αποτελείτε από 4 στήλες. Η πρώτη περιέχει τις διαστάσεις της σκακιέρας, η δεύτερη ένα νούμερο που δείχνει πόσες φορές δημιουργήθηκαν τυχαίες σειρές για την κάθε σκακιέρα, η τρίτη πόσες διαδρομές τελικά μείνανε ανολοκλήρωτες και η τέταρτη πόση ώρα, σε δευτερόλεπτα, πείρε να υπολογισθούν οι διαδρομές για κάθε σκακιέρα.

```
end = clock();
dif = double(end - start);
stats[(board-6)/2][1]=float(board);
stats[(board-6)/2][2]=float(reorders);
stats[(board-6)/2][3]=float(nonvalids);
stats[(board-6)/2][4]=float(dif/1000);
printf("%3d %3d %3d %.3f \n",board,reorders,nonvalids,dif/1000);
totalNonvalids +=nonvalids;
totalReorders +=reorders;
std::ofstream os("stats.txt");
for (i=0;i<=(m-6)/2;i++) {
	os << stats[i][1] << " " << stats[i][2] << " " << stats[i][3] << " " <<
stats[i][4] << "\n";
}
printf("Total reorders = %d\n",totalReorders);
printf("Total nonvalids = %d\n",totalNonvalids);
```

Πίνακας 36 - Διακοπή ρολογιού και αποθήκευση στατιστικών στοιχείων

6.11. Συνάρτηση problemKnight03_4

Παίρνουμε τυχαίες σειρές και υπολογίζουμε τις διαδρομές για ένα προκαθορισμένο πλήθος σκακιέρων. Οριοθετούμε το πόσες ανολοκλήρωτες διαδρομές μπορεί μα έχει μία σειρά. Εάν ξεπεραστεί αυτό το όριο τότε οι διαδρομές που έχουν υπολογιστεί για αυτή τη σειρά μέχρι εκείνη τη στιγμή δεν αποθηκεύονται. Τα αρχεία που αποθηκεύονται είναι ένα αρχείο για κάθε σειρά και ένα που περιέχει στατιστικά στοιχεία για όσες σειρές βγάλανε αποδεκτά αποτελέσματα.

```
for (w=0;w<100;w++) {
	for(int b=0;b<8;b++) {
		order[b]=0;
	}
	for(int b=0;b<8;b++) {
		do {
			ok=0;
			r=rand() % 8 + 1;
			for(int c=0;c<8;c++)
				if (order[c]==r)
					ok=1;
		} while (ok==1);
		order[b]=r;
	}
	.
	.
	.
		if (totalNonvalids>1000)
			break;
	}
```

```
	if (totalNonvalids<=1000) {
		sprintf_s(fNum, "%d", p);
		fstr =dName + fName + fNum + exName;
		std::ofstream os(fstr);
		for (i=0;i<=(m-6)/2;i++) {
			os << stats[i][1] << " " << stats[i][2] << " " << 
stats[i][3] <<"\n";
		}
	printf("Round %3d. Total nonvalids = %4d\n",p,totalNonvalids);
		std::fstream		os2("F:\\c_stats\\orders_of_boards.txt", 
std::ios_base::app);
		os2 << p << " " << totalNonvalids << " ";
		for (j=0;j<=7;j++) {
			os2 << order[j] << " ";
		}
		os2 << "\n";
		p++;
	}
}
```

Πίνακας 37 - Υπολογισμοί διαδρομών με τυχαία σειρά επίλυσης ισοπαλιών και αποθήκευση στατιστικών στοιχείων.

6.12. Συνάρτηση order_sequence

Για να πάρουμε αποτελέσματα από συγκεκριμένες σειρές επίλυσης ισοπαλιών χρειαζόμαστε κάποιον τρόπο να βρίσκουμε τις σειρές αυτές αυτόματα, γιατί το πλήθος των σειρών είναι 8! = 40320 και είναι εξαιρετικά αντιπαραγωγική η χειροκίνητη εισαγωγή τους. Επίσης για να είναι καλύτερα ελεγχόμενα τα αποτελέσματα δεν θέλουμε να είναι τυχαία η επιλογή της σειράς όπως σε προηγούμενες συναρτήσεις. Η συνάρτηση αυτή δημιουργεί τις σειρές με πρώτη την 12345678 και τελευταία την 87654321 και δίνει και έναν αύξον αριθμό για την κάθε μία.

```
void order_sequence(int boardEnd, int from, int to)
```

Πίνακας 38 - Ορίσματα συνάρτησης order_sequence

Επιδή μέσα στην order_sequence() καλούμε μία άλλη συνάρτηση και της δίνει τις σειρές που υπολογίζει για την εύρεση διαδρομών, βάζουμε σαν πρώτο όρισμα την διάσταση που θέλουμε να φτάσουν οι σκακιέρες, το δεύτερο και το τρίτο είναι ένα πλαίσιο μέσα στο οποίο βρίσκονται οι αύξων αριθμοί των σειρών που θέλουμε να υπολογίσουμε.

```
int a,b,c,d,e,f,g,h,order[8],num=0;
bool ok;
```

Πίνακας 39 - Δηλώσεις μεταβλητών

Οι μεταβλητές a,b,c,d,e,f,g,h είναι οι προσωρινές θέσεις τον αριθμών της σειράς που ψάχνουμε και όταν γίνουν σίγουρες μπαίνουν στον πίνακα order[]. Η num περιέχει τον αύξων αριθμό των σειρών. Η ok είναι μια σωστό/λάθος μεταβλητή που βοηθάει στην εύρεση της σειράς.

```
for(a=1;a<=8;a++) {
	for(int k=0;k<8;k++) {
		order[k]=0;
	}
	order[0]=a;
.
```

```
.
.
```

Πίνακας 40 - Εύρεση ενός αριθμού της σειράς επίλυσης ισοπαλιών

Με μία επανάληψη for επιλέγουμε ένα - ένα τα νούμερα από το 1 έως το 8 και με μία εμφωλευμένη επανάληψη τσεκάρουμε εάν αυτό το νούμερο είναι ήδη στην σειρά.

```
if (ok==true) {
	num++;
	if (num>=from && num<=to)
		problemKnight04(boardEnd, order, num);
	}
}
```

Πίνακας 41 - Κλήση της problemKnight04() από την order_sequence()

Όταν βρεθεί μία σειρά, η num ανεβαίνει κατά ένα και εάν αυτή βρίσκετε μέσα στις σειρές που ζητάμε τότε καλείτε η συνάρτηση problemKnight04() με ορίσματα το μέγεθος των σκακιέρων που θέλουμε να εξετάσουμε, την σειρά με την οποία θα επιλυθούν οι ισοπαλίες και τον αύξον αριθμό της σειράς αυτής.

6.13. Συνάρτηση problemKnight04

Οι σειρές που παράγει η συνάρτηση order_sequence() δίνονται στην problemKnight04() για να υπολογίσει τις διαδρομές που μπορούν να ολοκληρωθούν για τις σκακιέρες από 6 έως ένα μέγεθος διάστασης σκακιέρας που θέλουμε.

```
void problemKnight04(int m, int* order, int p)
```

Πίνακας 42 - Ορίσματα συνάρτησης problemKnight04

Τα ορίσματα της συνάρτησης είναι το ανώτατο μέγεθος των σκακιέρων που θέλουμε να εξετάσουμε, την σειρά με την οποία θα επιλυθούν οι ισοπαλίες και ο αύξον αριθμός της σειράς αυτής.

```
dif = double(end - start);
stats[(board-6)/2][1]=float(board);
stats[(board-6)/2][2]=float(nonvalids);
stats[(board-6)/2][3]=float(dif/1000);
printf("%3d %3d %.3f ",board,nonvalids,dif/1000);
totalNonvalids +=nonvalids;
printf("%4d\n",totalNonvalids);
if (totalNonvalids>5000)
	break;
}
if (totalNonvalids<=5000) {
	sprintf_s(fNum, "%d", p);
	fstr =dName + fName + fNum + exName;
	std::ofstream os(fstr);
	for (i=0;i<=(m-6)/2;i++) {
		os << stats[i][1] << " " << stats[i][2] << " " << stats[i][3]
<<"\n";
	}
	printf("Round %3d. Total nonvalids = %4d\n",p,totalNonvalids);
	std::fstream os2("F:\\c_stats\\orders_of_boards.txt",
	std::ios_base::app);
	os2 << p << " " << totalNonvalids << " ";
```

```
	for (j=0;j<=7;j++) {
		os2 << order[j] << " ";
	}
	os2 << "\n";
```

Πίνακας 43 - Αποθήκευση αποτελεσμάτων της συνάρτησης problemKnight04

Η αποθήκευση των αποτελεσμάτων γίνεται σε δύο στάδια. Στο πρώτο δημιουργείτε ένα αρχείο που είναι ξεχωριστό για κάθε σειρά επίλυσης και αποτελείτε από τρεις στήλες, μία με τα μεγέθη των σκακιέρων, μία που λέει πόσες διαδρομές καταλήξανε σε αδιέξοδο και μία με τον χρόνο που πείρε το πρόγραμμα να υπολογίσει τις διαδρομές σε κάθε σκακιέρα. Στο δεύτερο αρχείο αποθηκεύονται σε μία γραμμή για κάθε σειρά, ο αύξων αριθμός της σειράς όπως αυτός της δόθηκε από την συνάρτηση order_sequence(), πόσες φορές απέτυχε ο αλγόριθμος να δώσει ολοκληρωμένη διαδρομή και την ίδια την διαδρομή.

6.14. Συνάρτηση board_setup

Εδώ δημιουργείτε ο πίνακας μέσα στον οποίο θα υπολογιστεί η διαδρομή. Αρχικά κάθε θέση περιέχει ένα νούμερο που αντιστοιχεί στον βαθμό της. Κατά το τρέξιμο του προγράμματος αντικαθιστούνται από την σειρά κατά την οποία πέρασε ο ίππος από την εκάστοτε θέση.

```
void board_setup(int n, int* pin) {
int a,b,c; //ari8mos pi8anwn khnhsewn
.
.
.
}
```

Πίνακας 44 - Ορίσματα και μεταβλητές συνάρτησης

Τα ορίσματα της συνάρτησης είναι το μέγεθος της σκακιέρας και ο δείκτης του πίνακα. Οι μεταβλητές κρατάνε προσωρινά τους βαθμούς των θέσεων πριν γραφούν στον πίνακα.

```
for(int i=0;i<m;i++) { //arxikopoihsh pinaka me oles tis pi8anes khnhseis apo
ka8e 8esh
	if((i==0)||(i==m-1)) {
		a=2;
		b=3;
		c=4;
	}
	else if((i==1)||(i==m-2)) {
		a=3;
		b=4;
		c=6;
	}
	else {
		a=4;
		b=6;
		c=8;
	}
	for(int j=0;j<m;j++) {
		if((j==0)||(j==m-1))
			*(pin+i+j*300)=a;
		else if((j==1)||(j==m-2))
```

```
                        *(pin+i+j*300)=b;
                else
                        *(pin+i+j*300)=c;
        }
}
```

Πίνακας 45 - Υπολογισμός και γράψιμο βαθμών στον πίνακα

Ο πίνακας γεμίζει ανά γραμμή μέσα σε μία επανάληψη for η οποία τρέχει τόσες φορές όσες είναι και οι γραμμές της σκακιέρας. Κάθε φορά βλέπουμε πόσο μακριά βρίσκεται η τρέχουσα γραμμή από την πάνω ή την κάτω άκρη της σκακιέρας και ανάλογα γράφουμε τον βαθμό που αντιστοιχεί στην κάθε θέση. Αυτό που κάνει τον τρόπο αυτό να λειτουργεί σωστά είναι ότι ο ίππος στις γωνίες και στις άκρες της σκακιέρας έχει πάντα τον ίδιο αριθμό κινήσεων ανεξάρτητα από το μέγεθος της σκακιέρας.

6.15. Συνάρτηση move_check

Για να υπολογιστεί η επόμενη κίνηση του ίππου τρέχουμε την move_check() η οποία καλείτε 8 φορές για να ελέγξει και τις 8 πιθανές κατευθύνσεις που μπορεί να πάρει η κίνηση. Έχει σαν ορίσματα αρκετούς δείκτες για να παίρνει κάθε επόμενο κάλεσμα της συνάρτησης ως ορίσματα τις μεταβλητές που άλλαξε το προηγούμενο.

Ενώ, στο πλαίσιο αυτής της εργασίας, λύνουμε πάντα τις ισοπαλίες με βάση την σειρά επίλυσης ισοπαλιών που δίνουμε στο πρόγραμμα, εδώ έχουμε και μία υλοποίηση της επίλυσης με βάση την απόσταση της θέσης από το κέντρο.

```
void move_check(int n, int xtemp, int ytemp, int* xmax, int* ymax, int* moves,
int* invalidMove, int* pin) {
int m = n, temp=0, max=0;
```

Πίνακας 46 - Ορίσματα και αρχικοποιήσεις μεταβλητών

- n: μέγεθος σκακιέρας
- xtemp, ytemp: προσωρινοί καταχωρητές των συντεταγμένων της επόμενης θέσης του ίππου
- xmax, ymax: δείκτες οι οποίοι κρατάνε την επικρατέστερη επόμενη θέση του ίππου που υπολογίστικε από προηγούμενα καλέσματα της συνάρτησης. Συγκρίνονται με τις xtemp και ytemp και γίνεται αντικατάστασή τους αν κρίνεται σκόπιμο.
- moves: ο βαθμός της θέσης που εξετάζεται
- invalidMove: πόσες ακατάλληλες θέσεις έχει βρει ο αλγόριθμος για αυτήν την κίνηση. Εάν η μεταβλητή αυτή φτάσει στο 8 τότε ο ίππος βρίσκεται σε αδιέξοδο.
- pin: ο δείκτης του πίνακα της σκακιέρας

```
if((xtemp>=0)&&(ytemp>=0)&&(xtemp<m)&&(ytemp<n)) {
        if((*(pin+ytemp+xtemp*300)<*moves)&&(*(pin+ytemp+xtemp*300)<10)){
                *moves=*(pin+ytemp+xtemp*300);
                *xmax=xtemp;
                *ymax=ytemp;
        }
        else
                (*invalidMove)++;
}
else
        (*invalidMove)++;
```

Πίνακας 47 - Σύγκριση για επιλογή επόμενης κίνησης

Στην αρχή βλέπουμε εάν η το τετράγωνο βρίσκεται μέσα στην σκακιέρα, δηλαδή εάν οι συντεταγμένες είναι μεγαλύτερες του μηδενός και μικρότερες του μεγέθους της σκακιέρας. Μετά πρέπει η διαδρομή του ίππου να μη έχει περάσει ήδη από το τετράγωνο αυτό και ο βαθμός του να είναι μικρότερος από τον βαθμό που πείρε σαν όρισμα η συνάρτηση. Εάν ισχύουν τα προηγούμενα τότε βάζουμε στις μεταβλητές που δείχνουν οι δείκτες xmax και ymax την θέση που εξετάζουμε τώρα.

Η invalidMove ανεβαίνει κατά ένα εάν η θέση είναι έξω από την σκακιέρα ή ήταν ήδη μέσα στην διαδρομή και όταν φτάσει στο 8 τότε σημαίνει ότι ο ίππος έφτασε σε αδιέξοδο. Εάν υπάρχουν ακόμα ελεύθερες θέσεις τότε η διαδρομή είναι ανολοκλήρωτη, αλλιώς είναι κλειστή ή ανοιχτή ολοκληρωμένη διαδρομή.

```
if(*(pin+ytemp+xtemp*300)==*moves) {
	if (xtemp<=(m-1)/2) {
		temp += xtemp;
	}
	else {
		temp += (m-1)-xtemp;
	}
	if (ytemp<=(m-1)/2) {
		temp += ytemp;
	}
	else {
		temp += (m-1)-ytemp;
	}
	if (*xmax<=(m-1)/2) {
		max += *xmax;
	}
	else {
		max += (m-1)-*xmax;
	}
	if (*ymax<=(m-1)/2) {
		max += *ymax;
	}
	else {
		max += (m-1)-*ymax;
	}
	if(temp<max) {
		*moves=*(pin+ytemp+xtemp*300);
		*xmax=xtemp;
		*ymax=ytemp;
	}
}
```

Πίνακας 48 - Επιλογή επόμενης θέσης με βάση την απόσταση από τις γωνίες της σκακιέρας

Ένας εναλλακτικός τρόπος να υπολογίσουμε την επόμενη κίνηση του ίππου είναι να επιλέγουμε από τις ισόπαλες θέσεις αυτή που βρίσκεται ποιο μακριά από το κέντρο ή αλλιώς αυτή που είναι ποιο κοντά στις άκρες της σκακιέρας. Για να το υπολογίσουμε αυτό χρησιμοποιούμε την απόσταση Μανχάταν της θέσης από την ποιο κοντινή γωνία της σκακιέρας σε αυτή. Η απόσταση Μανχάταν ισούται με τον αριθμό των κινήσεων που θα έκανε το πιόνι του βασιλιά να πάει από την θέση που εξετάζεται μέχρι την γωνία.

6.16. Συνάρτηση board_respec

Μετά από κάθε κίνηση του ίππου οι βαθμοί των τετραγώνων γύρω από την θέση που καταλαμβάνει τώρα ο ίππος αλλάζουν, συγκεκριμένα μειώνονται κατά ένα. Αυτό γίνεται με την κλήση της συνάρτησης board_respec. Αυτό που κάνει είναι να καλείτε 8 φορές, μία φορά για κάθε πιθανή κατεύθυνση που μπορεί να πάρει ο ίππος, και εάν το τετράγωνο αυτό βρίσκεται μέσα στην σκακιέρα και δεν είναι μέρος της διαδρομής μειώνει τον βαθμό του κατά ένα.

```
void board_respec(int n, int xtemp, int ytemp, int* pin)
```

Πίνακας 49 - Ορίσματα της board_respec

- n: μέγεθος σκακιέρας
- xtemp, ytemp: η θέση του παρακείμενου τετραγώνου σε αυτό του ίππου που θα μειώσουμε τον βαθμό
- pin: δείκτης του πίνακα που περιέχει την σκακιέρα

```
	if((xtemp<m)&&(ytemp<m)&&(xtemp>=0)&&(ytemp>=0)) {
		if(*(pin+ytemp+xtemp*300)<10) {
			*(pin+ytemp+xtemp*300)-=1;
		}
	}
```

Πίνακας 50 - Επαναϋπολογισμός βαθμών σκακιέρας

Η συνάρτηση πρώτα ελέγχει εάν το τετράγωνο βρίσκετε μέσα στην σκακιέρα και μετά εάν είναι εκτός της διαδρομής που έχει κάνει ήδη ο ίππος. Εάν ισχύουν τα προηγούμενα τότε μειώνουμε τον βαθμό της θέσης κατά ένα.

7. Ανάλυση κώδικα Matlab για την οπτικοποίηση διαδρομών και στατιστικών

Το Matlab προσφέρει πολλά εργαλεία για την εύκολη επεξεργασία των δεδομένων που έχουμε από τον υπολογισμό των διαδρομών του ίππου και των διάφορων σειρών επίλυσης ισοπαλιών που έχουν βγάλει καλά αποτελέσματα. Συγκεκριμένα είναι εύκολο να σχεδιάσουμε τις διαδρομές και να οπτικοποιήσουμε τα αποτελέσματα των σειρών για ευκολότερη κατανόηση και επεξεργασία.

7.1. Οπτικοποίηση διαδρομής

Ενώ η γραφή μίας διαδρομής με νούμερα σε μορφή πίνακα είναι αρκετή για να την διαβάσει και να την επεξεργαστεί ένα πρόγραμμα ο καλύτερος τρόπος να την κατανοήσει ένας άνθρωπος είναι να την αποτυπώσουμε πάνω σε μία σκακιέρα. Η κίνηση του ίππου αναπαριστάται με γραμμές οι οποίες ενώνουν συνεχόμενα τις θέσεις που περνάει η διαδρομή.

```
file = (' F:\output.txt');
mat = importdata(file);
l=length(mat);
t=l*l;
```

Πίνακας 51 - εισαγωγή διαδρομής από αρχείο

Εισάγουμε το αρχείο που περιέχει την διαδρομή σε μία μεταβλητή – πίνακα, την mat, και το μέγεθος της σκακιέρας σε μία άλλη, την l.

```
img = (checkerboard(40,int16(l/2),int16(l/2)) > 0.5);
if (mod(l/2,1)~=0)
    cr = size(img)-40;
    img = imcrop(img, [0 0 cr]);
end
figure('Name',['position']), imshow(img);
hold on
```

Πίνακας 52 - δημιουργία της σκακιέρας και αποτύπωση στην οθόνη

Δημιουργείτε μία εικόνα, η img, της σκακιέρας με την συνάρτηση checkerboard() του Matlab. Εάν η σκακιέρα είναι μονού μεγέθους τότε 'κόβουμε' μία γραμμή και μία στήλη από την σκακιέρα, με την συνάρτηση imgcrop, γιατί η checkerboard() σχεδιάζει μόνο ζυγές σκακιέρες. Μετά εμφανίζουμε την σκακιέρα στην οθόνη και με την εντολή 'hold on' θα μπορέσουμε να σχεδιάσουμε ότι άλλο θέλουμε πάνω της.

```
for j=1:t-1
   [row1,col1] = find(mat==j);
   [row2,col2] = find(mat==j+1);
   line([col1*40-20 col2*40-20],[row1*40-20 row2*40-
20],'LineWidth',2,'Color',[0 0 .7]);
   if (j==1)
       plot(col1*40-20,row1*40-20,'g.','MarkerSize',30);
   else
       plot(col1*40-20,row1*40-20,'m.','MarkerSize',5);
   end
   if (j==(l*l)-1)
       plot(col2*40-20,row2*40-20,'r.','MarkerSize',30);
```

```
    end
end
```

Πίνακας 53 - σχεδίαση γραμμών και σημείων της διαδρομής

Με μία επανάληψη for που τρέχει τόσες φορές όσα είναι και τετράγωνα της σκακιέρας βρίσκει τις κινήσεις του ίππου από την πρώτη μέχρι την τελευταία και σχεδιάζει μία γραμμή μεταξύ των κέντρων των τετραγώνων της κίνησης. Στο κέντρο κάθε τετραγώνου που είναι μέρος της διαδρομής σχεδιάζεται μία μοβ τελεία για την καλύτερη αναπαράσταση της διαδρομής και στην πρώτη θέση σχεδιάζει μία πράσινη τελεία και στην τελευταία μία κόκκινη.

7.2. Διάγραμμα ανολοκλήρωτων διαδρομών

Παρουσιάζουμε σε μορφή γραφήματος τις ανολοκλήρωτες διαδρομές που βρέθηκαν στις σκακιέρες για την κάθε σειρά, με διαφορετικό χρώμα για άμεση οπτική σύγκριση τους. Το χρώμα δίνετε με την μορφή [R,G,B] και υπολογίζεται με τρεις επαναλήψεις for και αποθηκεύονται σε έναν πίνακα color[][][]. Εισάγουμε τα δεδομένα σε έναν πίνακα και μετά σχεδιάζουμε όλα τα γραφήματα σε μία εικόνα και την αποτυπώνουμε στην οθόνη.

```
clc, close all;
pin(48,3,11)=zeros();
colors=zeros([18 3]);
m=1;
for i=0:256:256
    for j=0:128:256
        for k=0:128:256
           colors(m,:)= [1/256*i 1/256*j 1/256*k];
            m=m+1;
        end
    end
end
file = 'F:\c_stats\good ones\orders_of_boards.txt';
k = importdata(file);
hold on;
for i=0:10
    file = ['F:\c_stats\good ones\stats' int2str(i) '.txt'];
    mat = importdata(file);
    pin(:,:,i+1)=mat;
    plot(pin(:,1,i+1),pin(:,2,i+1),'Color',colors(i+1,:));
end
xlabel('board dimensions (NxN)');
ylabel('time in seconds');
legend(int2str(k(1,3:10)), int2str(k(2,3:10)), int2str(k(3,3:10)),...
       int2str(k(4,3:10)), int2str(k(5,3:10)), int2str(k(6,3:10)),...
       int2str(k(7,3:10)), int2str(k(8,3:10)), int2str(k(9,3:10)),...
       int2str(k(10,3:10)), int2str(k(11,3:10)), 'Location', 'west')
```

Πίνακας 54 – σχεδίαση διαγράμματος ανολοκλήρωτων διαδρομών

7.3. Εισαγωγή και μορφοποίηση στατιστικών στοιχείων σειρών επίλυσης ισοπαλιών

Ο τρόπος που αποθηκεύονται οι σειρές επίλυσης ισοπαλιών σε αρχείο κειμένου από την συνάρτηση είναι τέτοιος ώστε η Matlab να μη μπορεί να πάρει τις σειρές σαν

οκταψήφια νούμερα, τα οποία θα απλοποιούσαν τον κώδικα, αλλά σαν οκτώ μονοψήφιους αριθμούς. Η παρακάτω συνάρτηση λύνει αυτό το πρόβλημα κάνοντας τις πρώτες οχτώ στήλες του πίνακα pin που περιέχουν τα μονοψήφια νούμερα, μία στήλη που περιέχει τις σειρές επίλυσης ισοπαλιών σαν οκταψήφιους αριθμούς.

```
File = ('D:\c_stats\orders_of_boards.txt');
pin = importdata(file);
pin = sortrows(pin,2);
pin2 = pin(:,10);
pin2 = pin2+pin(:,9)*10;
pin2 = pin2+pin(:,8)*100;
pin2 = pin2+pin(:,7)*1000;
pin2 = pin2+pin(:,6)*10000;
pin2 = pin2+pin(:,5)*100000;
pin2 = pin2+pin(:,4)*1000000;
pin2 = pin2+pin(:,3)*10000000;
pin3 = pin(:,1:2);
pin3(:,3)=pin2;
pin=pin3;
```

Πίνακας 55 - Εισαγωγή και μορφοποίηση στατιστικών στοιχείων σειρών επίλυσης ισοπαλιών.

7.4. Σχεδίαση γραφήματος αποτελεσμάτων σειρών επίλυσης ισοπαλιών

Με τα αποτελέσματα κάποιον σειρών επίλυσης ισοπαλιών σωσμένα σε αρχεία κειμένου χρησιμοποιούμε τον παρακάτω αλγόριθμο για να αποτυπώσουμε σε ένα γράφημα τις αστοχίες της εκάστοτε σειράς για όσες σκακιέρες υπάρχουν αποτελέσματα. Επίσης προστίθενται τα αποτελέσματα των σειρών με τέτοιο τρόπο ώστε να φαίνεται το πλήθος των αφετηριών των ανολοκλήρωτων διαδρομών και σε ποιες σκακιέρες βρίσκονται. Με αυτόν τον τρόπο μπορούμε να βρούμε τις λιγότερες σειρές που χρειάζονται για να επιλύσουν πλήρως τις περισσότερες σκακιέρες.

```
file = ('D:\c_stats\board46531827.txt');
mat2 = importdata(file);
l=1;
m=0;
figure
hold on
for i=6:2:300
    t=(i/2)*(i/2);
    n = nnz(mat2(l:l+t-1));
    l=l+t;
    plot(i,n,'-.ob')
    m=m+n;
end

file = ('D:\c_stats\board54361827.txt');
mat3 = importdata(file);
l=1;
m=0;
for i=6:2:300
    t=(i/2)*(i/2);
    n = nnz(mat3(l:l+t-1));
    l=l+t;
    plot(i,n,'-.or')
    m=m+n;
end
```

```
matall = mat2 & mat3;
l=1;
m=0;
for i=6:2:300
    t=(i/2)*(i/2);
    n = nnz(matall(l:l+t-1));
    l=l+t;
    plot(i,n,'-.og')
    m=m+n;
end
```

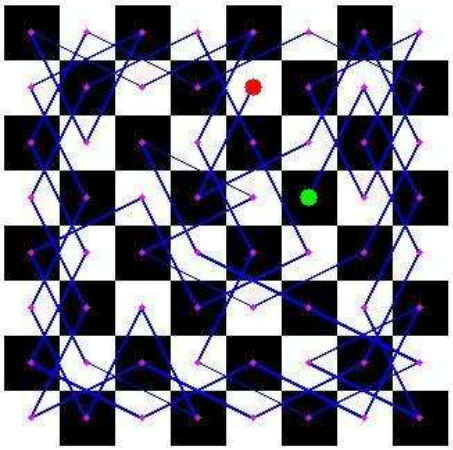
Πίνακας 56 - Σχεδίαση γραφήματος αποτελεσμάτων σειρών επίλυσης ισοπαλιών.

8. Αποτελέσματα της υλοποίησης και παρατηρήσεις

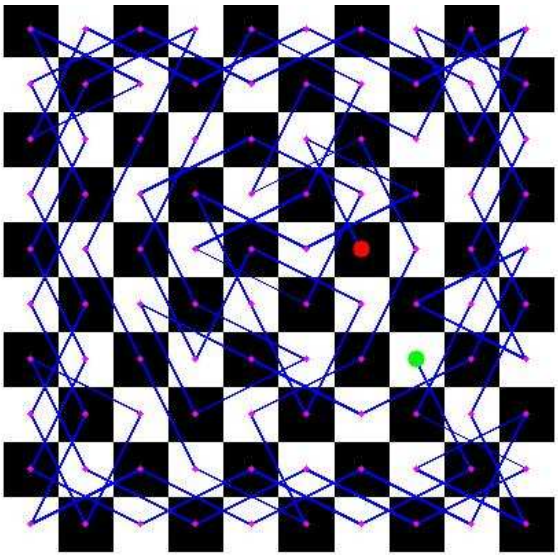
Η αποτελεσματικότητα του αλγορίθμου του Warnsdorff φαίνεται να εξαρτάτε άμεσα από την μέθοδο επίλυσης των ισοπαλιών που εφαρμόζουμε. Για αυτό θα αναλύσουμε τα αποτελέσματα του αλγορίθμου σε σχέση με την μέθοδο που χρησιμοποιούμε.

8.1. Αρχικά αποτελέσματα

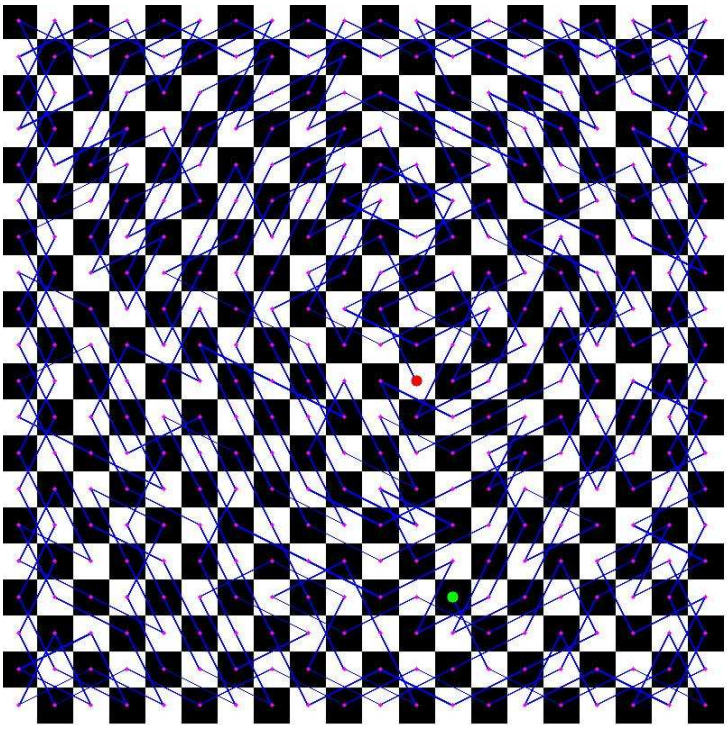
Παρακάτω παραθέτουμε κάποιες απεικονίσεις ολοκληρωμένων διαδρομών για διάφορες σκακιέρες ως παραδείγματα των αποτελεσμάτων που βγάζει ο αλγόριθμος υλοποίησης του κανόνα του Warnsdorff.



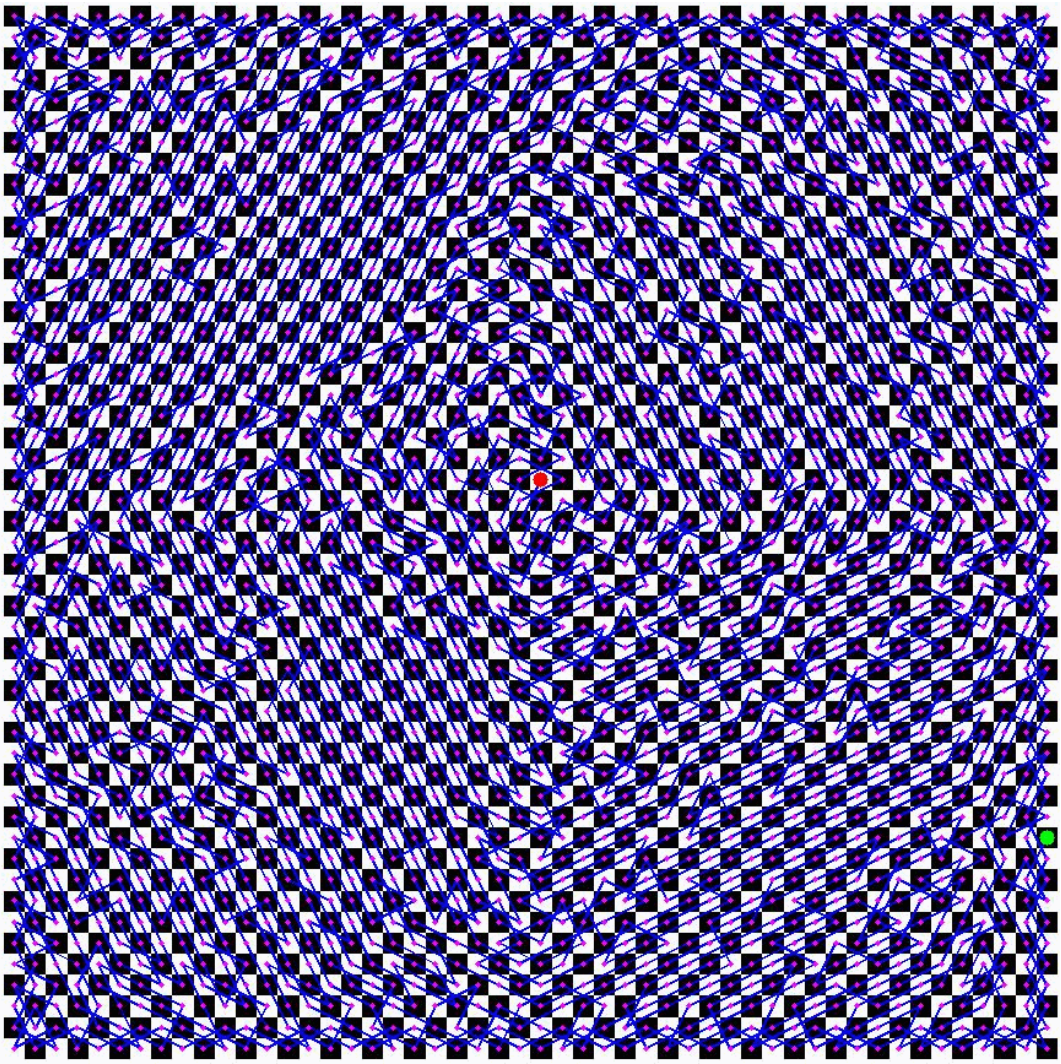
Εικόνα 11 - Σκακιέρα 8x8



Εικόνα 12 - Σκακιέρα 10x10



Εικόνα 13 - Σκακιέρα 20x20



Εικόνα 14 - Σκακιέρα 50*50

Δυστυχώς για τις σκακιέρες μεγαλύτερου μεγέθους οι εικόνες είναι πολύ μικρές για να δείξουν με κάποια ευκρίνεια την διαδρομή του ίππου.

Παρακάτω ακολουθούν στιγμιότυπα του command window που δείχνουν, για μία συγκεκριμένη σειρά επίλυσης ισοπαλιών, από πόσες και από ποιές θέσεις ξεκινάνε ολοκληρωμένες διαδρομές. Στα τετράγωνα από τα οποία ξεκινάνε ολοκληρωμένες διαδρομές γράφουμε τον αριθμό μηδέν και στα τετράγωνα που ξεκίνησαν ανολοκλήρωτες διαδρομές έναν αριθμό που αντιπροσωπεύει το πόσες θέσεις δεν επισκέφτηκε ο ίππος.

```
  0  0  0  4  0  0  0  0
  0  0  0  0  6  4  0  0
  0  0  0  0  0  0 20  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  8  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0

valids=59
nonvalids=5

Apotuxia 7.812500 ths ekato
```

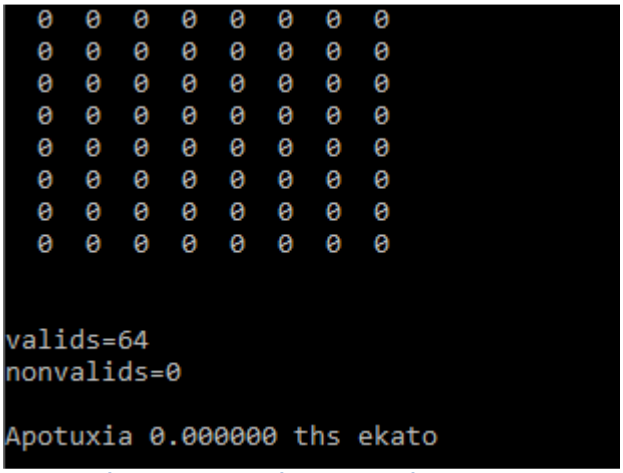
```
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  4  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
 14  0  0  0  0  0  0  0
  0  6  0  0  0  0  0  0
  0  6  0  0  0  0  0  0

valids=60
nonvalids=4

Apotuxia 6.250000 ths ekato
```

Εικόνα 15 - Σκακιέρες 8x8, σειρά 53461278 (αριστερά) και σειρά 46531827 (δεξιά)

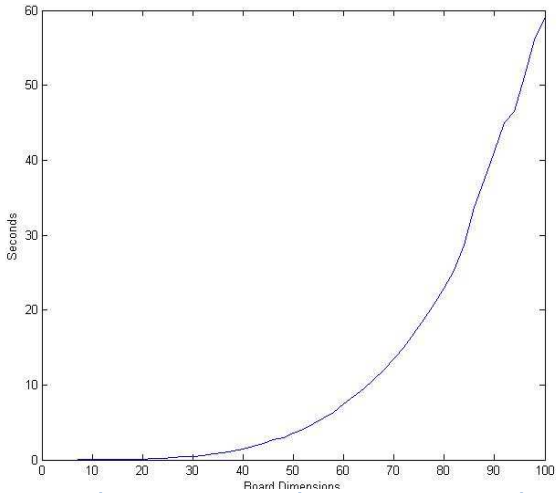
Εικόνα 16 - Σκακιέρα 10x10, σειρά 12345678



Εικόνα 17 - Σκακιέρα 8x8, σειρά 13472568, πλήρης επιτυχία

Τα αποτελέσματα αυτά είναι ενδεικτικά των μεγαλύτερων σκακιέρων οι οποίες βγάζουν ανάλογα αποτελέσματα και ποσοστά επιτυχίας με τις μικρότερες.

Ο χρόνος περαίωσης των υπολογισμών κυμαίνετε ανάλογα με το είδος των αποτελεσμάτων που θέλουμε. Ο υπολογισμός μίας διαδρομής στην 8x8 σκακιέρα είναι πιο γρήγορη από 1 χιλιοστό του δευτερολέπτου και όλων των διαδρομών κάνει περίπου 0,02 δευτερόλεπτα. Μία διαδρομή στην 300x300 κάνει περίπου 0.06 δευτερόλεπτα ενώ ο υπολογισμός όλων των διαδρομών 90 λεπτά.

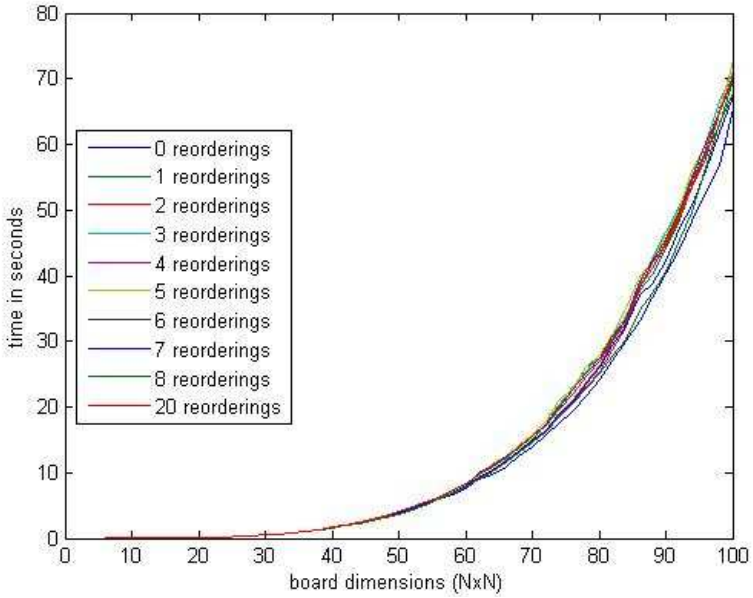


Εικόνα 18 - Τυπικός χρόνος περαίωσης υπολογισμών για 6≤N≤100 ζυγό.

8.2. Εξασφάλιση ολοκληρωμένων διαδρομών

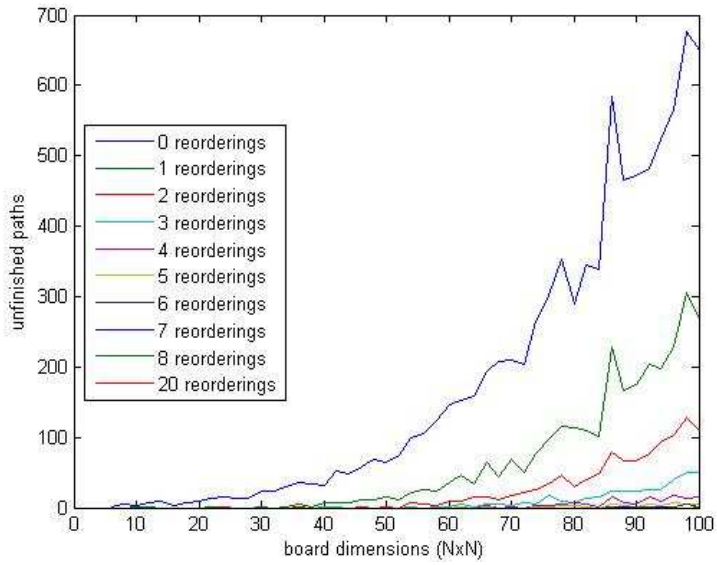
Ο στόχος οποιουδήποτε αλγορίθμου που προσπαθεί να λύσει το πρόβλημα του ίππου είναι να βρίσκει σε κάθε προσπάθεια του μια ολοκληρωμένη διαδρομή. Ο κανόνας του Warnsdorff, με τον τρόπο που υλοποιείτε στην παρούσα εργασία, μπορεί να βρει ολοκληρωμένες διαδρομές από όλες τις αφετηρίες σε τετράγωνες σκακιέρες από 5x5 μέχρι 300x300 (πλην τις ζυγές αφετηρίες σε μονές σκακιέρες όπως διευκρινίσαμε στο κεφάλαιο 3). Ο βασικός λόγος που γίνετε αυτό είναι επειδή όποια και να είναι η αφετηρία ο ίππος κατευθύνετε προς μια άκρη της σκακιέρας και μετά προχωράει προς τα μέσα καλύπτοντάς την χωρίς να αφήνει κενά. Αυτό μόνο όμως δεν εξασφαλίζει την εύρεση ολοκληρωμένης διαδρομής γιατί κάποιες φορές στην προσπάθεια να μην δημιουργούνται απομονωμένες περιοχές τετραγώνων που δεν είναι μέρος της διαδρομής, η διαδρομή καταλήγει σε αδιέξοδο. Σε τέτοιες περιπτώσεις χρησιμοποιούμε την σειρά που έχουμε επιλέξει για όσες διαδρομές ολοκληρώνονται και για τις υπόλοιπες χρησιμοποιούμε άλλες σειρές οι οποίες μπορούν να δώσουν ολοκληρωμένες διαδρομές.

Ο τρόπος με τον οποίο επιλέγονται οι εναλλακτικές σειρές μπορεί να είναι τυχαίος χωρίς μεγάλη χρονική επιβάρυνση για τους υπολογισμούς. Ακολουθεί ένα γράφημα με τον χρόνο που χρειάζεται για να βρεθούν όσες διαδρομές γίνεται με μία σειρά για τις σκακιέρες από 6x6 μέχρι 100x100 και οι χρόνοι υπολογισμού των ίδιων διαδρομών με την χρήση τυχαίων σειρών για τις ανολοκλήρωτες διαδρομές που προκύπτουν.



Εικόνα 19 – Χρόνος ολοκλήρωσης υπολογισμού διαδρομών με διαστάσεις σκακιέρας NxN για 6≤N≤100

Τα αποτελέσματα της εικόνας 19 προέρχονται από τον υπολογισμό του μέσου χρόνου περάτωσης των υπολογισμών των διαδρομών του ίππου στις σκακιέρες 6x6 μέχρι 100x100 με την χρήση δέκα τυχαίων σειρών για τις λύσεις των ισοβαθμιών. Για κάθε αρχική σειρά πήραμε διαδοχικά από μία και αργότερα περισσότερες επιπλέον τυχαίες σειρές για να καλύψουμε τις ανολοκλήρωτες διαδρομές που παρουσιάζονται. Παρατηρούμε ότι, στην χειρότερη περίπτωση για την σκακιέρα 100x100, ο χρόνος αυξάνεται κατά πέντε δευτερόλεπτα, περίπου 6,5%, ποσοστό αποδεκτό για αυτήν την υλοποίηση.



Εικόνα 20- Πλήθος ανολοκλήρωτων διαδρομών μετά από τους επαναϋπολογισμούς για την σειρά '12345678'

Η μείωση του αριθμού των ανολοκλήρωτων διαδρομών είναι αρκετά μεγάλη με κάθε επιπλέων χρήση κάποιας τυχαίας σειράς. Όπως βλέπουμε την εικόνα 20, η μπλε γραμμή δείχνει το πλήθος των ανολοκλήρωτων διαδρομών σε κάθε σκακιέρα. Οι υπόλοιπες γραμμές δείχνουν πόσο γίνετε το πλήθος αυτό με την χρήση όλο και περισσότερων, τυχαίων, σειρών επίλυσης ισοβαθμιών σε αυτές τις διαδρομές. Περεταίρω δοκιμές με άλλες σειρές έδειξαν ότι το ποσοστό των ανολοκλήρωτων διαδρομών είναι περίπου 15% του συνόλου και μετά από εφαρμογή πέντε τυχαίων σειρών σε αυτές τις διαδρομές το ποσοστό αυτό πέφτει κάτω από 3% ενώ ο χρόνος ανεβαίνει μόνο κατά 7-8%.

8.3. Κυκλικές διαδρομές

Ο αλγόριθμος βρίσκει αρκετά εύκολα κυκλικές διαδρομές για τις σκακιέρες με 6≤N≤300 ζυγό. Χρησιμοποιώντας απλώς την πρώτη σειρά επίλυσης ισοπαλιών '12345678', βρίσκουμε τις παρακάτω κυκλικές διαδρομές από τις δοσμένες αφετηρίες για την κάθε σκακιέρα.

board: 6x 6 start: 1, 5 order:12345678	board:154x154 start: 1,146 order:12345678
board: 8x 8 start: 1, 2 order:12345678	board:156x156 start: 2,155 order:12345678
board: 10x 10 start: 1, 7 order:12345678	board:158x158 start: 1,152 order:12345678
board: 12x 12 start: 1, 11 order:12345678	board:160x160 start: 2,155 order:12345678
board: 14x 14 start: 1, 9 order:12345678	board:162x162 start: 2,158 order:12345678
board: 16x 16 start: 1, 7 order:12345678	board:164x164 start: 2,159 order:12345678
board: 18x 18 start: 1, 7 order:12345678	board:166x166 start: 2,162 order:12345678
board: 20x 20 start: 1, 16 order:12345678	board:168x168 start: 1,165 order:12345678
board: 22x 22 start: 2, 16 order:12345678	board:170x170 start: 1,165 order:12345678
board: 24x 24 start: 1, 18 order:12345678	board:172x172 start: 2,166 order:12345678
board: 26x 26 start: 1, 22 order:12345678	board:174x174 start: 3,169 order:12345678
board: 28x 28 start: 1, 22 order:12345678	board:176x176 start: 4,165 order:12345678
board: 30x 30 start: 1, 23 order:12345678	board:178x178 start: 2,173 order:12345678
board: 32x 32 start: 1, 28 order:12345678	board:180x180 start: 2,179 order:12345678
board: 34x 34 start: 1, 24 order:12345678	board:182x182 start: 3,182 order:12345678
board: 36x 36 start: 3, 31 order:12345678	board:184x184 start: 2,179 order:12345678
board: 38x 38 start: 1, 34 order:12345678	board:186x186 start: 2,181 order:12345678
board: 40x 40 start: 2, 39 order:12345678	board:188x188 start: 1,185 order:12345678
board: 42x 42 start: 1, 32 order:12345678	board:190x190 start: 1,184 order:12345678
board: 44x 44 start: 1, 42 order:12345678	board:192x192 start: 3,191 order:12345678
board: 46x 46 start: 1, 45 order:12345678	board:194x194 start: 6,191 order:12345678
board: 48x 48 start: 1, 44 order:12345678	board:196x196 start: 1,193 order:12345678

board: 50x 50 start: 1, 44 order:12345678	board:198x198 start: 5,192 order:12345678
board: 52x 52 start: 1, 48 order:12345678	board:200x200 start: 4,195 order:12345678
board: 54x 54 start: 1, 53 order:12345678	board:202x202 start: 5,198 order:12345678
board: 56x 56 start: 1, 53 order:12345678	board:204x204 start: 1,198 order:12345678
board: 58x 58 start: 1, 54 order:12345678	board:206x206 start: 2,201 order:12345678
board: 60x 60 start: 1, 56 order:12345678	board:208x208 start: 1,203 order:12345678
board: 62x 62 start: 1, 57 order:12345678	board:210x210 start: 2,205 order:12345678
board: 64x 64 start: 1, 60 order:12345678	board:212x212 start: 3,207 order:12345678
board: 66x 66 start: 1, 63 order:12345678	board:214x214 start: 3,210 order:12345678
board: 68x 68 start: 1, 65 order:12345678	board:216x216 start: 2,210 order:12345678
board: 70x 70 start: 2, 70 order:12345678	board:218x218 start: 2,212 order:12345678
board: 72x 72 start: 3, 64 order:12345678	board:220x220 start: 2,220 order:12345678
board: 74x 74 start: 2, 66 order:12345678	board:222x222 start: 3,220 order:12345678
board: 76x 76 start: 2, 75 order:12345678	board:224x224 start: 2,218 order:12345678
board: 78x 78 start: 1, 74 order:12345678	board:226x226 start: 2,223 order:12345678
board: 80x 80 start: 1, 77 order:12345678	board:228x228 start: 1,225 order:12345678
board: 82x 82 start: 1, 81 order:12345678	board:230x230 start: 3,228 order:12345678
board: 84x 84 start: 1, 76 order:12345678	board:232x232 start: 1,229 order:12345678
board: 86x 86 start: 1, 80 order:12345678	board:234x234 start: 1,230 order:12345678
board: 88x 88 start: 1, 81 order:12345678	board:236x236 start: 2,231 order:12345678
board: 90x 90 start: 1, 87 order:12345678	board:238x238 start: 1,234 order:12345678
board: 92x 92 start: 5, 87 order:12345678	board:240x240 start: 1,236 order:12345678
board: 94x 94 start: 2, 90 order:12345678	board:242x242 start: 2,241 order:12345678
board: 96x 96 start: 1, 95 order:12345678	board:244x244 start: 2,240 order:12345678
board: 98x 98 start: 1, 95 order:12345678	board:246x246 start: 3,241 order:12345678
board:100x100 start: 1,100 order:12345678	board:248x248 start: 2,247 order:12345678
board:102x102 start: 2, 97 order:12345678	board:250x250 start: 3,243 order:12345678
board:104x104 start: 1,100 order:12345678	board:252x252 start: 2,246 order:12345678
board:106x106 start: 1,105 order:12345678	board:254x254 start: 3,254 order:12345678
board:108x108 start: 4,103 order:12345678	board:256x256 start: 2,255 order:12345678
board:110x110 start: 3,105 order:12345678	board:258x258 start: 2,254 order:12345678
board:112x112 start: 2,111 order:12345678	board:260x260 start: 6,254 order:12345678
board:114x114 start: 3,111 order:12345678	board:262x262 start: 2,255 order:12345678
board:116x116 start: 3,109 order:12345678	board:264x264 start: 3,259 order:12345678
board:118x118 start: 2,113 order:12345678	board:266x266 start: 2,266 order:12345678
board:120x120 start: 2,115 order:12345678	board:268x268 start: 1,265 order:12345678
board:122x122 start: 2,117 order:12345678	board:270x270 start: 3,263 order:12345678
board:124x124 start: 2,123 order:12345678	board:272x272 start: 3,266 order:12345678
board:126x126 start: 1,122 order:12345678	board:274x274 start: 2,267 order:12345678
board:128x128 start: 1,122 order:12345678	board:276x276 start: 4,269 order:12345678
board:130x130 start: 3,126 order:12345678	board:278x278 start: 2,274 order:12345678
board:132x132 start: 2,132 order:12345678	board:280x280 start: 2,274 order:12345678
board:134x134 start: 1,133 order:12345678	board:282x282 start: 2,282 order:12345678
board:136x136 start: 1,133 order:12345678	board:284x284 start: 1,280 order:12345678
board:138x138 start: 2,135 order:12345678	board:286x286 start: 1,280 order:12345678
board:140x140 start: 2,135 order:12345678	board:288x288 start: 2,287 order:12345678
board:142x142 start: 3,137 order:12345678	board:290x290 start: 4,286 order:12345678
board:144x144 start: 2,143 order:12345678	board:292x292 start: 3,286 order:12345678
board:146x146 start: 3,137 order:12345678	board:294x294 start: 4,291 order:12345678
board:148x148 start: 2,143 order:12345678	board:296x296 start: 4,296 order:12345678
board:150x150 start: 2,147 order:12345678	board:298x298 start: 1,290 order:12345678
board:152x152 start: 2,148 order:12345678	board:300x300 start: 1,298 order:12345678

Πίνακας 57 - Κυκλικές διαδρομές για σκακιέρες με 6≤N≤300 ζυγό.

Τα αποτελέσματα αυτά υπολογίζονται σε 20 περίπου λεπτά, με την εφαρμογή του αλγορίθμου σε ένα μέσο οικιακό υπολογιστή.

8.4. Συμμετρίες στις σειρές επίλυσης ισοπαλιών

Λόγο του τετράγωνου σχήματος της σκακιέρας είναι δόκιμο να μελετήσουμε την συμμετρία της σε σχέση με τις διαδρομές και τις σειρές επίλυσης ισοπαλιών που χρησιμοποιούμε. Ακολουθεί μια περιγραφή των συμμετριών αυτών και τι συνέπιες μπορεί να έχουν στον υπολογισμό των διαδρομών του ίππου.

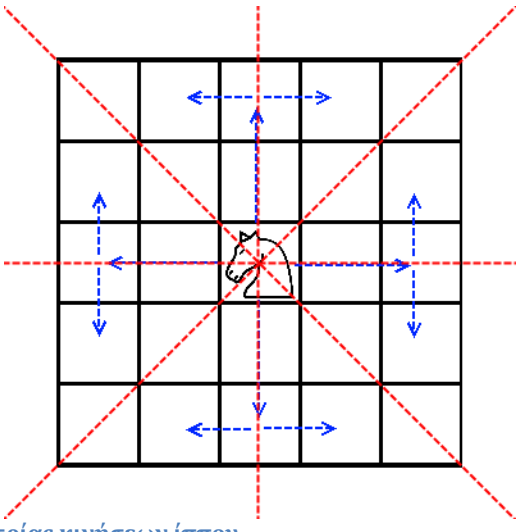
8.4.1. Εύρεση των συμμετριών

Από τις 40.320(=8!) σειρές επίλυσης κάποιες βρίσκουν ολοκληρωμένες διαδρομές για όλα τα τετράγωνα κάποιων μικρών σκακιέρων. Για παράδειγμα για την 8x8 σκακιέρα οι σειρές είναι οι ακόλουθες.

1 3 4 7 2 5 6 8
1 3 4 7 5 2 6 8
2 8 7 4 1 6 5 3
2 8 7 4 6 1 5 3
3 5 6 1 4 7 8 2
3 5 6 1 7 4 8 2
4 2 1 6 3 8 7 5
4 2 1 6 8 3 7 5
5 7 8 3 1 6 2 4
5 7 8 3 6 1 2 4
6 4 3 8 2 5 1 7
6 4 3 8 5 2 1 7
7 1 2 5 3 8 4 6
7 1 2 5 8 3 4 6
8 6 5 2 4 7 3 1
8 6 5 2 7 4 3 1

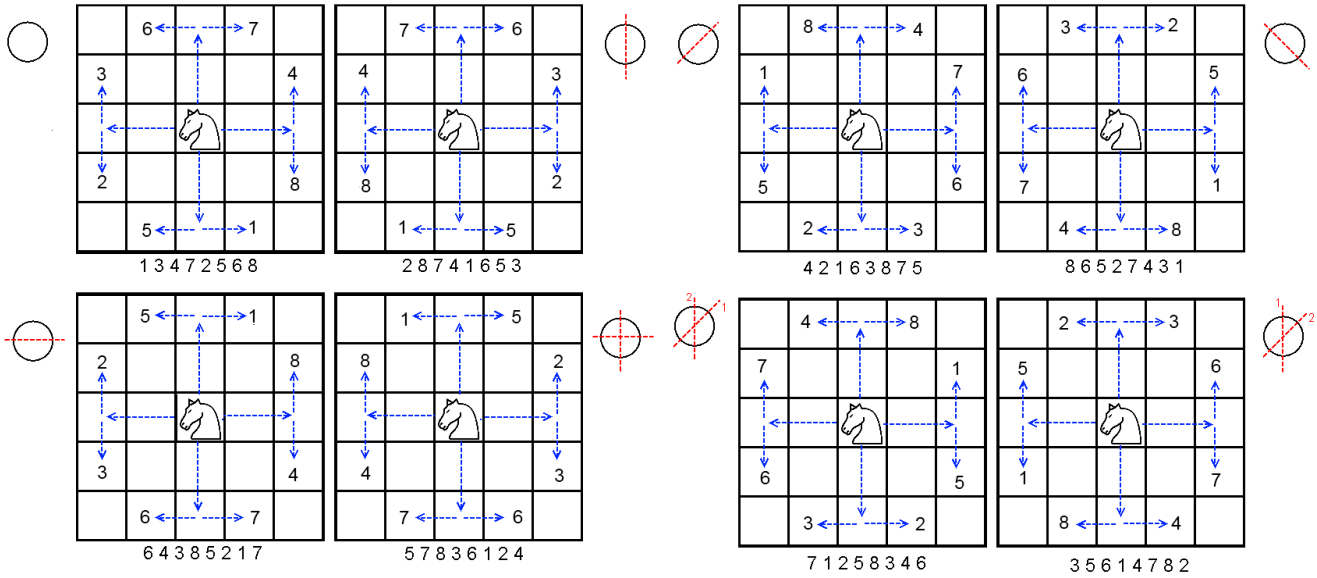
Πίνακας 58 – Σειρές επίλυσης ισοπαλιών που δίνουν ολοκληρωμένες διαδρομές από όλα τα τετράγωνα στην σκακιέρα 8x8

Στον παραπάνω πίνακα αρχίζουμε και παρατηρούμε κάποιες ομοιότητες μεταξύ των σειρών, ανά δύο βλέπουμε ότι είναι σχεδόν ίδιες. Για παράδειγμα η πρώτη '13472568' και η δεύτερη '13475268' είναι παρόμοιες, με διαφορά να έχουν μόνο στα πέμπτα και έκτα στοιχεία τους, τα οποία στην μία έχουν ανεστραμμένη σειρά από την άλλη. Αυτό επαναλαμβάνεται και στις άλλες σειρές. Ανά δύο διαφέρουν μόνο στις ίδιες δύο θέσεις οι οποίες περιέχουν τα ίδια νούμερα σε διαφορετική σειρά. Στην προσπάθεια να βρούμε κάποιον λόγο για αυτήν την ομοιότητα ανακαλύπτουμε ότι υπάρχει κάποια συμμετρία στις σειρές.



Εικόνα 21 – Άξονες συμμετρίας κινήσεων ίππου

Εφαρμόζουμε την πρώτη σειρά του πίνακα 58 στις κατευθύνσεις του ίππου και από την εικόνα που δημιουργείτε παίρνουμε τις συμμετρικές της. Βρίσκουμε ότι από μία εικόνα, περιστρέφοντας την γύρω από τους άξονες συμμετρίας της μία ή δύο φορές, μπορεί να γίνει η παραγωγή άλλων εφτά μοναδικών εικόνων. Κάθε εικόνα αντιπροσωπεύει μία σειρά επίλυσης ισοβαθμιών. Παρακάτω παραθέτουμε ένα τέτοιο παράδειγμα με τις εικόνες και σειρές που προκύπτουν.



Εικόνα 22 - Σειρά επίλυσης 13472568 και οι συμμετρικές της

Παρατηρούμε ότι οι σειρές αυτές, που προέκυψαν από τις συμμετρίες, είναι μέσα στις σειρές του πίνακα 58. Οι υπόλοιπες σειρές του πίνακα είναι και αυτές συμμετρικές μεταξύ τους, άρα μόνο οι δύο πρώτες σειρές είναι μοναδικές και οι υπόλοιπες βγαίνουν από τις συμμετρίες τους. Για αυτόν λόγο φαίνεται να μοιάζουν όλες οι σειρές μεταξύ τους. Οι δύο πρώτες έχουν μία μικρή διαφορά σε δύο στοιχεία τους και άρα οι συμμετρικές τους θα έχουν ανάλογες διαφορές.

8.4.2. Η ερμηνεία και οι χρησιμότητα των συμμετριών των σειρών πάνω στις σκακιέρες

Μια σειρά επίλυσης ισοπαλιών έχει κάποιο ποσοστό επιτυχίας σε κάθε σκακιέρα. Αυτό το ποσοστό είναι ίδιο για όλες τις συμμετρικές της σειρές. Οι παρακάτω εικόνες δείχνουν τα αποτελέσματα μίας σειράς καθώς και τα αποτελέσματα των συμμετρικών της.

Εικόνα 23 - Τα αποτελέσματα της σειράς 12345678 και των συμμετρικών της στην σκακιέρα 8x8

Παρατηρούμε ότι η επιτυχία των σειρών είναι η ίδια σε όλες και ότι η απεικόνιση των αποτελεσμάτων των διαδρομών ακολουθούν την συμμετρία που εφαρμόσαμε και στις σειρές. Αυτό επιτρέπει μεγάλη μείωση στους υπολογισμούς, αφού από 40320 (=8!) σειρές αρκεί να εξετάσουμε τις 5040 πρώτες και να εξάγουμε από αυτές τις υπόλοιπες μέσω των συμμετριών τους.

Ένας ακόμα τρόπος να επιβεβαιώσουμε την συμμετρία των σειρών είναι να υπολογίσουμε για κάθε τετράγωνο μιας σκακιέρας την διαδρομή που δίνει κάθε μία από τις 40320 σειρές επίλυσης. Τα αποτελέσματα, σε μία NxN με N ζυγό, θα πρέπει να είναι συμμετρικά οριζόντια, κάθετα και διαγώνια όπως δείξαμε και παραπάνω για τις σειρές. Ο πίνακας 59 δείχνει ακριβώς αυτό για την σκακιέρα 8x8.

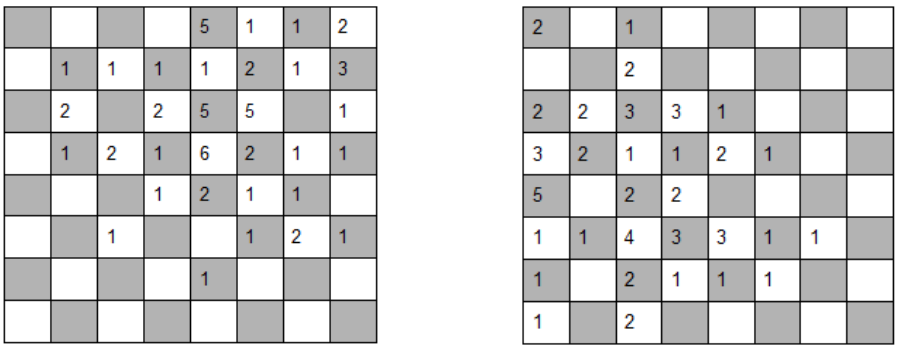
810	1445	2497	2183	2183	2497	1445	810
1445	8080	5942	5154	5154	5942	8080	1445
2497	5942	9622	3970	3970	9622	5942	2497
2183	5154	3970	7948	7948	3970	5154	2183
2183	5154	3970	7948	7948	3970	5154	2183
2497	5942	9622	3970	3970	9622	5942	2497
1445	8080	5942	5154	5154	5942	8080	1445
810	1445	2497	2183	2183	2497	1445	810

Πίνακας 59 - Αριθμός των σειρών επίλυσης ισοπαλιών από όλες (8!) που αποτυγχάνουν από κάθε τετράγωνο της 8x8 σκακιέρας.

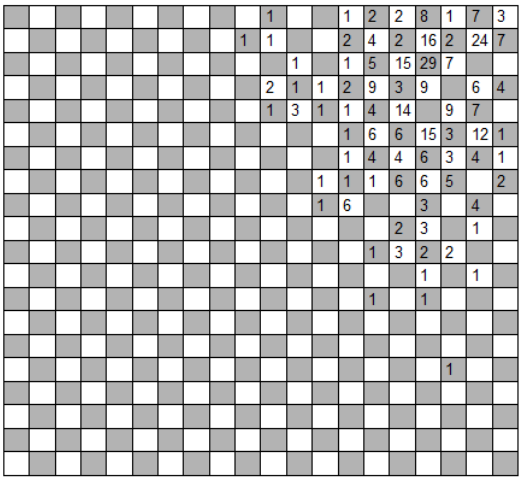
8.5. Κατάληξη της διαδρομής του ίππου ανάλογα με την επιλογή σειράς επίλυσης ισοβαθμιών

Ένα χαρακτηριστικό της μεθόδου επίλυσης ισοβαθμιών που χρησιμοποιούμε σε αυτή την υλοποίηση του κανόνα του Warnsdorff, είναι ότι κάποιες, από τις οκτώ πιθανές, κατευθύνσεις που μπορεί να κινηθεί ο ίππος έχουν πάντα προτεραιότητα στο να επιλεχθούν σε σχέση με κάποιες άλλες. Αυτό είναι ιδιαίτερα εμφανές όταν οι κατευθύνσεις με παρόμοια προτεραιότητα βρίσκονται κοντά η μια στην άλλη. Σε μια τέτοια περίπτωση ο ίππος καλύπτει πρώτα την περιοχή της σκακιέρας που αντιστοιχούν οι κατευθύνσεις με την μεγαλύτερη προτεραιότητα και το τελευταίο τετράγωνο της διαδρομής βρίσκεται, σχεδόν πάντα, στην περιοχή της σκακιέρας που αντιστοιχούν οι κατευθύνσεις με μικρή προτεραιότητα.

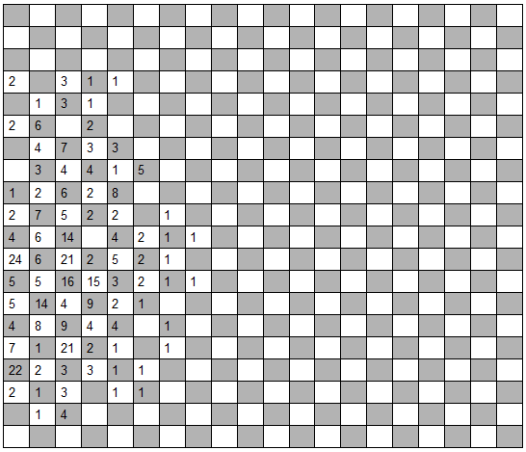
Οι παρακάτω εικόνες είναι κάποια δείγματα σκακιέρων με τα νούμερα να δείχνουν πόσες ολοκληρωμένες σειρές τελειώνουν σε κάθε τετράγωνο. Στις μικρές σκακιέρες, αν και υπάρχουν ενδείξεις, δεν είναι ξεκάθαρη η συγκέντρωση των τερματισμών, ενώ σε μεγαλύτερες τα αποτελέσματα είναι ποιο εμφανή.



Εικόνα 24 - Αριθμός ολοκληρωμένων διαδρομών που καταλήγουν σε κάθε τετράγωνο της σκακιέρας 8x8 με σειρά επίλυσης ισοπαλιών '12345678' (αριστερά) και '18765432' (δεξιά).

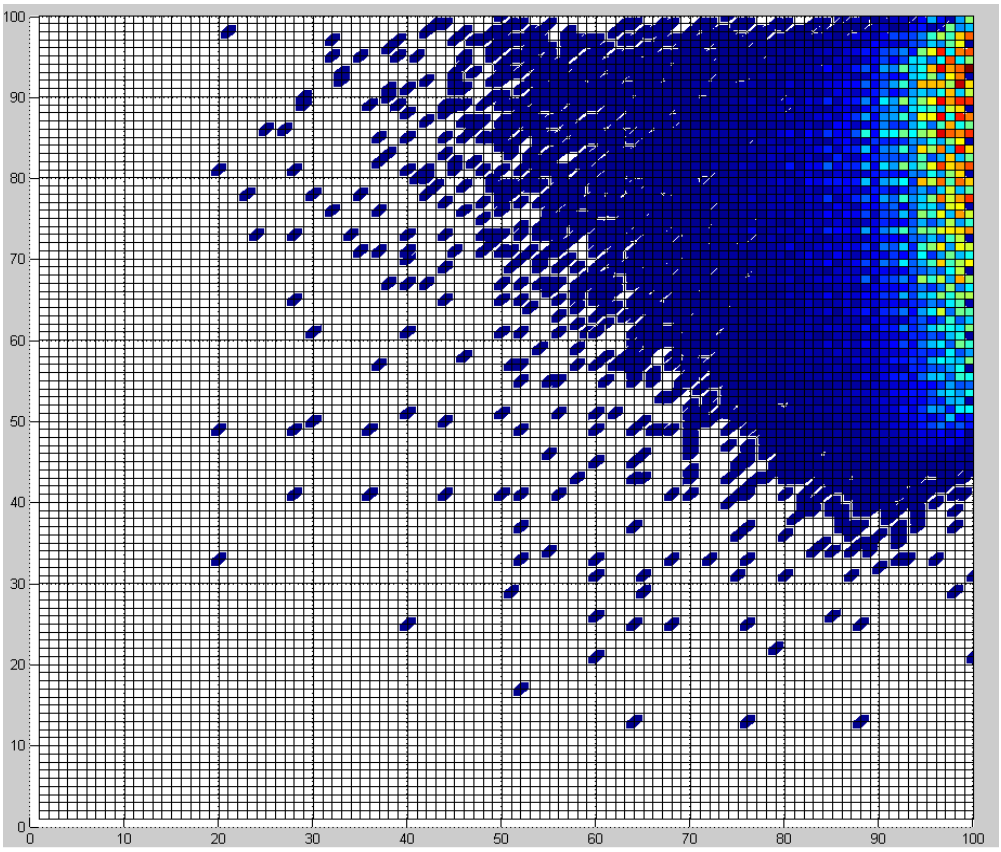


Εικόνα 25 - Αριθμός ολοκληρωμένων διαδρομών που καταλήγουν σε κάθε τετράγωνο της σκακιέρας 20x20 με σειρά επίλυσης ισοπαλιών '12345678'.

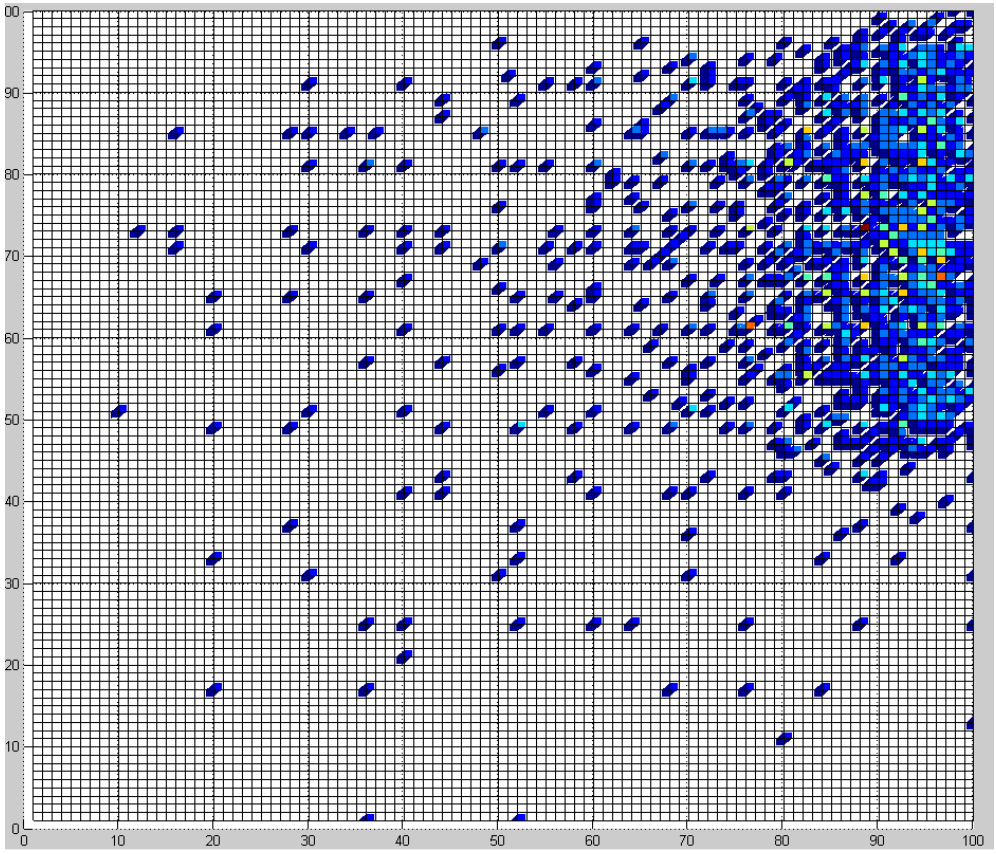


Εικόνα 26 - Αριθμός ολοκληρωμένων διαδρομών που καταλήγουν σε κάθε τετράγωνο της σκακιέρας 20x20 με σειρά επίλυσης ισοπαλιών '18765432'.

Το γράφημα της εικόνας 27 δείχνει ότι τα τελευταία τετράγωνα, των ολοκληρωμένων διαδρομών όλων των σκακιέρων από 6x6 μέχρι 100x100, συγκεντρώνονται πάνω-δεξιά. Στην σειρά επίλυσης ισοπαλιών που χρησιμοποιούμε σε αυτήν την περίπτωση, την '12345678', οι κατευθύνσεις με την μικρότερη προτεραιότητα βρίσκονται επίσης πάνω και δεξιά. Μια ακόμα παρατήρηση που αξίζει να κάνουμε είναι ότι οι αφετηρίες των κυκλικών διαδρομών συγκεντρώνονται και αυτές σε ένα μέρος της σκακιέρας ανάλογα με την σειρά επίλυσης ισοβαθμιών που χρησιμοποιούμε. Για να είναι κυκλική, ή κλειστή, μια διαδρομή πρέπει στο τέλος μιας ολοκληρωμένης διαδρομής του, ο ίππος να απειλεί το τετράγωνο της αφετηρίας. Το γράφημα της εικόνας 28 δείχνει τις αφετηρίες, όλων των σκακιέρων από 6x6 μέχρι 100x100, που οδηγούν σε ολοκληρωμένη κυκλική διαδρομή.



Εικόνα 27 – Τερματισμοί όλων των ολοκληρωμένων διαδρομών στις σκακιέρες NxN με 6≤n≤100.



Εικόνα 28 - Αφετηρίες των κυκλικών διαδρομών στις σκακιέρες NxN με 6≤n≤100.

9. Πέρα του NxN και του 6≤N≤300.

9.1. Μη τετράγωνες σκακιέρες

Χρησιμοποιήσαμε τετράγωνες σκακιέρες διότι αρκούσαν για να δείξουμε αυτά που θέλαμε για τον κανόνα του Warnsdorff και τον αλγόριθμο υλοποίησης του αλλά με λίγες επεμβάσεις στον κώδικα θα μπορούσαμε να βρούμε διαδρομές σε σκακιέρες KxN. Ουσιαστικά η μόνη πραγματική διαφορά στον κώδικα θα ήταν η δημιουργία των βαθμών της σκακιέρας στην συνάρτηση board_setup() στην αρχή αφού ο κώδικας ελέγχει κάθε θέση του ίππου ξεχωριστά και άρα δεν επηρεάζεται από το σχήμα της σκακιέρας.

Σε μια τέτοια περίπτωση όμως, όπου K≠N, θα πρέπει να λάβουμε υπόψη το θεώρημα του Schwenk το οποίο λέει ότι σε τυχαία KxN σκακιέρα (εκτός της εντελώς τετριμμένης 1x1 περίπτωσης) με K≤N η κλειστή διαδρομή του ίππου είναι πάντα εφικτή εκτός αν :

1. Οι K και N είναι και οι δύο περιττοί
2. K= 1, 2, ή 4
3. K = 3 and *N* = 4, 6, or 8.

Υπάρχει μόνο μερική απόδειξη του θεωρήματος αυτού (δεν έχει αποδειχτεί ότι σε όλες τις άλλες περιπτώσεις υπάρχουν πάντα κυκλικές διαδρομές του ίππου) αλλά τουλάχιστον σε αυτές τις περιπτώσεις η έρευνα για διαδρομή ίππου αποτυγχάνει.

9.2. Μεγαλύτερες διαστάσεις

Η υλοποίηση του κανόνα του Warnsdorff στην παρούσα εργασία βρίσκει διαδρομές για NxN σκακιέρες με 6≤N≤300 αλλά με κάποιες επιπρόσθετες συναρτήσεις και κάποιες μικρές αλλαγές θα μπορούσε να επεκταθεί σε οποιεσδήποτε διαστάσεις επιθυμούμε. Ακολουθεί μια περιγραφή ενός τρόπου που κάνει ακριβώς αυτό αλλά δεν θα εμβαθύνουμε πολύ στην εξήγηση του γιατί ξεφεύγει από τους στόχους της πτυχιακής αυτής.

Η ιδέα του τρόπου εύρεσης διαδρομών σε πολύ μεγάλες σκακιέρες που θα δούμε εδώ δεν είναι καινούργιος, ονομάζεται «διαίρει και βασίλευε» και έχει χρησιμοποιηθεί πολλές φορές στην προσπάθεια επίλυσης του προβλήματος του ίππου. Αυτό που θα κάνουμε θα είναι να πάρουμε τις διαστάσεις της σκακιέρας για την οποία επιθυμούμε να βρούμε διαδρομή και να την ‘σπάσουμε’ σε μικρότερες KxN σκακιέρες για τις οποίες μπορούμε να βρούμε διαδρομές χρησιμοποιώντας τον κανόνα του Warnsdorff.

Συγκεκριμένα τα σημεία που πρέπει να επισημάνουμε είναι τα εξής:

- Μεγέθη κομματιών:
  Οι σκακιέρες που θα παίξουν τον ρόλο των κομματιών πρέπει να είναι όσο ποιο μικρές και λίγες γίνετε για ταχύτερες λύσεις. Αποδεικνύεται ότι τα μεγέθη που αρκούν για την κάλυψη όλων των διαστάσεων πάνω από 300x300 είναι τα 5x6, 6x6, 8x6 και 10x6.

- Εύρεση κατάλληλων διαδρομών:
  Για τις διαδρομές των κομματιών μπορούμε να χρησιμοποιήσουμε τον κανόνα του Warnsdorff όπως υλοποιείτε στην παρούσα εργασία για να βρούμε κυκλικές

διαδρομές για τις ζυγές σκακιέρες και μία διαδρομή για την 5Χ6 που η αφετηρία της είναι στην γωνία της σκακιέρας και τερματίζει σε ένα από τα διπλανά τετράγωνα της αφετηρίας. Αυτό βοηθάει στην ένωση των κομματιών.

- Διαίρεση αρχικής σκακιέρας:
  Θα πρέπει η σκακιέρα να διαιρεθεί σωστά σε κομμάτια μεγέθους όπως αυτά που ορίσαμε πιο πάνω. Η διαίρεση γίνετε ως εξής: Έστω Κ η μία διάσταση μιας ΚxΝ σκακιέρας. Τότε οι διαστάσεις χωρίζονται Kmod(6) φορές και εάν το Κ/6 δεν είναι ακριβής διαίρεση τότε αφαιρείτε ένα Kmod(6) και προστίθεται μία φορά το KMod(6) συν το υπόλοιπο της διαίρεσης Κ/6. Επίσης εάν το Κ είναι μονό τότε πριν από όλες τις άλλες πράξεις αφαιρούμε 5 από το Κ. Κάνουμε το ίδιο και για την Ν διάσταση. Στο τέλος έχουμε πόσα και ποια κομμάτια χρειαζόμαστε.

- Συνένωση διαδρομών:
  Η ένωση των διαδρομών των κομματιών γίνετε στις γωνίες των σκακιέρων. Στις κυκλικές διαδρομές μπορούμε να ‘σπάσουμε’ την ένωση δύο θέσεων στις γωνίες και να ενώσουμε την κάθε θέση με μια της άλλης σκακιέρας και να φτιάξουμε έτσι μία κυκλική διαδρομή που εκτίνετε και στις δύο σκακιέρες. Για να ενώσουμε μία ζυγή με μία μονή σκακιέρα ενώνουμε τις άκρες της διαδρομής της μονής σκακιέρας με τις δύο θέσεις μία ένωσης που ‘σπάσαμε’ στην γωνία της ζυγής.

Με τον παραπάνω τρόπο μπορούμε να δημιουργήσουμε διαδρομές για όσο μεγάλες σκακιέρες θέλουμε σε αρκετά μικρό και γραμμικό χρόνο.

10.Δημοσίευση

10.1. Ελληνικά

Χρήση του ευριστικού αλγορίθμου που προτάθηκε από τον H. C. Warnsdorff στις αρχές του 19ου αιώνα για το πρόβλημα της εύρεσης της διαδρομής του ίππου και επέκταση αυτού για την αποδοτικότερη λύση σε περιπτώσεις «ισοπαλίας». Εφαρμογή της διαδικασίας σε σκακιέρες NxN με 5≤N≤300. Παρατηρήσεις για την αποτελεσματικότητα του αλγορίθμου και κάποιων ενδιαφερόντων χαρακτηριστικών των αποτελεσμάτων.

1.Εισαγωγή

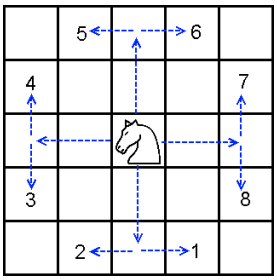
Το πρόβλημα του ίππου είναι το ερώτημα της εύρεσης μίας διαδρομής που μπορεί να ακολουθήσει ο ίππος περνώντας από όλα τα τετράγωνα της σκακιέρας χωρίς να πατήσει δεύτερη φορά σε ένα από αυτά. Ζητείτε δηλαδή μία Χαμιλτόνια διαδρομή με κόμβους τα τετράγωνα της σκακιέρας και ακμές τις κινήσεις του ίππου από κάθε τετράγωνο.

Οι τρόποι επίλυσης του προβλήματος με την βοήθεια υπολογιστών είναι πολλοί και έχουν χρησιμοποιηθεί όλοι αρκετές φορές σε εργασίες και μελέτες. Εδώ θα εφαρμόσουμε τον κανόνα του Warnsdorff, έναν ευριστικό αλγόριθμο που πρότεινε το 1823 ο H.Warnsdorff και είναι ο εξής: *Από την παρούσα θέση κινούμαστε στην επόμενη από την οποία ο ίππος απειλεί τις λιγότερες ελεύθερες θέσεις, αλλά όχι καμία. Σε περίπτωση ισοπαλίας η επιλογή μπορεί να είναι οποιαδήποτε από τις ισόπαλες.*

Η εφαρμογή του αλγορίθμου ακριβώς όπως τον αποτύπωσε ο Warnsdorff έχει μεγάλο ποσοστό επιτυχίας (>88%) σε πολύ μικρές σκακιέρες, μέχρι 24x24, αλλά όσο οι διαστάσεις αυξάνονται το ποσοστό πέφτει απότομα. Αυτό αποδίδεται στην τυχαία επιλογή της επόμενης κίνησης στην περίπτωση ισοπαλιών, διότι είναι εύκολο με μία λάθος επιλογή να δημιουργηθούν απομονωμένες περιοχές ανεπίσκεπτων τετραγώνων. Η αντικατάσταση της τυχαίας επιλογής με έναν τρόπο καθοδήγησης του ίππου σε μία συγκεκριμένη κατεύθυνση κατά την επίλυση των ισοπαλιών επιτρέπει στον αλγόριθμο να έχει ένα καλό ποσοστό επιτυχίας για όλα τα μεγέθη, τουλάχιστον μέχρι το N=300.

2.Σειρές προτίμησης για την επιλογή της επόμενης κίνησης του ίππου

Για να κινείτε ο ίππος σε μία συγκεκριμένη κατεύθυνση δίνεται μία σειρά προτίμησης για τις επιλογές που έχει. Ξεκινώντας με την κίνηση «δυο κάτω, ένα δεξιά» σημειώνεται ως κατεύθυνση ‘1’ και μετά με την φορά των δεικτών του ρολογιού ονομάζουμε τις υπόλοιπες κατευθύνσεις ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’ και ‘8’. Η εικόνα 1 δείχνει το αποτέλεσμα.



Εικόνα 1

Τα ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’ είναι οι θέσεις που μπορεί να προχωρήσει ο ίππος και η σειρά με την οποία είναι γραμμένες δείχνει την σειρά προτεραιότητάς τους.

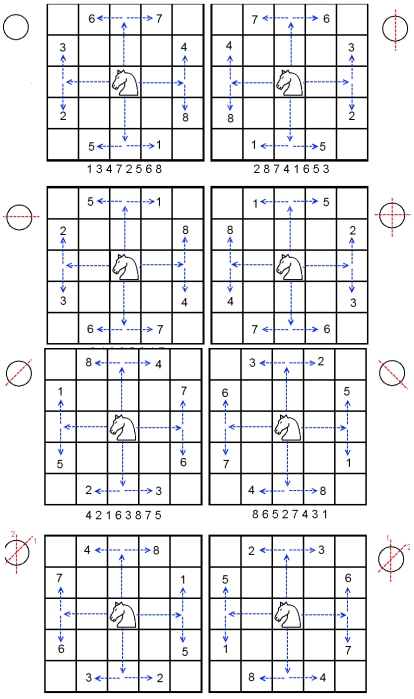
Αυτό που πετυχαίνει η μέθοδος είναι να ‘σπρώχνει’ τον ίππο πάντα σε μία συγκεκριμένη κατεύθυνση και έτσι να καλύπτονται οι θέσεις της σκακιέρας σε εκείνη την μεριά της χωρίς να αφήνονται κενά και μετά προχωρεί για να καλύψει την υπόλοιπη. Με την τυχαία επίλυση ισοπαλιών το ποσοστό αποτυχίας φτάνει το 95% στην

120x120 σκακιέρα, ενώ με την εφαρμογή της ίδιας τυχαίας σειράς σε όλες τις αφετηρίες η αποτυχία φτάνει το 45%.

3.Συμμετρία των σειρών επίλυσης ισοπαλιών

Ένα χαρακτηριστικό των κινήσεων του ίππου που βοηθάει στους υπολογισμούς των διαδρομών είναι η συμμετρία τους. Οι 8 κινήσεις του ίππου δημιουργούν ένα σχήμα το οποίο έχει μια οριζόντια, μια κάθετη και δυο διαγώνιους άξονες συμμετρίας. Με την περιστροφή του σχήματος των κινήσεων του ίππου γύρο από τους άξονες αυτούς δημιουργούνται άλλα 7 μοναδικά σχήματα.

Κατ' επέκταση αυτές οι συμμετρίες μπορούν να εφαρμοστούν και στις σειρές επίλυσης ισοπαλιών, άρα μπορούν να χωριστούν οι 40320(=8!) σειρές σε (40320/8=)5040 ομάδες των 8 σειρών οι οποίες βρίσκουν τις ίδιες διαδρομές σε μία σκακιέρα αλλά γυρισμένες γύρο από τους ίδιους άξονες όπως και οι σειρές επίλυσης. Για παράδειγμα η σειρά '12345678' έχει συμμετρικές τις '21876543', '34567812', '43218765', '56781234', '65432187', '78123456' και '87654321'.



Εικόνα 2

Για να βρούμε όλες τις διαδρομές που δίνει η μέθοδος των σειρών επίλυσης σε μία σκακιέρα αρκεί να βρούμε τις διαδρομές των 5040 «μοναδικών» σειρών - σειρών που δεν είναι καμία συμμετρική της άλλης - και να εξάγουμε από αυτές τις υπόλοιπες με χρήση της συμμετρίας τους. Παίρνουμε μία θέση ως σταθερή στις σειρές, για παράδειγμα '1xxxxxxx', και βρίσκουμε τους υπόλοιπους συνδυασμούς, οι οποίοι είναι οι 5040 μοναδικές σειρές. Η μέθοδος αυτή ελαττώνει τον χρόνο των υπολογισμών κατά 8 φορές.

4.Αποτελεσματικότητα της μεθόδου

Η χρήση συγκεκριμένης σειράς προτίμησης για την επιλογή της επόμενης κίνησης του ίππου σε περίπτωση ισοπαλίας δημιουργεί κάποια πρότυπα στην πορεία της διαδρομής τα οποία εμφανίζονται συχνά ιδικά στις μεγάλες σκακιέρες. Επίσης όσο οι σκακιέρες μεγαλώνουν σε διαστάσεις το ποσοστό αποτυχίας φαίνεται να συγκλίνει στο 70%-80% και να μένει σταθερά εκεί για 150≤N≤300. Αυτό χρειάζεται να επαληθευτεί και για μεγαλύτερες διαστάσεις αλλά αυτό το γεγονός σε συνδυασμό με την περιοδική εμφάνιση συγκεκριμένων προτύπων στις διαδρομές του ίππου ενισχύει την εντύπωση ότι με τις σειρές επίλυσης μπορούμε να βρούμε τουλάχιστον κάποιες διαδρομές σε όλα τα μεγέθη σκακιέρων.

Για να εξασφαλιστεί η εύρεση μιας ολοκληρωμένης διαδρομής σε οποιαδήποτε σκακιέρα από όλες τις θέσεις της, εφαρμόζεται μία τυχαία σειρά επίλυσης ισοπαλιών μέχρι να αποτύχει. Εάν αποτύχει παίρνουμε μία άλλη, μη συμμετρική με την πρώτη, σειρά και εφαρμόζουμε αυτήν από την αφετηρία που απέτυχε η προηγούμενη και επαναλαμβάνουμε - εάν αποτύχει και αυτή - μέχρι να βρεθεί ολοκληρωμένη διαδρομή. Πειραματικά αποτελέσματα δείχνουν ότι η αναζήτηση ολοκληρωμένων διαδρομών από όλες τις αφετηρίες μίας σκακιέρας με την χρήση τυχαίων σειρών σε περιπτώσεις αποτυχίας αυξάνει τον

χρόνο περάτωσης των υπολογισμών μόνο κατά 7%-8% σε σχέση με την περίπτωση που μία και μόνο σειρά θα εύρισκε διαδρομές από όλες τις αφετηρίες.

Άλλο ένα ενδιαφέρον αποτέλεσμα της μεθόδου είναι η σχετική ευκολία με την οποία βρίσκει κυκλικές διαδρομές στις σκακιέρες, αφού ο αλγόριθμος που γράφτηκε για την εφαρμογή της βρήκε κυκλώματα Hamilton σε όλες τις σκακιέρες $6 \leq N \leq 300$ για N ζυγό σε λίγα μόλις λεπτά τρέχοντας σε ένα μέσο υπολογιστή.

10.2. English

Use of the heuristic algorithm proposed by H. C. Warnsdorff in the early 19th century to solve the knights path problem and extend it for the most efficient solution in cases of "ties". Application of the procedure on NxN boards with 5 ≤ N ≤ 300. Comments on the effectiveness of the algorithm and on some interesting features of the results.

1. Introduction

The problem of the knights' path is the question of finding a path that the knight can follow and go through all the squares of the chessboard without passing a second time over one of them. We seek a Hamiltonian path in a grid with the nodes representing the board's squares and the edges representing the knights moves between the squares.

The ways to solve the problem with the help of computers are many and all have been used several times on papers and studies. Here we apply the rule of Warnsdorff, a heuristic algorithm proposed in 1823 by H.Warnsdorff: *From a given square move to the next square from which the knight is threatening the least free squares, but not none. In the case of a tie the choice can be arbitrary.*

In the implementation of the algorithm exactly as depicted Warnsdorff the success rate is high (> 88%) for very small boards up to 24x24, but as the size increases the rate drops sharply. This is attributed to the arbitrary selection of the next move in the event of a tie, because it is easy to choose a wrong option and create isolated areas of unvisited squares. Replacing the arbitrary selection with a way to guide the knight in a certain direction to solve the ties allows the algorithm to have a good to acceptable Symmetry. Turning the shape of knights movements around these axes create 7 other unique shapes.

success rate for all sizes, at least up to N = 300.

2. Order of priority for selecting the next knights move.

To move the knight in a certain direction, it is given an order of priority for the possible moves that are available to it. Starting with the move "Two down, one right" this direction is marked as '1' and then, going clockwise, the other directions are named '2', '3 ', '4', '5 ', '6', '7 'and '8'. Figure 1 shows the result.

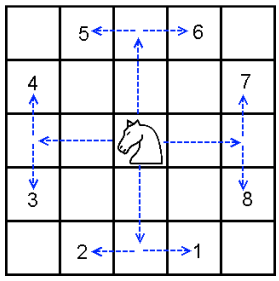


Figure 1

The '1 ', '2', '3 ', '4', '5 ', '6', '7 ', '8' are the moves available to the knight and the order in which they are written shows the order of priority.

What this method does is to 'push' the knight in a certain direction and thus covering the squares of the board on that side without leaving any empty squares and then proceeds to cover the rest. By solving ties randomly the failure rate reaches 95% at the 120x120 chessboard, while implementing the same random order of priority on all starting points the failure reaches 45%.

3. Symmetry of the order of priority.

A characteristic of the movements of the knight, that helps the calculations of the paths, is their symmetry. The 8 knights movements create a shape that has a horizontal, vertical and two diagonal axes of

By extension these symmetries can be applied to the orders of priority, so the 40320 (= 8!) series can be

divided in (40320/8 =) 5040 groups of 8 lines which find the same paths in a chessboard but turned around the same symmetry lines as the orders of priority. For example, the number '12345678' is the symmetric of '21876543', '34567812 ', '43218765', '56781234 ', '65432187', '78123456 'and '87654321'.

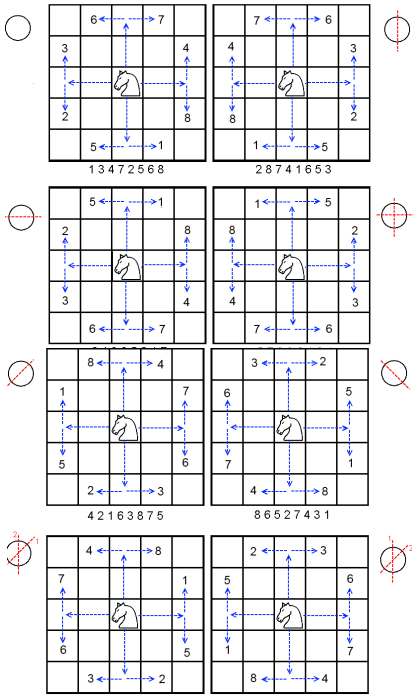


Figure 2

To find all the paths, given by the method of the order of priority, on a board, is enough to find the paths of the 5040 "unique" orders - orders that are not symmetrical to one another - and to draw from these the paths of the other orders of priority using their symmetries. We take a direction as a constant in the order, for example '1xxxxxxx ', and find the other combinations from that, which are 5040 unique orders. This method reduces the calculation time by 8.

4. Effectiveness of the method

The use of a particular series of priority for selecting the next movement of the knight in case of a tie creates some patterns in the paths, which often appear especially in larger boards. Also as the boards grow in size the failure rate seems to converge to 70% - 80% and stays there constantly for $150 \leq N \leq 300$. This needs to be verified for larger dimensions, but this event in conjunction with the periodic appearance of specific patterns in the knights paths reinforces the impression that at least some solutions can be found from some starting squares on all board sizes.

To ensure the discovery of a knights path on any board from all starting squares, the same random order of priority is applied constantly on all ties until a path fails. If it fails, another, random and non-symmetrical with the first, order is chosen and applied from the starting square that failed to produce a path and the procedure is repeated until a complete path is found. Experimental findings show that the use of a new, random, order of priority each time a path fails to complete, applied from the starting square increases the completion time of the calculations by only 7% - 8% compared to the time it would take for a single order of priority to find all paths from all starting squares.

Another interesting result of the method is the relative ease with which it finds circular routes on boards, since the algorithm found Hamilton circuits in all boards $6 \leq N \leq 300$ for N even in just a few minutes by running on an average computer.

11. Βιβλιογραφία

1. **Brendan McKay (1997). "Knight's Tours on an 8x8 Chessboard". *Technical Report TR-CS-97-03* (Department of Computer Science, Australian National University).**
2. **Conrad, A.; Hindrichs, T.; Morsy, H. & Wegener, I. (1994). "Solution of the Knight's Hamiltonian Path Problem on Chessboards". *Discrete Applied Mathematics* 50 (2): 125–134.**
3. **Ira Pohl (1967). "A Method for Finding Hamilton Paths and Knight's Tours". Communication of the ACM. Stanford University, Stanford, California**
4. **Martin Loebbing; Ingo Wegener (1996). "The Number of Knight's Tours Equals 33,439,123,484,294 — Counting with Binary Decision Diagrams". *The Electronic Journal of Combinatorics* 3 (1): R5.** (**Remark**: The authors later admitted that the announced number is incorrect. According to McKay's report, the correct number is 13,267,364,410,532 and this number is repeated in Wegener's 2000 book.)
5. **Parberry, Ian (1997). "An Efficient Algorithms for the Knight's Tour Problem". *Discrete Applied Mathematics* 73: 251–260.**
6. **Sam Ganzfried(2004). "A Simple Algorithm for Knight's Tours". Oregon State University**
7. **Σουρή Μυρτώ(2007). Το Πρόβλημα των Τεσσάρων Χρωμάτων, Μια Διήγηση με Έμφαση στο Γεωμετρικό Χαρακτήρα".Εθνικό και Καποδιστριακό Πανεπηστήμιο Αθηνών, Μεταπτυχιακό Τμήμα Διδακτικής και Μεθοδολογίας των Μαθηματικών.**
8. **Squirrel, Douglas; Cull, P. (1996). "A Warnsdorff-Rule Algorithm for Knight's Tours on Square Boards". Retrieved 2011-08-21.**
9. **Watkins, John J. (2004). *Across the Board: the Mathematics of Chessboard Problems*. Princeton University Press. ISBN 0-691-11503-6.**
10. **Weisstein, Eric W., "Knight's Tour" from MathWorld.**
11. **Wegener, I. (2000). *Branching Programs and Binary Decision Diagrams*. Society for Industrial & Applied Mathematics. ISBN 0-89871-458-3.**
12. **Y. Takefuji, K. C. Lee. "Neural network computing for knight's tour problems." *Neurocomputing*, 4(5):249–254, 1992**