Maze Generator with Iterative Backtracking & BFS Solving algorithms

Dimitrios Mpouziotas
Department of Informatics and Telecommunications
University of Ioannina
Arta, Greece
email: thl1816188@gmail.com

I. Abstract

Η εργασία αυτή υλοποιεί τους αλγορίθμους δημιουργίας λαβυρίνθου με τη χρήση γραφικών. Επίσης, η εργασία εμπεριέχει αλγόριθμους επίλυσης του λαβυρίνθου με σκοπό να βρεθεί ένα μονοπάτι από την αρχή έως ένα τέλος.
Υπάρχουν τρεις διαφορετική αλγόριθμοι δημιουργίας λαβυρίνθου, οι οποίοι είναι βασισμένοι στους αλγορίθμους BFS, DFS με τυχαία επιλογή γειτόνων και Disjoint-sets. Υπάρχουν επίσης δύο αλγόριθμοι επίλυσης του λαβυρίνθου ένας με τη χρήση του αλγορίθμου αναζήτηση κατά πλάτος και τον αλγόριθμο Iterative Backtracking. Ο χρήστης έχει την επιλογή να επιλέξει ποίος αλγόριθμος θα τρέξει με το να αλλάζει τους παραμέτρους.

II. Αλγόριθμος Δημιουργίας Λαβυρίνθου

Ο αλγόριθμος δημιουργίας λαβυρίνθου αρχικά δηλώνει τον λαβύρινθο με όλους τους τοίχους ανεβασμένους. Τα δεδομένα των τοίχων αποθηκεύονται σε μια λίστα και όπου η κάθε μια τιμή στη λίστα είναι ένας πόλος (Π.χ βόρεια, ανατολικά...). Χρησιμοποιώντας μια σταθερή τιμή που παριστάνει το πλάτος ενός κελιού, μπορούμε να τη χρησιμοποιήσουμε ώστε να ζωγραφίσουμε τον λαβύρινθο με βάση την τιμή αυτή. Με όλα αυτά τα δεδομένα το ζωγράφισμα του λαβυρίνθου θα είναι ένα grid. Ύστερα τα δεδομένα του λαβυρίνθου περνάνε στον αλγόριθμο Iterative Backtracking ο οποίος είναι βασισμένος από τον DFS αλγόριθμο Section IV. Το αποτέλεσμα θα είναι ένας λαβύρινθος

III. Αλγόριθμος Disjoint-Sets

Ο αλγόριθμος αυτός χρησιμοποιείται ειδικά για τη δημιουργία του λαβυρίνθου. Ο αλγόριθμος λειτουργεί σε ένα αντικείμενο **Disjointset** και μια λίστα **Edges**.
Το αντικείμενο Disjointset αποτελείται από τρείς βασικούς παραμέτρους, μια λίστα που αποθηκεύει το κάθε κελί το οποίο λειτουργεί ως parent, το μέγεθος τον κελιών που έχει ένα parent κελί και μια ακέραια τιμή που δείχνει τα συνολικά κελιά που έχουν συνενωθεί.
Η λίστα **Edges** παριστάνει την ακμή με ένα τυχαίο κελί και έναν τυχαίο γείτονα του. Οι πράξεις που γίνονται στο αντικείμενο **Disjointset** είναι η συνάρτηση **Find** που βρίσκει τη ρίζα ενός κελιού με αναδρομική επανάληψη. Επίσης, περιέχει και τη συνάρτηση **Union** στην οποία συνενώνει το σύνολο τις ρίζας ενός κελιού με τη ρίζα του γειτονικού του κελιού αν ολοκληρωθεί η συνενώσει τότε το **numSets** θα μειωθεί.
Ο αλγόριθμος ξεκινάει με επανάληψη όσο το $numSets > 1$, μέσα στην επανάληψη, επιλέγεται από τη λίστα **Edges**

IV. Αλγόριθμος Iterative Backtracking - DFS

Βασισμένος από τον DFS αλγόριθμο, ο αλγόριθμους Iterative backtracking λειτουργεί με τη χρήση των stacks δηλαδή τη μέθοδο LIFO (Last In First Out). Ο αλγόριθμος ψάχνει από ένα κελί, αν έχει έγκυρους γείτονες και επιλέγει έναν τυχαίο γείτονα, αυτός ο γείτονας μπαίνει στο stack και ονομάζεται ότι έχει επισκεφθεί. Ένας γείτονας μπορεί να χαρακτηριστεί ως έγκυρος αν ισχύει τα παρακάτω:
1) Αν ο τοίχος δεξιά / αριστερά / πάνω/ κάτω είναι πεσμένος
2) Αν έχει επισκεφθεί ήδη τον γείτονα δεξιά / αριστερά / πάνω /κάτω
3) Αν το κελί δεξιά, αριστερά, πάνω, κάτω, βρίσκεται στα εύρη του λαβύρινθου

Αν ισχύουν τα παραπάνω τότε ο γείτονας είναι έγκυρος και μπορεί να επισκεφθεί.
Αν δεν υπάρχει γείτονας να εξερευνήσει τότε ο αλγόριθμος ξεκινάει να κάνει backtrack αφαιρώντας από το stack το τελευταίο δεδομένο που έχει μπει μέσα. Τέλος, αν το stack αδειάσει τότε σημαίνει πως δεν έχει να εξερευνήσει άλλο γείτονα στον λαβύρινθο. Ο αλγόριθμος επίσης θα σταματήσει αν βρει τον τελικό προσδιορισμό (Maze Target). Ο αλγόριθμος Iterative Backtracking, χρησιμοποιείται και για τη δημιουργία του λαβυρίνθου και για την επίλυση του αλλά ο κανόνας 2) για την εγκυρότητα γείτονα δεν ισχύει.

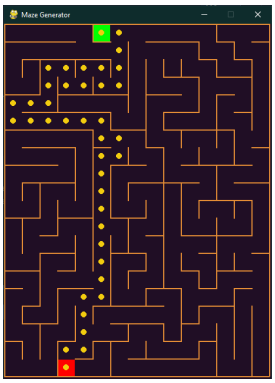

Figure 1. Iterative Backtracking Result

V. Αλγόριθμος Αναζήτησης κατά πλάτος - BFS

Ο αλγόριθμος αναζήτησης κατα πλάτος BFS λειτουργεί με τη χρήση των Queues δηλαδή τη μέθοδο FIFO (First in First Out). Στην εργασία υπάρχουν δύο διαφορετικές μορφές του αλγορίθμου BFS, ένας για τη δημιουργία του λαβυρίνθου που χρησιμοποιεί μια παρόμοια μορφή του αλγορίθμου Iterative Backtracking αλλά με τη χρήση των Queues και τέλος για την επίλυση του λαβυρίνθου. Στην εργασία ο BFS αλγόριθμος αρχικά δημιουργεί με αντικειμενοστραφή προγραμματισμό ένα γράφο με σκοπό να είναι συμβατός με τον λαβύρινθο και ύστερα εκτελεί τον αλγόριθμό BFS.
Ο αλγόριθμος BFS αναπτύχθηκε στο αρχείο Canvas.py και όχι στο Maze.py με σκοπό να εμφανίζονται οι αλλαγές όπως και οι γείτονες που έχουν διερευνηθεί από τον αλγόριθμο. Ο αλγόριθμος λειτουργεί ψάχνοντας όλους τους έγκυρους γείτονας από ένα κελί ξεκινώντας από την αρχή. Σε κάθε επανάληψη το αριστερότερο κελί βγαίνει από τη λίστα του Queue και επεξεργάζεται, ύστερα σε αυτό το κελί εξάγονται οι έγκυροι γείτονες. Κανόνες έγκυρου γείτονα:
1) Αν ο τοίχος πάνω, δεξιά, κάτω, αριστερά, είναι πεσμένος
2) Αν έχει επισκεφθεί ήδη τον γείτονα δεξιά / αριστερά / πάνω / κάτω

3) Αν βρίσκεται στα εύρη του λαβύρινθου

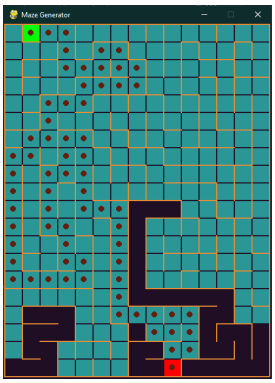

Figure 2. Breadth-First search result

Αν οι γείτονες είναι έγκυροι τότε τα δεδομένα των γειτόνων του γράφου θα αλλάξουν. Ο έγκυρος γείτονας είναι ο κόμβος ο οποίος έχει τοποθετηθεί στον γράφο και δεν έχει επισκεφθεί. Τα δεδομένα που αλλάζουν είναι
1) Αποθηκεύεται η επίσκεψη του γείτονα στον γράφο
2) Αποθηκεύεται ο πατέρας του γείτονα X, ο οποίος είναι το αριστερότερο κελί που αφαιρέθηκε από τη λίστα
3) Η τοποθεσία του γείτονα X αποθηκεύεται στον γράφο
4) Ο γείτονας X τοποθετείτε στη λίστα από αριστερά
5) Ο γείτονας X γίνεται το τρέχον κελί (Parent)

Αφού ο αλγόριθμος τελείωση έχουμε έναν γράφο έτοιμο ώστε να εξάγουμε το τελικό μονοπάτι, στο Figure 2 μπορούμε να δούμε πού έχει εξερευνήσει ο αλγόριθμος με τα μπλε τετράγωνα όπως και το τελικό μονοπάτι με τις μοβ κουκίδες. Για να εξάγουμε το τελικό μονοπάτι, κάνουμε επανάληψη μέχρι ο επόμενος κόμβος να είναι ίσως με την αρχή, ξεκινώντας από τον τέλος του γραφήματος. Όλοι οι κόμβοι ξεκινώντας από το τέλος εισάγονται σε μια λίστα στα αριστερά της λίστας. Τα στοιχεία που εισάγονται είναι ο πατέρας του επόμενου κελιού μέχρι ο πατέρας του επόμενου κελιού να είναι η αρχή του λαβύρινθου/γράφου.

VI. Πίνακας Λειτουργικών Συναρτήσεων

Συναρτήσεις αρχείου **Canvas.py**
Το αρχείο Canvas.py εμπεριέχει από ένα αντικείμενο Canvas το οποίο λειτουργεί ως το αντικείμενο επεξεργασίας γραφικών με τη χρήση της βιβλιοθήκης Pygame.
Το Canvas.py λειτουργεί ως το κύριο κομμάτι της εργασίας το οποίο περιέχει διάφορες παραμέτρους που μπορεί να αλλάξει ο χρήστης με βάση τον αλγόριθμο που επιθυμεί να τρέξει.

Table I: Canvas.py

Όνομα Συνάρτησης	Σύντομη Περιγραφή
runMaze()	Η κεντρική συνάρτηση στην οποία ελέγχει ποίος αλγόριθμος θα εκτελεστεί πρώτα, με ποιόν τρόπο και αν, με βάση παραμέτρων
drawMaze()	Το ζωγράφισμα γραφικών του λαβυρίνθου χρησιμοποιώντας τα δεδομένα ενός python Dictionary που επεξεργάζεται στο Maze.py αρχείο
drawBorder()	Το ζωγράφισμα γραφικών των συνόρων του λαβυρίνθου.
pathIndicator()	Το ζωγράφισμα γραφικών του δείκτη ο οποίος δημιουργεί το αίσθημα κινούμενων σχεδίων για τους αλγορίθμους generate maze και maze solvers.
drawFinalPathCell(Current)	Το ζωγράφισμα του τελικού μονοπατιού μιας της τοποθεσίας ενός Cell
getBFSPath()	Getter για την επιστροφή του τελικού μονοπατιού του BFS αλγορίθμου.
BreadthFirstSearch()	Ο αλγόριθμος επίλυσης Λαβυρίνθου, BFS (Breadth-First-Search). Επιλύει τον λαβύρινθο με τη χρήση του αλγορίθμου αναζήτηση πρώτα κατά πλάτος
BreadthFirstSearch_CreateGraph()	Δημιουργία του BFS γράφου με σκοπό τη συμβατότητα του αλγορίθμου BFS και τον λαβύρινθο
RemoveXWall()	Αφαίρεση τοίχου λαβυρίνθου ζωγραφίζοντας γραφικά πάνω από τη γραμμή του τοίχου που βρίσκεται στον X πόλο.

Continued on next page

Table I: Canvas.py (Continued)

Όνομα Συνάρτησης	Σύντομη Περιγραφή
runMaze()	Η κεντρική συνάρτηση στην οποία ελέγχει ποίος αλγόριθμος θα εκτελεστεί πρώτα, με ποιόν τρόπο και αν, με βάση παραμέτρων
drawMaze()	Το ζωγράφισμα γραφικών του λαβυρίνθου χρησιμοποιώντας τα δεδομένα ενός python Dictionary που επεξεργάζεται στο Maze.py αρχείο
drawBorder()	Το ζωγράφισμα γραφικών των συνόρων του λαβυρίνθου.
run()	Εκτελείται ως πρώτη συνάρτηση αφού ολοκληρωθεί το __init__ constructor του Canvas

Table II: Maze.py

Όνομα Συνάρτησης	Σύντομη Περιγραφή
createMaze()	Δημιουργεί τα δεδομένα του λαβυρίνθου χρησιμοποιώντας Dictionaries. Το κλειδί είναι η τοποθεσία του κελιού και η τιμή είναι οι τοίχοι βόρεια, ανατολικά, νότια, δυτικά.
makeEdges()	Συνάρτηση για τη δημιουργία της λίστας Edges για τον αλγόριθμο δημιουργίας λαβυρίνθου με τη χρήση Disjoint sets
generateMaze_Disjoint_Sets_Method()	Δημιουργίας του λαβυρίνθου phrumy;yuovkgaw τη δομή δεδομένων Disjoint-sets
generateMaze_LIFO_DFS_Method()	Δημιουργία του λαβυρίνθου χρησιμοποιώντας τον αλγόριθμο Iterative backtracking με τυχαίους γείτονες.
generateMaze_FIFO_BFS_Method()	Δημιουργία του λαβυρίνθου βασισμένο από τον αλγόριθμο BFS με τυχαίους γείτονες.
solveMazeRandomNeighbor()	Επίλυση του λαβυρίνθου με τη χρήση του αλγορίθμου Iterative backtracking.
getXWall(Location)	Επιστρέφει το τοίχο του X πόλου του επιθυμητού κελιού.
checkValidNeighbor(x, y)	Αναζήτηση έγκυρου γείτονα με βάση διάφορον κανόνων Κεφάλαιο IV
removeXWall()	Αφαιρεί τα δεδομένα του τοίχου X πόλου βόρεια, ανατολικά, νότια, δυτικά
checkNeighbors(x, y)	Βλέπει αν το κελί στην τοποθεσία x, y αν έχει γείτονες που έχει δεν επισκεφθεί και επιστρέφει τη λίστα τον γειτόνων.
Getters	Το αρχείο Maze.py περιέχει διάφορους Getters οι οποίοι είναι αυτό εξηγούνται με το όνομα τους.

VII. Οδηγίες εκτέλεσης κώδικα

Requirements: Python 3.x ≥, pip, pygame, pycharm

```
> pip install pygame
> cd My Maze Generator/
> python main.py
```

VIII. Οδηγίες που ακολούθησα στην εργασία

Maze Generation Source:
The Coding Train - Breadth-First Search
Wikipedia - Breadth-First Search

Breadth-First Search Source:
The Coding Train - Maze Generator with p5.js part 1 - 4
Wikipedia - Maze Generation Algorithm

Disjoint-sets Sources:
Using Disjoint Set (Union-Find) to Build a Maze Generator - Article
Αλγόριθμος εργασίας - PDF