**Τεχνητή Νοημοσύνη – 1η εργασία**

Ονοματεπώνυμο: Δεληγιαννάκης Χαράλαμπος

ΑΕΜ: 4383

**ΠΡΟΛΟΓΟΣ**

Σκοπός της συγκεκριμένης εργασίας είναι η λύση ενός προβλήματος σύμφωνα με το οποίο υπάρχει ένας αριθμός από αμάξια που προσπαθούν να βγουν από ένα πάρκινγκ (χωρισμένο σε κουτάκια σχηματίζοντας ένα δισδιάστατο grid) έχοντας να αντιμετωπίσουν εμπόδια (βράχους ή άλλα αμάξια). Κάθε αμάξι μπορεί να μετακινηθεί μόνο πάνω και κάτω ή μόνο δεξιά και αριστερά. Η λύση του προβλήματος χωρίστηκε σε 2 μέρη, τη μοντελοποίησή του σε C++, καθώς και την ανάπτυξη του αλγορίθμου Breadth First Search (BFS) για την επίλυσή του.

**ΠΑΡΑΔΟΧΕΣ**

-Κάθε αμάξι που βγαίνει εκτός πάρκιγνκ (Maze) εντοπίζεται και στη συνέχεια αποκτά συντεταγμένες (-1, -1).

-Τα αμάξια δημιουργούνται με κάποια σειρά, και με την ίδια σειρά θα μετακινηθούν. (π.χ Το 1ο αμάξι που δημιουργήθηκε είναι και το πρώτο που θα καταφέρει να βγει από το maze)

-Τα προβλήματα είναι σχεδιασμένα ώστε κάθε αμάξι που ξεκινάει να κινείται, να καταφέρνει να βγει και να συνεχίζει το επόμενο. (Οι άλυτες περιπτώσεις ανιχνεύονται.)

-Ένα σημείο με συντεταγμένες (x, y) αντιστοιχεί σε κελί Α[y][x].

**1ο ΥΠΟΕΡΩΤΗΜΑ – ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ**

-Ο κώδικας μπορεί να περιέχει διάφορες μεθόδους που δεν χρησιμοποιούνται, αλλά συμβάλλουν στην επεκτασιμότητά του. Υπάρχουν αρκετά επεξηγηματικά σχόλια για την κατανόηση της λειτουργίας του.

-Η κλάση Car είναι προαιρετική και αναπαριστά ένα αμάξι. Δηλαδή, έχει τις τρέχουσες συντεταγμένες του (x, y), καθώς και τη σταθερή κατέυθυνσή του (Α/Δ ή Β/Ν). Δημιουργήθηκε για λόγους ομοιομορφίας, ώστε αντί να υπάρχουν 3 x n διαφορετικές μεταβλητές (3 για κάθε αμάξι), να υπάρχουν n αντικείμενα της κλάσης Car.

-Η κλάση Maze μοντελοποιεί την κατάσταση του λαβύρινθου - πάρκινγκ. Περιέχει πληροφορίες για οτιδήποτε σχετίζεται με αυτόν, όπως μήκος, πλάτος και αρχικό αριθμό αμαξιών, ποια κελιά είναι κενά και ποια όχι, τα αμάξια που περιέχει, τον προηγούμενο λαβύριθνο-κατάσταση από τον οποίο προήλθε, την κίνηση του αμαξιού (Δ-Α-Π-Κ) που οδήγησε στην τρέχουσα κατάσταση και το βάθος του στο δέντρο αναζήτησης του αλγορίθμου BFS. Επίσης, μπορεί να επεκταθεί εκτελώντας κινήσεις με κάποιο αμάξι ανάλογα με το πως μπορεί να κινηθεί και οδηγώντας σε μία νέα κατάσταση, δηλαδή έναν νέο λαβύρινθο. Τέλος, υπερφορτώνει 2 τελεστές (= και ==) καθώς και την built-in συνάρτηση toString() για την ομαλή λειτουργία του προγράμματος.

-Η συνάρτηση main() δημιουργεί 3 διαφορετικά προβλήματα (δηλαδή 3 διαφορικές αρχικές καταστάσεις Maze) με τη βοήθεια των συναρτήσεων first(), second() και third() που υλοποιούνται στο ίδιο αρχείο (main.cpp). Σε αυτές τις τρεις συναρτήσεις, πρώτα δημιουργείται ο λαβύρινθος χωρίς τα αμάξια, έπειτα δημιουργούνται τα αμάξια με τις συντεταγμένες και τον τρόπο κίνησής τους, και έπειτα με τη βοήθεια της μεθόδου setCars() που υλοποιεί η κλάση Maze, προστίθονται σε αυτήν. Επίσης το αρχείο περιέχει και τη συνάρτηση graph() που παρουσιάζει οπτικά τον λαβύρινθο. Στη συνέχεια, η main() καλεί την BFS() για κάθε ένα από τα τρία προβλήματα και παρουσιάζει τη λύση τους και τα στατιστικά τους. Υπάρχουν τα κατάλληλα βοηθητικά μηνύματα. Η λειτουργία της BFS() εξηγείται παρακάτω.

**2ο ΥΠΟΕΡΩΤΗΜΑ – ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΟΥ BFS**

-Αρχικοποιούνται:

* Το μέτωπο αναζήτησης ως ουρά, επειδή πρόκειται για BFS, δηλαδή οι κόμβοι εξετάζονται κατά πλάτος.
* Το διάνυσμα του κλειστού συνόλου, που περιέχει τους κόμβους που έχουν επισκεπτεί.
* Μεταβλητή (s) που κάθε φορά θα περιέχει την τρέχουσα κατάσταση του Maze.
* Αριθμός κόμβων που έχουν εξεταστεί (examined - αρχικοποιείται στο 0).
* Αριθμός κόμβων του δέντρου αναζήτησης (totalNodes - αρχικοποιείται στο 1 – υπάρχει ήδη η αρχικά κατάσταση ως ρίζα).
* Το index του αμαξιού που κινείται (carIndex - αρχικοποιείται στο 0).

Στη συνέχεια, προστίθεται η αρχική κατάσταση στο μέτωπο αναζήτησης και ξεκινάει ο βασικός βρόγχος while.

-Ο βρόγχος τρέχει όσο το μέτωπο αναζήτησης δεν είναι άδειο και δεν έχουμε ξεπεράσει το όριο των αρχικών αμαξιών, δηλαδή υπάρχουν ακόμα αμάξια μέσα στο Maze. Επομένως, όταν θα τελείωσει ελέγχεται αν υπάρχουν ακόμα αμάξια στο Maze. Αν δεν υπάρχουν, τότε το πρόβλημα λύθηκε και επιστρέφεται η τελική κατάσταση, αλλιώς σημαίνει ότι δεν λύνεται και επιστρέφεται nullptr.

-Σε κάθε επανάληψη η μεταβλητή s, που αναπαριστά την τρέχουσα κατάσταση, παίρνει την τιμή του μπροστινού στοιχείου στο μέτωπο αναζήτησης, και στη συνέχεια αυτό αφαιρείται από το μέτωπο. Μετά, ελέγχεται αν αυτό το στοιχείο υπάρχει στο κλειστό σύνολο, δηλαδή αν έχει επισκεπτεί ήδη και αν ισχυέι τότε ξεκινάει πάλι ο βρόγχος.

-Αν το στοιχείο δεν έχει επισκεπτεί ήδη, τότε ξεκινάει η διαδικασία της εξέτασης του κόμβου. Αρχικά, παρουσιάζεται οπτικά με τη βοήθεια της graph() και βοηθητικών μηνυμάτων. Έπειτα, ο δείκτης examined αυξάνεται κατά ένα και το στοιχείο προστίθεται στο κλειστό σύνολο καθώς εξετάζεται. Στη συνέχεια το στοιχείο επεκτείνεται με τη βοήθεια της μεθόδου expand() της κλάσης Maze, όπου θα χρησιμοποιήσει τις μεθόδους goUp(), goDown(), goRight(), goLeft() για να εξετάσει τις πιθανές κινήσεις. Σε αυτές τις μεθόδους παιρνάμε το carIndex με το οποίο γίνονται οι κινήσεις. Τα παιδιά που προκύπτουν από την επέκταση του κόμβου s αποθηκέυονται σε ένα διάνυσμα children.

-Έπειτα, τρέχει ένας βρόγχος for για κάθε παιδί (children). Μέσα στον βρόγχο, αρχικά, αυξάνεται η μεταβλητή totalNodes κατά ένα, και εξετάζεται αν στο παιδί το αμάξι με index το carIndex έχει βγει από το πάρκινγκ. Συγκεκριμένα, ελέγχεται αν κάποια από τις συνετατεγμένες του είναι -1 ή έχει ξεπέρασει το όριο του Maze.

* Αν ισχύει, σημαίνει ότι το αμάξι βγήκε από το πάργκινγκ. Έτσι, αρχικά, θέτονται και οι δύο συντεταγμένες του με -1 για να τηρηθούν οι παραδοχές και αυξάνεται το carIndex κατά ένα, ώστε στην επόμενη επανάληψη να εξεταστεί το επόμενο αμάξι. Τέλος, ελέγχεται αν το carIndex έχει βγει εκτός ορίων. Αν ισχύει, σημαίνει ότι όλα τα αμάξια έχουν βγει εκτός πάρκιγνκ, επομένως έχουμε τελική κατάσταση, της οποίας την τιμή παίρνει η μεταβλητή s. Ο βρόγχος ξεκινάει με την εντολή continue και προφανώς θα σταματήσει καθώς έχουν βγει όλα τα αμάξια. Έτσι, επιστρέφεται η μεταβλητή s που αυτή τη στιγμή περιέχει την τελική κατάσταση.
* Αν δεν ισχύει, τότε κάθε παιδί προστίθεται στο μέτωπο αναζήτησης (ουρά) και ξαναξεκινάει ο βρόγχος.

**ΠΑΡΑΔΕΙΓΜΑΤΑ – ΣΤΑΤΙΣΤΙΚΑ**

-Δημιουργήθηκαν 3 προβλήματα και εκτελέσθηκε BFS για το καθένα. Το κάθε πρόβλημα έχει μεγαλύτερη πολυπλοκότητα από το προηγούμενο (μεγαλύτερες διαστάζεις λαβυρίνθου, περισσότεροι βράχοι, περισσότερα αμάξια).

1ο ΠΡΟΒΛΗΜΑ - ΛΥΣΗ

A diagram of a mathematical equation

Description automatically generated with medium confidence

Depth of BFS tree: 4, Examined Nodes: 3, Total Nodes Discovered: 6

2ο ΠΡΟΒΛΗΜΑ - ΛΥΣΗ

A diagram of arrows and letters

Description automatically generated

Depth of BFS tree: 9, Examined Nodes: 8, Total Nodes Discovered: 14

3ο ΠΡΟΒΛΗΜΑ - ΛΥΣΗ

A diagram of a circuit

Description automatically generated

Depth of BFS tree: 21, Examined Nodes: 20, Total Nodes Discovered: 35

-Ο αριθμός των κόμβων του δέντρου κάθε προβλήματος είναι σχετικά πολύ μικρός καθώς κάθε αμάξι βγήκε με την πρώτη, χωρίς να δοκιμάσει επιπλέον επιλογές-κινήσεις, αφού είτε άνηκαν στο κλειστό σύνολο, είτε ήταν αδύνατες λόγω βράχου ή κατεύθυνσης του αμαξιού. Γενικά, επειδή κάθε αμάξι έχει μόνο μία κατεύθυνση, άρα δύο δυνατές κινήσεις, αν διαλέγοντας μόνο τη μία καταφέρει να βγει από το πάρκινγκ, δεν θα έχει δημιουργήσει πολλούς κόμβους καθώς κάθε προηγούμενη κατάσταση θα βρίσκεται στο κλειστό σύνολο, δηλαδή θα έχει εκτελεστεί ήδη.