ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνητή Νοημοσύνη
Εαρινό Εξάμηνο 2021

ΕΥΤΥΧΙΑ ΜΠΟΥΡΛΗ , 4441

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

1. Τρόπος Εκτέλεσης των Προγραμμάτων UCS και A*

```
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ gcc -o A Astar.c
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ ./A 5
The number of elements is: 5
Give number:  3
Give number:  5
Give number:  4
Give number:  1
Give number:  2
Found it!
g(n): 4
Path: 35412 -> 45312 -> 54312 -> 21345 -> 12345
Extensions: 84
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ gcc -o u ucs.c
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ ./u 5
The number of elements is: 5
Give number:  3
Give number:  5
Give number:  4
Give number:  1
Give number:  2
Found it!
Cost: 4
Path: 35412 -> 45312 -> 54312 -> 21345 -> 12345
Extensions: 644
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$
```

Εικόνα 1

Το Ν δίνεται από την είσοδο π.χ. ./a.out <πλήθος στοιχείων>.

Για το παιχνίδι:

```
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ gcc -o g Game.c
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ ./g 7 7

0 0 0 A 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 B 0 0 0
```

Εικόνα 2

2. Επεξήγηση της Ευρετικής Συνάρτησης του A*

```
87  //Find h(n) Heuristic function
88  int hfunc(int N, struct node* child){
89      int i,h=1;
90
91      for(i=0; i<N-1; i++){
92          if(abs(child->numbers[i]-child->numbers[i+1])!= 1){
93              h++;
94          }
95      }
96      return h;
97  }
```

Εικόνα 3

Στην παραπάνω εικόνα(εικ.1) φαίνεται η ευρετική συνάρτηση του αλγορίθμου A*(A-Star). Η συνάρτηση hfunc παίρνει δύο ορίσματα , την μεταβλητή N η οποία έχει αποθηκευμένο το πλήθος των στοιχείων και την μεταβλητή child η οποία είναι ένα δεδομένο τύπου struct node , πρόκειται δηλαδή για τον κόμβο που θα του υπολογίσουμε το h(child).

Έπειτα διατρέχει την ακολουθία των αριθμών που περιέχεται στον κόμβο child από την θέση 0 μέχρι την θέση N-1 και ελέγχει την τιμή της απόλυτης διαφοράς του κάθε στοιχείου με το ακριβώς επόμενο του. Αν η απόλυτη τιμή της διαφοράς είναι διάφορη του 1 αυτό σημαίνει ότι τα δύο στοιχεία που ελέγχθηκαν δεν είναι γειτονικά και αυξάνουμε την τιμή του h κατά μία μονάδα. π.χ.

```
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ gcc -o A Astar.c
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$ ./A 5
The number of elements is: 5
Give number:  3
Give number:  5
Give number:  2
Give number:  4
Give number:  1
Found it!
g(n): 5
Path: 35241 -> 25341 -> 43521 -> 34521 -> 54321 -> 12345
Extensions: 44
eftychia@DESKTOP-0GDNS5T:/mnt/c/Users/bourl/OneDrive/Desktop$
```

Εικόνα 4

Path: 35241 -> 25341 -> 43521 -> 34521 -> 54321 -> 12345

| | | | |

3 4 2 4 1 2 5 34 1 43 5 21 345 21 54321

h=5 h=4 h=3 h=2 h=1

Μέσω της ευρετικής μας συνάρτησης μπορούμε να δούμε σε τι βαθμό είναι ταξινομημένη η ακολουθία με τους αριθμούς μας και να προβλέψουμε περίπου πόσο ακόμα απέχουμε από την Τελική Κατάσταση, διαλέγοντας έτσι τους κόμβους με το μικρότερο κάθε φορά e(n) και αποφεύγοντας τους κόμβους που είτε μας απομακρύνουν από την λύση, είτε μπορούμε να φτάσουμε σε αυτούς με μικρότερο κόστος διαδρομής.

Επίσης μέσω εκτεταμένων παραδειγμάτων και διαφορετικών περιπτώσεων επαληθεύσαμε ότι για όποιο n και αν διαλέγαμε η τιμή του h(n) ήταν μικρότερη ή το πολύ ίση με την πραγματική απόσταση a(n) και έτσι βεβαιωθήκαμε ότι η ευρετική μας συνάρτηση είναι αποδεκτή.

Άρα και οι δύο αλγόριθμοι (UCS & A*) είναι βέλτιστοι και πλήρεις, ωστόσο παρατηρήσαμε ότι ο A* θα χρειαστεί πολύ λιγότερες επεκτάσεις σε σχέση με τον UCS.

3. Άσκηση

Στην 3η άσκηση έχουμε υλοποιήσει πλήρως τις λειτουργίες για τον χρήστη (MIN ή playerB) και σκεφτήκαμε την υλοποίηση για τον υπολογιστή (MAX ή playerA) ωστόσο δεν προλάβαμε να την υλοποιήσουμε πλήρως.

Η υλοποίηση του Max θα πήγαινε ως εξής:

- Παίρνουμε τα δεδομένα της τρέχουσας τοποθεσίας του Max και δημιουργούμε μία ρίζα.
- Ελέγχουμε τις θέσεις γύρω από την ρίζα αν είναι άδειες (δεν έχουν X ή B) και για κάθε ελεύθερη θέση δημιουργούμε και ένα παιδί.
- Τα παιδιά της ρίζας καλούν στην συνέχεια μία άλλη συνάρτηση στην οποία το καθένα δημιουργεί τα δικά του παιδιά με παρόμοιο τρόπο με την ρίζα και η διαδικασία αυτή επαναλαμβάνεται μέχρι να δημιουργηθεί το δέντρο.
- Κρατάμε όλους τους κόμβους σε έναν πίνακα. Κάθε κόμβος περιέχει πεδίο κόστος (απόσταση από την τρέχουσα ρίζα),καθώς επίσης και έναν πίνακα με τα παιδιά του.
- Στη συνέχεια όταν σχηματιστεί το δέντρο (ιδανικά ολόκληρο, δηλαδή μέχρι όλα τα φύλλα να φτάσουν σε τελική κατάσταση) με φύλλα κόστους 4 καλούμε μία συνάρτηση δίνοντάς της αρχικά την τρέχουσα ρίζα η οποία για κάθε παιδί της θα καλέσει τον εαυτό της και ίδια διαδικασία θα επαναλαμβάνεται από τα παιδία μέχρι να φτάσουμε στα φύλλα (δηλαδή τους κόμβους χωρίς παιδιά) και

στην συνέχεια από το τέλος προς την αρχή (αναδρομικά) κάθε πατέρας ανάλογα με το αν έχει ζυγό ή μονό αριθμό κόστους θα παίρνει την καλύτερη ή χειρότερη τιμή score των παιδιών του και θα επιστρέφει.

- Η τιμή score των φύλλων, δηλαδή των τελευταίων κόμβων χωρίς παιδιά, θα υπολογίζεται με βάση το πόσες ελεύθερες θέσεις υπάρχουν τριγύρω τους.