ΑΤΟΜΙΚΉ ΕΡΓΑΣΊΑ
ΣΤΗΝ
ΤΕΧΝΗΤΉ ΝΟΗΜΟΣΎΝΗ

Όνομα: Άννα
Επώνυμο: Γκίκα Ζοσάν
ΑΕΜ: 3349

Αριστοτέλειο Πανεπιστήμιο
Θεσσαλονίκης

Τμήμα
Πληροφορικής

ΠΕΡΙΕΧΌΜΕΝΑ

1.Γενική Περιγραφή

Το πρόγραμμα δέχεται ένα ταμπλό(πίνακας μορφής 2x2) και ο σκοπός του είναι να αλλάξει τις θέσει των κελιών με σκοπό να ταξινομηθούν.

1	3	6
4		2
7	5	8

Αρχική Κατάσταση

1	2	3
4	5	6
7	8	

Τελική Κατάσταση

Η αλλαγή αυτή γίνεται με τρεις διαφορετικούς τρόπους-αλγορίθμους. Ο πρώτος αλγόριθμος που χρησιμοποιείται είναι ο DFS(Depth First Search-Αναζήτηση κατά βάθος),ο δεύτερος είναι ο BFS (Breadth First Search - Αναζήτηση κατά πλάτος) και ο τελευταίος είναι ο BestFS (Best First Search).

Παρακάτω αναλύονται,όχι μόνο ο κάθε αλγόριθμος ξεχωριστά,αλλά και το τι προβλήματα προκύπτουν για διαφορετικές εισόδους και οι καταστάσεις που εξετάζονται για μία εξαιρετικά απλή περίπτωση(μία αλλαγή) ,μία μέση περίπτωση(5 αλλαγές) και για μία ακραία περίπτωση(όλα τα κελιά χρειάζονται αλλαγή).

Υποσημείωση1: Όπου αναφέρεται ότι τερματίζει το πρόγραμμα,εννοείται ότι σταματάει λόγω μνήμης.

Υποσημείωση2:Καλύτερη Περίπτωση/Χειρότερη Περίπτωση

1	2	3
4	5	6
7		8

5	6	8
7	4	2
1	3	

2.Αλγόριθμοι

Προτού αναλυθούν οι αλγόριθμοι που χρησιμοποιήθηκαν για την εύρεση λύσης θα γίνει μία ανάλυση της κλάσης Game που χρησιμοποιήθηκε για την εύρεση παιδιών,του βάθους και όλων των κοινών ή/και απαραίτητων μεθόδων που χρησιμοποιήθηκαν.

vector<Game* expand() : Βρίσκει τα παιδιά της κάθε κατάστασης.
bool goUp(/Down/Left/Right)(Game &n): Πάει στην πάνω/κάτω/αριστερή/δεξιά κατάσταση και ελέγχει αν είναι παιδί.
int getDepth(): Βρίσκει το βάθος του μονοπατιού-αν υπάρχει.
void setPrevious(Game *p): Σε κάθε παιδί, ορίζει ως προηγούμενο το γονιό του.
unsigned long getKey(): Το κλειδί του κάθε παιδιού.*
string toString(): Παίρνει τον δοθέντα αριθμό και τον μετατρέπει σε τύπο δεδομένων string/

*Ο τρόπος εύρεσης του κλειστού είναι ο εξής. Μπαίνουμε σε δύο for loop και ελέγχουμε αν το στοιχείο του πίνακα είναι αυτό που πρέπει να είναι (π.χ. a[1][2]-δηλαδή στην 6η θέση==6) . Αν είναι τότε σε ένα δεκαψήφιο αριθμό(1000000000 αν κανένας αριθμός δεν είναι στη θέση του) βάζουμε στην αντίστοιχη θέση τον αριθμό που είναι στη θέση του(θεωρούμε το κενό ως το 9 αλλά δεν το εμφανίζουμε).

πχ.

1	4	6
7	3	2
5	8	

Κλειδί: 1 1 0 0 0 0 0 0 8 0

2α. DFS

Ο πρώτος αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του προβλήματος-ολοκλήρωσης του παιχνιδιού είναι ο Depth First Search. Τα βήματα του συγκεκριμένου αλγορίθμου είναι τα εξής.

Βήματα
1. Βάζουμε την αρχική κατάσταση στο μέτωπο αναζήτησης,
2. Αν το μέτωπο αναζήτησης είναι κενό σταματάμε.
3. Βγάζουμε την κατάσταση από το μέτωπο αναζήτησης(έχουμε τα δεδομένα της κρατημένα).
4. Αν η κατάσταση ανήκει στο κλειστό σύνολο, επιστρέφουμε στο βήμα 2.
5. Αν η λύση είναι τελική σταματάμε και την επιστρέφουμε,αν θέλουμε περισσότερες λύσεις συνεχίζουμε.
6. Προσθέτουμε την κατάσταση στο κλειστό σύνολο.
7. Βρίσκουμε τα παιδιά της κατάστασης και τα βάζουμε στην αρχή του μετώπου αναζήτησης.
8. Επιστρέφουμε στο βήμα 2 και επαναλαμβάνουμε.

Προβλήματα που εντοπίστηκαν
Αν τα βήματα του αλγορίθμου επαναληφθούν πολλές φορές,τότε ο αλγόριθμος όχι μόνο αργεί να τρέξει,αλλά πολλές φορές τερματίζει λόγω έλλειψη μνήμης.
Το συγκεκριμένο πρόβλημα μπορεί να λυθεί εν μέρη αν δεν εμφανίσουμε το μονοπάτι.
Ωστόσο,παρατηρήθηκε ότι κρασάρει πάρα πολύ εύκολα, με αποτέλεσμα να διακόπτεται όλο το πρόγραμμα. Για αυτό λοιπόν τον λόγο έχει μπει το κομμάτι που καλεί τον DFS σε comment.

Καλύτερη Περίπτωση: -Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-UpRight-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-Left-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-Left-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-Left-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-Left-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-DownDown-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-Left-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Down-Left-Up-Up-Right-Down-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-Up-Left-Down-Down-Right-Up-Left-Down-Right-Up-UpLeft-Down-Down-Right-Up-Left-Down-Right-Up-Left-Down-Right-Up-UpLeftDownDownRightUpLeftDownRightUpUpLeftDownDownRightUp-Left-Down-Right-Up-Left-Down-Right-Up-Up-Right-Down-Down-/1063
Mem: 1912, Examined: 1071

Μέση Περίπτωση: Τερματίζει.

Χειρότερη Περίπτωση: Τερματίζει

2.β BFS

Ο δεύτερος αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του προβλήματος-ολοκλήρωσης του παιχνιδιού είναι ο Breadth First Search. Τα βήματα του συγκεκριμένου αλγορίθμου είναι τα εξής.

Βήματα

1. Βάζουμε την αρχική κατάσταση στο μέτωπο αναζήτησης,
2. Αν το μέτωπο αναζήτησης είναι κενό σταματάμε.
3. Βγάζουμε την κατάσταση από το μέτωπο αναζήτησης(έχουμε τα δεδομένα της κρατημένα).
4. Αν η κατάσταση ανήκει στο κλειστό σύνολο, επιστρέφουμε στο βήμα 2.
5. Αν η λύση είναι τελική σταματάμε και την επιστρέφουμε,αν θέλουμε περισσότερες λύσεις συνεχίζουμε.
6. Προσθέτουμε την κατάσταση στο κλειστό σύνολο.
7. Βρίσκουμε τα παιδιά της κατάστασης και τα βάζουμε στο τέλος του μετώπου αναζήτησης.
8. Επιστρέφουμε στο βήμα 2 και επαναλαμβάνουμε.

Προβλήματα που εντοπίστηκαν

Σε αντίθεση με τον DFS, ο συγκεκριμένος αλγόριθμος δεν αντιμετωπίζει κάποιο συγκεκριμένο πρόβλημα στην εμφάνιση του μονοπατιού.

Καλύτερη Περίπτωση:-Right-/1, Mem: 4, Examined: 2

Μέση Περίπτωση(ζητούμενη):Path: -Right-Up-Left-Down-Down-Right- Mem: 140, Examined: 76

Χειρότερη Περίπτωση: Δεν βρίσκει λύση

2.γ Best First

Ο επόμενος αλγόριθμος που χρησιμοποιήθηκε είναι ο BestFS.

Βήματα
1. Βάζουμε την αρχική κατάσταση στο μέτωπο αναζήτησης.
2.Αν είναι κενό σταματάμε.
3.Παίρνουμε την πρώτη κατάσταση από το μέτωπο αναζήτησης.
4.Αν είναι μέλος του κλειστού συνόλου πάμε στο βήμα 2.
5.Αν η κατάσταση είναι τελική τότε την επιστρέφουμε και αν θέλουμε περισσότερες συνεχίζουμε(από την αρχή).
6.Βρίσκουμε τα παιδιά και εφαρμόζουμε την ευριστική συνάρτηση σε κάθε παιδί.
7.Βάζουμε τα παιδιά στο μέτωπο αναζήτησης.
8.Αναδιατάσσουμε το μέτωπο αναζήτησης ώστε η κατάσταση με την καλύτερη ευριστική τιμή να είναι πρώτη.
9.Βάζουμε την κατάσταση στο κλειστό σύνολο.
10.Επιστρέφουμε στο βήμα 2.

Προβλήματα που εντοπίστηκαν
Όμοια με τον BFS δεν εντοπίστηκε κάποιο ουσιαστικό πρόβλημα.

Καλύτερη Περίπτωση: -Right-/1, Mem: 4, Examined: 2

Μέση Περίπτωση(ζητούμενη): Path: -Right-Up-Left-Down-Down-Right- Mem: 70, Examined: 43

Χειρότερη Περίπτωση: Τερματίζει.

3.Bonus Part

Ο κώδικας που χρησιμοποιείται για τον αλγόριθμο A* είναι σχεδόν ίδιος με τον κώδικα του BestFS με την εξής διαφορά: Στην ευριστική συνάρτηση προσθέτουμε την απόσταση από την αρχική κατάσταση.

Καλύτερη Κατάσταση: -Right-/1, Mem: 4, Examined: 2

Μέση Κατάσταση(ζητούμενη): -Right-Up-Left-Down-Down-Right-/6, Mem: 104, Examined: 59

Χειρότερη Κατάσταση: Δεν βρίσκει λύση