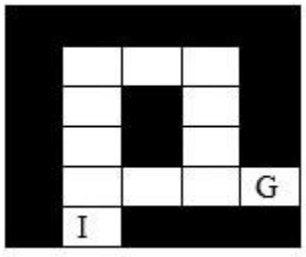
<<Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα>>

Νεκταρία Ευαγγελία Καλλιούπη

Υλοποίηση σε προγραμματιστική γλώσσα JAVA

Θέμα Project: Αναπτύξτε πρόγραμμα υπολογισμού της διαδρομής εξόδου στον παρακάτω λαβύρινθο (I: είσοδος, G: έξοδος).



➔ Αλγόριθμος Επίλυσης BFS branch and bound

I => Αρχική Κατάσταση

G => Τελική Κατάσταση

1.Μέλετη και Σύγκριση Βασικών Αλγορίθμων Τυφλής Αναζήτησης

Ο αλγόριθμος που καλούμαστε να υλοποιήσουμε ανήκει στην κατηγορία των αλγορίθμων τυφλής αναζήτησης. Αυτή η κατηγοριοποίηση έχει γίνει με βάση κάποιες ιδιότητες του αλγορίθμου .Συγκεκριμένα , στους αλγορίθμους αυτούς δεν γίνεται αξιολόγηση των καταστάσεων με βάση κάποια πληροφορία(πχ τι απεικονίζουν) ,συνεπώς αντιμετωπίζουν με τον ίδιο τρόπο όλα τα προβλήματα .

Όλη η βαρύτητα δίνεται στην χρονική σειρά με την οποία παράγονται και εξετάζονται οι καταστάσεις .

Έτσι έχουμε δυο βασικές <<κατηγορίες>> αλγορίθμων τυφλής αναζήτησης τον DFS(αναζήτηση σε βάθος)& BFS(αναζήτηση σε πλάτος) ,με βάση των οποίων εκτελούνται και άλλοι αλγόριθμοι(πχ branch and bound-B&B).

Για την κατανόηση της επίλυσης της εργασίας είναι αναγκαίο να δούμε κάποιές από τις ιδιότητες και διαφορές αυτών των αλγορίθμων .

DFS (αναζήτηση σε βάθος)

- Προτεραιότητα στα παιδιά .
- Το μέτωπο αναζήτησης είναι μια δομή LIFO(LAST IN FIRST OUT).
- Το μέτωπο αναζήτησης δεν μεγαλώνει πολύ άρα έχει μικρές απαιτήσεις σε χώρο .
- Η πρώτη λύση που θα βρούμε , δεν είναι σίγουρο πως θα είναι η βέλτιστη .(*1,δες σελ 4)
- Εφαρμόζοντας τον DFS , υπάρχει κίνδυνος να μην βρούμε ποτέ λύση .Αυτό μπορεί να συμβεί ,διότι 1) δεν υπάρχει έλεγχος για το αν ο χώρος αναζήτησης είναι περιορισμένος καθώς και 2) έλεγχος βρόχων .Συνεπώς ο DFS μπορεί να <<μπλεχτεί>> σε μονοπάτια άπειρου ή τεράστιου μήκους .(*2 ,δες σελ 4)

BFS(αναζήτηση σε πλάτος)

- Προτεραιότητα στις υπάρχουσες καταστάσεις .
- Το μέτωπο αναζήτησης είναι μια δομή FIFO(FIRST IN FIRST OUT).
- Το μέτωπο αναζήτησης μεγαλώνει πολύ(η επέκταση των καταστάσεων μεγαλώνει εκθετικά με το βάθος αναζήτησης),άρα μεγάλες απαιτήσεις μνήμης .
- Ο BFS βρίσκει πάντα την μικρότερη σε μέγεθος λύση-Καλύτερη λύση όταν οι τελεστές μετάβασης είναι όλοι ίδιου κόστος(όπως στο πρόβλημα του λαβυρίνθου με το οποίο ασχολούμαστε). (*3 δες σελ4)

Η πιο κατάλληλη επιλογή αλγορίθμου επίλυσης μπορεί να γίνει μόνο αν γνωρίζουμε την μορφή του χώρου αναζήτησης !

Branch and Bound-B&B(Επέκταση και Οριοθέτηση)

- Εφαρμόζεται σε προβλήματα που αναζητείται η βέλτιστη λύση(ελάχιστο κόστος) .
- Δεν εξετάζει καταστάσεις ,οι οποίες είναι σίγουρο ότι δεν οδηγούν σε καλύτερη λύση .
- Εφαρμόζεται συνήθως σε προβλήματα ,στα οποία οι τελεστές μετάβασης δεν είναι όλοι του ίδιου κόστους .

2.Επίλυση

ΠΑΡΑΔΟΧΗ=> Έχουμε θεωρήσει ότι όλοι οι τελεστές μετάβασης έχουν το ίδιο κόστος (1) και ως τελεστή μετάβασης θεωρούμε την μετάβαση από μια κατάσταση σε μια άλλη

Ο λαβύρινθος του σχήματος ,παρατηρούμε πως έχει τα εξής χαρακτηριστικά :

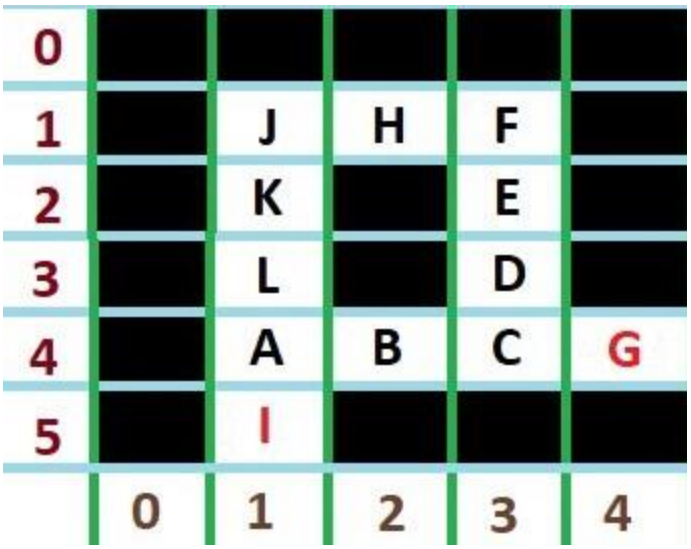
- I => Αρχική Κατάσταση
  G => Τελική Κατάσταση
- Τελεστές Μετάβασης ίδιου κόστους (1 : δηλαδή από την κατάσταση I η μετάβαση στην A έχει κόστος 1)
- Καταστάσεις (''black'') στις οποίες δεν μπορεί να γίνει μετάβαση .

Συνεπώς για την υλοποίηση του λαβυρίνθου , χρησιμοποιήσαμε μια δομή πίνακα διαστάσεων 6(γραμμές)X 5(στήλες) τύπου String.

Στον πίνακα αυτόν ,

- Τις θέσεις του λαβυρίνθου , στις οποίες δεν μπορεί να γίνει μετάβαση (μαύρο χρώμα στο σχήμα) ,τις έχουμε εκχωρήσει στον πίνακα ως στοιχεία με τιμή ''black''.
- Τις θέσεις του λαβυρίνθου , στις οποίες η μετάβαση είναι δυνατή , τις έχουμε εκχωρήσει στον πίνακα ως στοιχεία με τιμές από ''A'' έως ''L" .

Για καλύτερη κατανόηση παρέχεται οπτική αναπαράσταση :



Για την επίλυση του προβλήματος ,θα χρησιμοποιήσουμε αλγόριθμο BFS BRANCH AND BOUND .

Ο B&B μπορεί να υλοποιηθεί με διάφορες παραλλαγές .

Στο συγκεκριμένο πρόβλημα λόγω των παρατηρήσεων που αναφέρθηκαν παραπάνω (1,2,3) σε συνδυασμό με το γεγονός ότι έχουμε πλήρη γνώση του χώρου αναζήτησης καταλήγουμε στο συμπέρασμα ότι είναι προτιμότερη η χρήση του BFS σε συνδυασμό με τον B&B.

Συγκεκριμένα , επειδή ο χώρος αναζήτησης είναι πεπερασμένος και οι τελεστές μετάβασης είναι όλοι ίδιου κόστους ,σε συνάρτηση με τα 1,2,3 μπορούμε να συμπεράνουμε πως η χρήση του BFS B&B είναι πιο αποτελεσματική.

Σύντομη Περιγραφή Κώδικα

ΠΑΡΑΔΟΧΗ=> Έχουμε θεωρήσει ότι όλοι οι τελεστές μετάβασης έχουν το ίδιο κόστος (1) και ως τελεστή μετάβασης θεωρούμε την μετάβαση από μια κατάσταση σε μια άλλη

Η υλοποίηση του προγράμματος έγινε με βάση τον αλγόριθμο B&B που παρέχεται στο βιβλίο καθώς και στις σημειώσεις του μαθήματος ,με μόνη διαφορά ότι αυτήν την φορά οι καταστάσεις παιδιά εισέρχονται στο τέλος του μέτωπού αναζήτησης (BFS) ,συγκεκριμένα :

- ➔ Στο μέτωπο αναζήτησης(searchSet-> arrayList τύπου String) διατηρούμε τα μονοπάτια των καταστάσεων που χρειάζεται να εξετάσουμε .
- ➔ Στο κλειστό σύνολο (closedSet->arrayList τύπου String) στο οποιο διατηρούμε τις καταστάσεις (όχι τα μονοπάτια) που έχουμε ήδη εξετάσει .
- ➔ Στο μικροσκόπιο(microscope-> μεταβλητή τύπου String), εισέρχεται σε κάθε επανάληψη μια κατάσταση την οποία πρέπει να εξετάσουμε . Η κατάσταση αυτή λαμβάνεται με την εξής διαδικασία : από το πρώτο μονοπάτι (searchSet.get(0)) που βρίσκεται στο μέτωπο αναζήτησης ,με την εντολή :

```
microscope=searchSet.get(0).substring(searchSet.get(0).length()-1 );//inserting last state of the path into the microscope
```

Παίρνουμε την τελευταία κατάσταση από αυτό το μονοπάτι και την εκχωρούμε στο μικροσκόπιο.

Στην συνέχεια πραγματοποιούμε έναν έλεγχο που οδηγεί σε τρείς περιπτώσεις :

1) <u>Η κατάσταση έχει ήδη εξεταστεί (εμπεριέχεται στο κλειστό σύνολο) .</u>
   Σε αυτήν την περίπτωσή ,το πρόγραμμα δεν βρίσκει τις καταστάσεις παιδιά ,εμφανίζει μήνυμα στο χρήστη και συνεχίζει στην επόμενη επανάληψη ,αν αυτή υπάρχει .

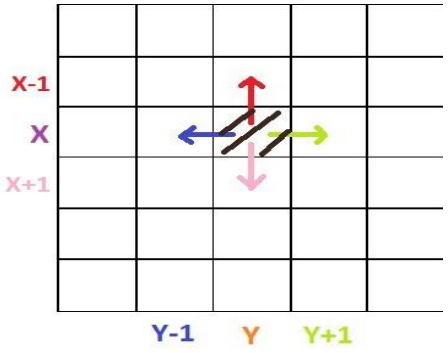
2) <u>Η κατάσταση ταυτίζεται με την τελική ,οπότε έχουμε νέα λύση</u> .
   Σε αυτήν την περίπτωσή, το πρόγραμμα εκχωρεί στην μεταβλητή finalSolution το path(Current Path in microscope) με την εντολή(searchSet.get(0)) και ενημερώνει το όριο(limit).

   **Υποσημείωση => αφού έχουμε θεωρήσει ότι όλοι οι τελεστές μετάβασης έχουν το ίδιο κόστος (1) και επειδή ως τελεστή μετάβασης έχουμε θεωρήσει την μετάβαση από μια κατάσταση σε μια άλλη μπορούμε ευκολά να βρούμε με το όριο παίρνοντας το length του path και αφαιρώντας 1 (limit=searchSet.get(0).length()-1).

3) <u>Το άθροισμα των τελεστών μετάβασης του μονοπατιού είναι μεγαλύτερο του ορίου(limit)</u> .
   Σε αυτήν την περίπτωσή το πρόγραμμα δεν βρίσκει τις καταστάσεις παιδιά ,εμφανίζει μήνυμα στο χρήστη ότι δεν έχει νόημα η περαιτέρω εξέταση του μονοπατιού αφού δεν θα οδηγήσει σε καλύτερη λύση και συνεχίζει στην επόμενη επανάληψη ,αν αυτή υπάρχει .

4) <u>Η κατάσταση δεν έχει εξεταστεί και συνεπώς χρειάζεται να βρούμε τις καταστάσεις παιδιά .</u>
   Σε αυτήν την περίπτωσή το πρόγραμμα βρίσκει μια μια τις καταστάσεις παιδιά με την λογική που παρατίθεται παρακάτω και στην συνεχεία με την εντολή(String childPath=searchSet.get(0) + newState) δημιουργεί το μονοπάτι παιδί που περιέχει το προηγούμενο μονοπάτι και την κατάσταση παιδί.
   Πχ μονοπάτι= IA
   παιδί=L
   μονοπάτι παιδί = IAL

➔ <u>Λογική Εύρεσης Καταστάσεων παιδιών</u>



Για να βρούμε τις καταστάσεις παιδία μιας κατάστασης στον λαβύρινθο ,πρέπει απλά να βρούμε τα γειτονικά στοιχεία της κατάστασης ,των οποίων οι συντεταγμένες φαίνονται στην από πάνω εικόνα και επιπρόσθετα να ελέγξουμε αν στον δικό μας λαβύρινθο (πίνακας 6X5) οι καταστάσεις αυτές είναι καταχωρημένες ως ‘’black’’.Αν είναι τότε ,μιλάμε για καταστάσεις στις οποίες δεν μπορούμε να μετάβουμε και έτσι δεν τις συμπεριλαμβάνουμε στα παιδιά .

Κώδικας Εύρεσης Καταστάσεων Παιδιά :

```
//find children of the <<microscope>> state
 if((x-1)>=0){
        if((maze[x-1][y] != "black")) {

            String newState= maze[x-1][y];
            String childPath=searchSet.get(0) + newState;
            children.add(childPath);



        }
     }

     if(((y-1)>=0)){
            if(maze[x][y-1] != "black") {
                String newState=maze[x][y-1] ;
                String childPath=searchSet.get(0) + newState;
                children.add(childPath);

            }
    }



     if(((x+1)<=5)){
        if(maze[x+1][y]!= "black") {
            String newState= maze[x+1][y];
            String childPath=searchSet.get(0) + newState;
            children.add(childPath);

        }
    }


    if(((y+1)<=4)){
      if(maze[x][y+1]!= "black") {
          String newState=maze[x][y+1];
            String childPath=searchSet.get(0) + newState;
            children.add(childPath);


      }
   }
```

Μετά από την επιλογή μιας εκ των παραπάνω περιπτώσεων , το πρόγραμμα προσθέτει την κατάσταση που βρίσκεται στο μικροσκόπιο στο κλειστό σύνολο (αν δεν συμπεριλαμβάνεται ήδη) και αφαιρεί το μονοπάτι της κατάστασης που εξετάσαμε (Current Path in microscope)από το σύνολο αναζήτησης .

Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να αδειάσει το μέτωπο αναζήτησης ,οπότε και εμφανίζουμε στον χρήστη την τελική λύση και το κόστος της .

Παράδειγμα Εκτέλεσης

```
Initial State : I
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ I ]
Closed Set : [  ]
Current Path in microscope:I
Current State in microscope:I
Children States : [ IA ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IA ]
Closed Set : [ I ]
Current Path in microscope:IA
Current State in microscope:A
Children States : [ IAL , IAI , IAB ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IAL , IAI , IAB ]
Closed Set : [ I , A ]
Current Path in microscope:IAL
Current State in microscope:L
Children States : [ IALK , IALA ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IAI , IAB , IALK , IALA ]
Closed Set : [ I , A , L ]
Current Path in microscope:IAI
Current State in microscope:I
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IAB , IALK , IALA ]
Closed Set : [ I , A , L ]
Current Path in microscope:IAB
Current State in microscope:B
Children States : [ IABA , IABC ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALK , IALA , IABA , IABC ]
Closed Set : [ I , A , L , B ]
Current Path in microscope:IALK
Current State in microscope:K
Children States : [ IALKJ , IALKL ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALA , IABA , IABC , IALKJ , IALKL ]
Closed Set : [ I , A , L , B , K ]
Current Path in microscope:IALA
Current State in microscope:A
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABA , IABC , IALKJ , IALKL ]
Closed Set : [ I , A , L , B , K ]
Current Path in microscope:IABA
Current State in microscope:A
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABC , IALKJ , IALKL ]
Closed Set : [ I , A , L , B , K ]
Current Path in microscope:IABC
Current State in microscope:C
Children States : [ IABCD , IABCB , IABCG ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALKJ , IALKL , IABCD , IABCB , IABCG ]
Closed Set : [ I , A , L , B , K , C ]
Current Path in microscope:IALKJ
Current State in microscope:J
Children States : [ IALKJK , IALKJH ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALKL , IABCD , IABCB , IABCG , IALKJK , IALKJH ]
Closed Set : [ I , A , L , B , K , C , J ]
Current Path in microscope:IALKL
Current State in microscope:L
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABCD , IABCB , IABCG , IALKJK , IALKJH ]
Closed Set : [ I , A , L , B , K , C , J ]
Current Path in microscope:IABCD
Current State in microscope:D
Children States : [ IABCDE , IABCDC ]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABCB , IABCG , IALKJK , IALKJH , IABCDE , IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D ]
Current Path in microscope:IABCB
Current State in microscope:B
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABCG , IALKJK , IALKJH , IABCDE , IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D ]
Current Path in microscope:IABCG
Current State in microscope:G

! Solution Found ! : IABCG

Path Length : 4  --> New limit!!(best solution/path we have so far)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALKJK , IALKJH , IABCDE , IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D , G ]
Current Path in microscope:IALKJK
Current State in microscope:K
Path length is too long in comparison to the current limit(path length of the best solution we have so far).As a result we have to <<cut>>(not examine) this certain path since it will not provide a better solution .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IALKJH , IABCDE , IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D , G ]
Current Path in microscope:IALKJH
Current State in microscope:H
Path length is too long in comparison to the current limit(path length of the best solution we have so far).As a result we have to <<cut>>(not examine) this certain path since it will not provide a better solution .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABCDE , IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D , G , H ]
Current Path in microscope:IABCDE
Current State in microscope:E
Path length is too long in comparison to the current limit(path length of the best solution we have so far).As a result we have to <<cut>>(not examine) this certain path since it will not provide a better solution .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Search Set : [ IABCDC ]
Closed Set : [ I , A , L , B , K , C , J , D , G , H , E ]
Current Path in microscope:IABCDC
Current State in microscope:C
Path length is too long in comparison to the current limit(path length of the best solution we have so far).As a result we have to <<cut>>(not examine) this certain path since it will not provide a better solution .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


!Final Solution! : IABCG

Path length:4


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Τα παραπάνω στιγμιότυπα εμπεριέχονται στο pdf αρχείο με όνομα results.pdf