Τεχνητή Νοημοσύνη

1η Εργασία

Ημερομηνία: 11-11-2015

ΟΝ/ΜΟ: Κωνσταντάκης Δημήτριος

ΑΜ: 1115201300079

Πρόβλημα 1:

Το πρόγραμμα αποτελείται από τα επιμέρους αρχεία: *BlocksWorldPloblem.py* , *BlocksWorldPloblemMain.py*, *search.py* και *utils.py* . Οι υλοποιήσεις που έχουν γίνει είναι η κατασκευή της κλάσης Blocks World Problem και των σθναρτήσεων που περιέχει. Στο *BlocksWorldPloblemMain.py* έχουν υλοποιηθεί οι ευρετικές συναρτήσεις h1(υπήρχε ήδη), h2, h3 , καθώς και μία συνάρτηση καταμέτρησης χρόνου που εκτελούνταν πειραματικά. Η αρχική κατάσταση και η τελική κατάσταση ορίζεται στην κλάση *BlocksWorld Ploblem.*

Τα αποτελέσματα που προέκυψαν κατά μέσο όρο παίρνοντας 5 κύβους για τη δοθείσα αρχική κατάσταση για κάθε ευρετική είναι:

Ευρετικές	**Nodes**	**Time (sec)**
h1	320	0.96909
h2	161	0.61171
h3	52	0.25044

Τα αποτελέσματα που προέκυψαν κατά μέσο όρο παίρνοντας 5 κύβους για τη μία τυχαία αρχική κατάσταση για κάθε ευρετική είναι:

Ευρετικές	**Nodes**	**Time (sec)**
h1	224	0.77130
h2	83	0.37472
h3	44	0.20035

Τα αποτελέσματα που προέκυψαν κατά μέσο όρο παίρνοντας 6 κύβους για κάθε ευρετική είναι:

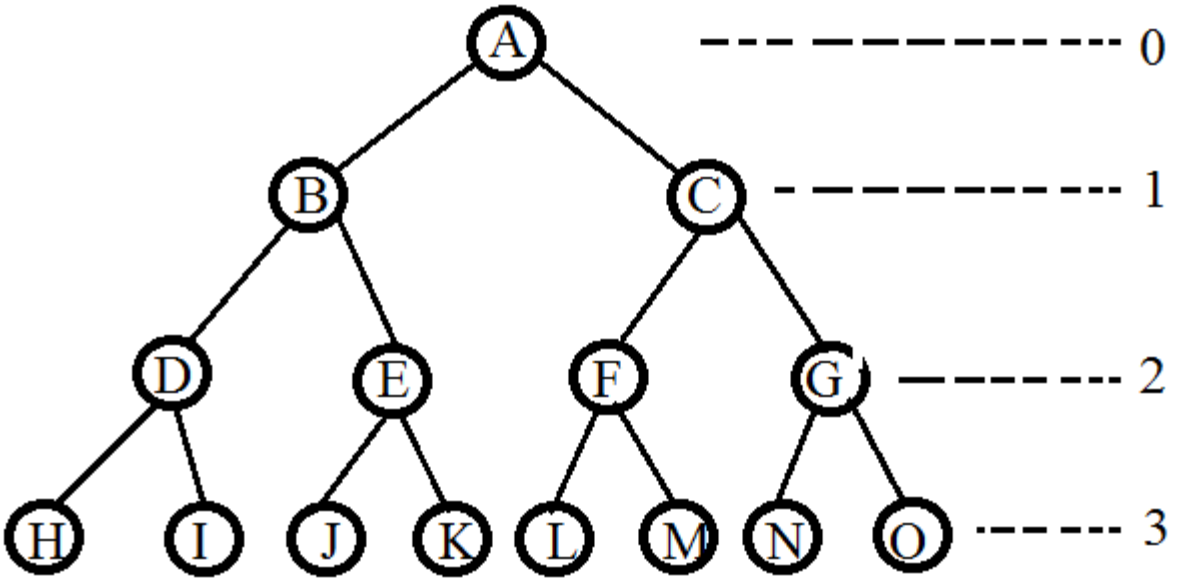
Ευρετικές	**Nodes**	**Time (sec)**
h1	1334	37.09574
h2	211	2.58163
h3	435	17.65923

Τα αποτελέσματα που προέκυψαν κατά μέσο όρο παίρνοντας 7 κύβους για κάθε ευρετική ήταν αδύνατο να υπολογιστούν. Άρα ο μέγιστος αριθμός κύβων είναι 6.

Γενικά παρατηρούμε ότι η h1 είναι η χειρότερη ευρετική συνάρτηση. Δημιουργέι σε όλες τις περιπτώσεις τους περισσότερους κόμβους. Αντίθετα, οι h2 και h3 είναι καλύτερες από την h1 . Και οι δύο ευρετικές είναι συνεπείς, καθώς ισχύει η ανισότητα $h(n) \geq c(n, a, n') + h(n')$, άρα είναι και παραδεχτές.

Πρόβλημα 2:

Έστω ότι έχουμε το δέντρο T με βάθος g = 3 και παράγοντα διακλάδωσης b = 2.



Ο αλγόριθμος IDF , αν υποθέσουμε πως ο κόμβος στόχος είναι ο H τότε θα τρέξει ως εξής:

Η βέλτιστη περίπτωση, για βάθος g, είναι για τον κόμβο D ,ενώ η χείριστη για τον κόμβο G. Για την βέλτιστη, θα πρέπει να γίνει επαναληπτική εκβάθυνση μέχρι το βάθος 2 και στη συνέχεια DFS μέχρι τον D. Για τη χείριστη, επαναληπτική εκβάθυνση μέχρι το βάθος 3 , αφαιρώντας τους κόμβους που προκύπτουν από την αναζήτηση DFS για τον κόμβο G , καθώς είναι ο κόμβος – στόχος.

Βέλτιστη περίπτωση:

Ο αριθμός των κόμβων που δημιουργούνται από μία αναζήτηση IDS μέχρι ένα βάθος g είναι, με παράγοντα διακλάδωσης b :

$(g + 1) + g*b + (g - 1)*b^2 + \ldots + 2b^{g-1} + 1b^g$

Αν σκεφτούμε όμως ότι στο βάθος g θα προστεθούν άλλοι b κόμβοι , για να φτάσουμε στον κόμβο – στόχο , τότε ο συνολικός αριθμός των κόμβων γίνεται:

$(g + 1) + g*b + (g - 1)*b^2 + \ldots + 2b^{g-1} + 1b^g + (g + 1)$

Χείριστη περίπτωση:

Ο αριθμός των κόμβων που δημιουργούνται από μία αναζήτηση IDS μέχρι ένα βάθος g είναι, με παράγοντα διακλάδωσης b :

$(g + 1) + g*b + (g-1)*b^2 + \ldots + 2b^{g-1} + 1b^g$

Για να φτάσουμε στον κόμβο – στόχο που βρίσκεται σε βάθος g , τότε ο συνολικός αριθμός των κόμβων γίνεται:

$(g + 1) + g*b + (g-1)*b^2 + \ldots + 2b^{g-1} + 1b^g = \sum_{i=1}^{g}(g + 1 - i)b^i$

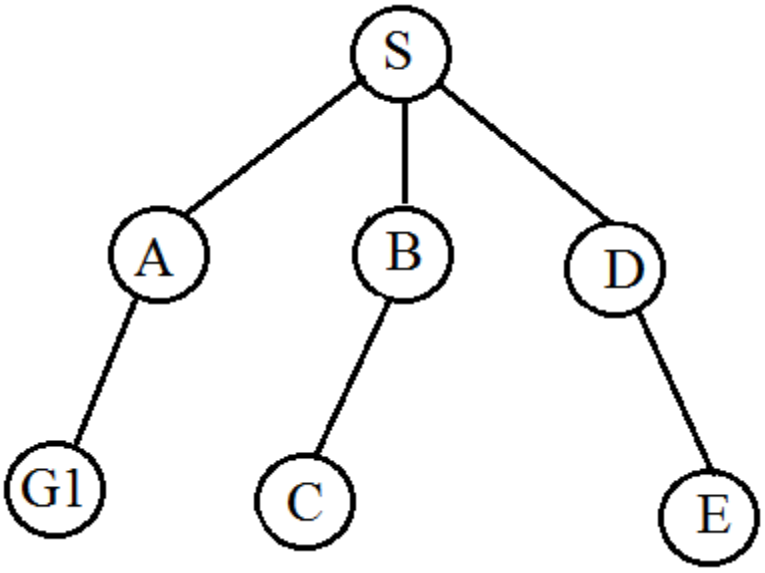
Πρόβλημα 3:

Ο αλγόριθμος αναζήτησης στους γράφους είναι ο εξής:

```
function GRAPHSEARCH(problem,QUEUINGFN)
returns a solution, or failure
    closed ← an empty set
    fringe ← MAKEQUEUE(MAKENODE(INITIALSTATE[problem]))
    loop do
        if fringe is empty then return failure
        node ← REMOVEFRONT(fringe)
        if GOALTEST[problem] applied to STATE[node] succeeds then
            return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            fringe ← QUEUINGFN(EXPAND(node, problem), fringe)
    end
```

Ο παραπάνω αλγόριθμος, δηλαδή, τοποθετεί τους κόμβους που συναντά σε μία κλειστή λίστα. Οπότε την επόμενη φορά που θα τους συναντήσει , δεν θα τους προσθέσει στην ουρά. Έτσι, αποφεύγουμε τις επαναλήψεις μέσα στον γράφο.

<u>BFS</u>

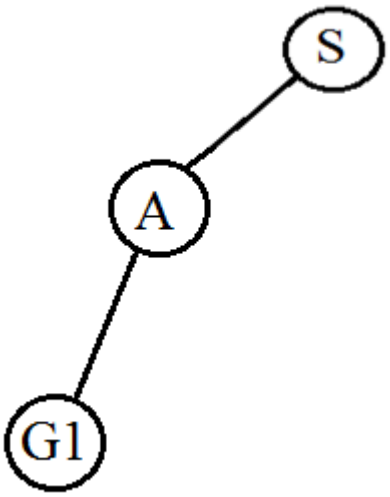


Η ουρά παίρνει τους εξής κόμβους:

{S}, {A, B, D } , { B, D, G1} , { D, G1, C}, { G1, C, E}

Ο επόμενος κόμβος που θα ελεγχθεί είναι ο G1, ο οποίος αντιστοιχεί σε κατάσταση στόχου. Το σύνορο είναι : { G1, C, E}

DFS

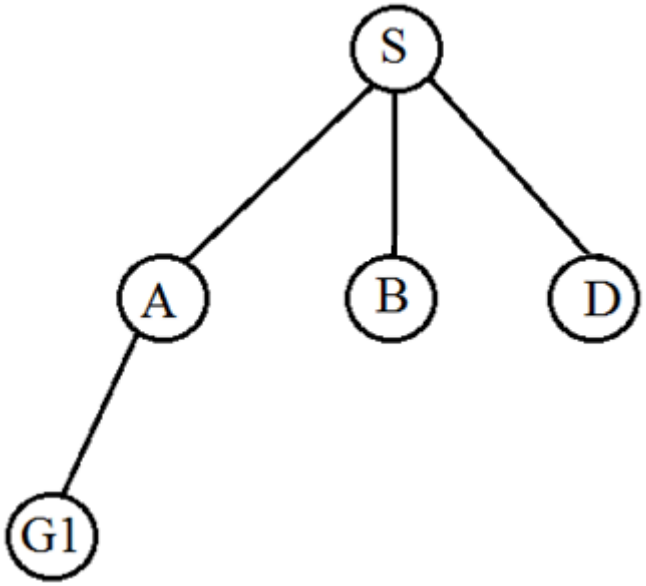


Η ουρά παίρνει τους εξής κόμβους:

{S}, {A, S}, {G1, A, S}

Ο επόμενος κόμβος που θα ελεγχθεί είναι ο G2, ο οποίος αντιστοιχεί σε κατάσταση στόχου. Το σύνορο είναι : {G2, A, S}

IDS

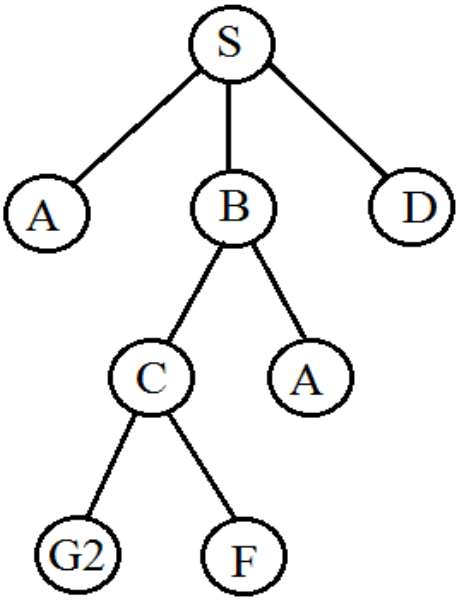


Η ουρά παίρνει τους εξής κόμβους:

{S}, {A, B, D} , { G1, B, D}

Ο επόμενος κόμβος που θα ελεγχθεί είναι ο G1, ο οποίος αντιστοιχεί σε κατάσταση στόχου. Το σύνορο είναι : { G1, B, D}

Greedy



Η ουρά παίρνει τους εξής κόμβους:

{S}, {B} , {C}, {G2}

Ο επόμενος κόμβος που θα ελεγχθεί είναι ο G1, ο οποίος αντιστοιχεί σε κατάσταση στόχου. Το σύνορο είναι : {C}

A*

f = g + h

- S, 0 + 5
- S – A , (0+5) + 7 = 12
  S – B , (0+9) + 3 = 12
  S – D , (0+6) + 6 = 12
- S – A – B, (5+3) + 3
  S – A – G1, (5+9) + 0
  S – B , (0+9) + 3

S – D , (0+6) + 6

- S – A – B – A, (8+2) + 7
  - S – A – B – C, (8+1) + 4
  - S – A – G1, (5+9) + 0
  - S – B , (0+9) + 3
  - S – D , (0+6) + 6
- S – A – B – A, (8+2) + 7
  - S – A – B – C, (8+1) + 4
  - S – A – G1, (5+9) + 0
  - S – B – A , (9+2) + 7
  - S – B – C , (9+1) + 4
  - S – D , (0+6) + 6
- S – A – B – A, (8+2) + 7
  - S – A – B – C, (8+1) + 4
  - S – A – G1, (5+9) + 0
  - S – B – A , (9+2) + 7
  - S – B – C , (9+1) + 4
  - S – D – C , (6+2) + 4
  - S – D – E , (6+2) + 5

- S – A – B – A, (8+2) + 7
  - S – A – B – C, (8+1) + 4
  - S – A – G1, (5+9) + 0
  - S – B – A , (9+2) + 7
  - S – B – C , (9+1) + 4
  - S – D – C – G2, (8+5) + 0
  - S – D – C – F, (8+7) + 6
  - S – D – E , (6+2) + 5
- S – A – B – A, (8+2) + 7
  - S – A – B – C – G2 , (9+5) + 0
  - S – A – B – C – F , (9+7) + 6
  - S – A – G1, (5+9) + 0
  - S – B – A , (9+2) + 7
  - S – B – C , (9+1) + 4
  - S – D – C – G2, (8+5) + 0
  - S – D – C – F, (8+7) + 6

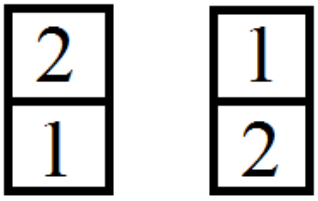
S – D – E , (6+2) + 5

- S – A – B – A, (8+2) + 7
  S – A – B – C – G2 , (9+5) + 0
  S – A – B – C – F , (9+7) + 6
  S – A – G1, (5+9) + 0
  S – B – A , (9+2) + 7
  S – B – C , (9+1) + 4
  S – D – C – G2, (8+5) + 0
  S – D – C – F, (8+7) + 6
  S – D – E , (6+2) + 5

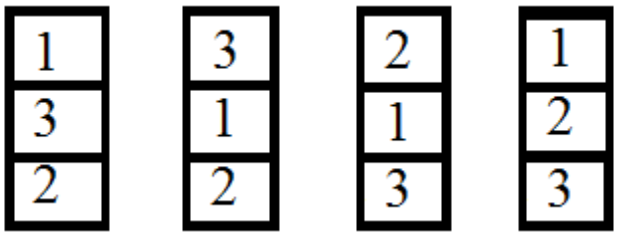
  Παίρνουμε το S – D – C – G2 , βλέπουμε ότι είναι κατάσταση στόχου το G2.

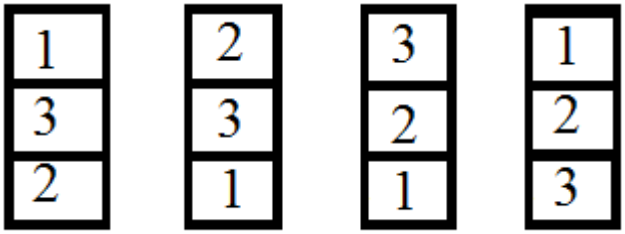
Πρόβλημα 4:

I. Για n = 1, f(n) = 0
   Για n = 2, f(n) = 1



Για n = 3, f(n) = 3





Για n = 4, f(n) = 4

Αν η μεγαλύτερη πίτα βρίσκεται στην κορυφή θα γίνουν 1 + f(3) = 4 κινήσεις . Αν η μεγαλύτερη πίτα βρίσκεται στον πάτο,τότε θα γίνουν f(3) = 3 κινήσεις. Αν βρίσκεται στην $2^{η}$ ή $3^{η}$ θέση τότε θα γίνει διερύνηση.

1	3	4	1	
2	4	3	2	
4	2	2	3	
3	1	1	4	
2	4	3	1	
1	1	2	2	
4	2	1	3	
3	3	4	4	
3	1	4	2	1
1	3	3	1	2
4	4	1	3	3
2	2	2	4	4
1	4	2	1	
3	3	1	2	
4	1	3	3	
2	2	4	4	
2	4	1		
3	3	2		
4	2	3		
1	1	4		

3	2	4	1	
2	3	3	2	
4	4	2	3	
1	1	1	4	
1	4	3	1	
4	1	2	2	
2	2	1	3	
3	3	4	4	
1	4	2	3	1
4	1	3	2	2
3	3	1	1	3
2	2	4	4	4
2	4	3	2	1
4	2	1	1	2
1	1	2	3	3
3	3	4	4	4
2	3	4	1	
4	4	3	2	
3	2	2	3	
1	1	1	4	
3	4	1		
4	3	2		
2	2	3		
1	1	4		

3	4	2	1
4	3	1	2
1	1	3	3
2	2	4	4

Γενικά παρατηρούμε ότι σε καμία περίπτωση δεν χρειάστηκε να κάνουμε πάνω από 4 κινήσεις . Οπότε f(4) = 4.

II. Αφού δείξαμε ότι για n = 4 έχουμε f(4) = 4, για n ≥ 4 έχουμε f(n) ≥ 4.

Πράγματι, αν έχω n πίτες , και η μεγαλύτερη πίτα δεν είναι στον πάτο, τότε θα χρειαστούμε το λιγότερο f(n-1) + 1 κινήσεις . Συνεπώς, με επαγωγή προκύπει ότι f(n-1) ≥ n-1 , όπου αν θέσουμε n-1 = n

=> f(n) ≥ n.

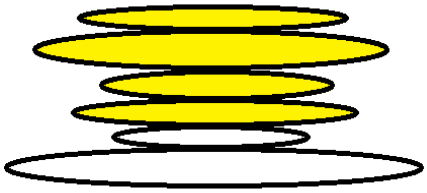
III. Αν σκεφτούμε ότι η πιο απλή αλλά και ταυτόχρονα η πιο αργή λύση για να επιλύσουμε το πρόβλημα των πιτών είναι να τις αναποδογυρίσουμε με τέτοιο τρόπο ώστε να είναι η μεγαλύτερη πίτα που δεν βρίσκεται στην θέση της στην κορυφή και στη συνέχεια να τις αναποδογυρίζουμε όλες μαζί για να πάει η μεγαλύτερη πίτα στη θέση που πρέπει. Δηλαδή για κάθε πίτα θα χρειαστούμε δύο κινήσεις, και αν έχουμε n πίτες οι συνολικές κινήσεις θα είναι $f(n) \leq 2n$

4	5	3	4	1	3	2	1
1	1	2	2	3	1	1	2
5	4	4	3	2	2	3	3
2	3	1	1	4	4	4	4
3	2	5	5	5	5	5	5

IV. Έκφραση Προβλήματος:

- Κατάσταση: Κάθε πίτα έχει το δικό της μοναδικό μέγεθος το οποίο εκφράζεται ως αριθμός (π.χ. 1 για την μικρότερη, 2 για την αμέσως μεγαλύτερη, κ.ο.κ). Οι πίτες τοποθετούνται σε στοίβα. Οποιαδήποτε ανακατανομή των πιτών της στοίβας θεωρείται κατάσταση.
- Αρχική Κατάσταση: Μπορεί να θεωρηθεί οποιαδήποτε κατάσταση.

- Κατάσταση Στόχος: Θεωρείται η επιθυμητή τελική κατάσταση , που στο συγκεκριμένο πρόβλημα είναι η κατάσταση στην οποία οι πίτες είναι τοποθετημένες από την μεγαλύτερη στην μικρότερη.(από κάτω προς τα πάνω).Δηλαδή το κατώτερο στοιχείο της στοίβας είναι και ο μεγαλύτερος αριθμός, κ.ο.κ.
- Ενέργειες: Επιλογή ενός αριθμού πιτών.



Αλλαγή θέσεων - τουμπάρισμα

- Έλεγχος Κατάστασης: Ελέγχει αν η κατάσταση που βρισκόμαστε είναι κατάσταση στόχου.
- Κόστος: Το κόστος είναι 1 για όλες τις ενέργειες, συνεπώς το συνολικό κόστος είναι ίσο με το σύνολο των ενεργειών που έχουν γίνει.
- Δράση – Συνάρτηση Δημιουργίας Διαδόχων: Ανάλογα με τον αλγόριθμο αναζήτησης , επιλέγεται μια συγκεκριμένη ενέργεια και με βάση εκείνη παράγεται μια κατάσταση.
- Πολυπλοκότητα. Έστω n ο αριθμός των πιτών. Κάθε φορά μπορούμε να εκτελέσουμε n-1 ενέργειες που μπορούν να μας οδηγήσουν σε n-1 καταστάσεις. Άρα, ο παράγοντας διακλάδω-σης είναι b = n-1 .

  Αν χρησιμοποιήσουμε τον αλγόριθμο αναζήτησης BFS τότε το μέγεθος του χώρου είναι $O(b^{d+1})$ καθώς έχουμε το ίδο μη αρνητι-κό κόστος σε κάθε κατάσταση.

Αν χρησιμοποιήσουμε τον αλγόριθμο αναζήτησης DFS τότε το μέγεθος του χώρου είναι $O(bd)$ καθώς έχουμε το ίδο μη αρνητι-κό κόστος σε κάθε κατάσταση. Ομοίως και για τον αλγόριθμο IDS.

Ο Σήφης από τη στιγμή που είναι ειδικός στο αναποδογύρισμα των πιτών , θα επιθυμούσε τη βέλτιστη αναζήτηση , δηλαδή την ανάζητηση Πρώτα σε Βάθος ή την αναζήτηση Επαναληπτικής Εκβάθυνσης.