Project 1 για το μάθημα της Τεχνητής νοημοσύνης Ι

Ονοματεπώνυμο: Απόστολος Καρβέλας

ΑΜ: 1115201800312

Πρόβλημα 2:

Ο αριθμός των κόμβων που δημιουργούνται από μια αναζήτηση με τον IDS μέχρι το βάθος d είναι: $(d+1)+db+(d-1)b^2+\cdots+2b^{d-1}+1b^d$ (I)

Άρα ο μικρότερος αριθμός κόμβων που μπορούν να δημιουργηθούν είναι ο τύπος (I) για $d=g$ και χωρίς την τελευταία πρόσθεση αφού ο κόμβος στόχος μπορεί να βρίσκεται στην πιο αριστερή πλευρά του δένδρου, οπότε:

$$(g+1)+gb+(g-1)b^2+\cdots+2b^{g-1}+1$$

Που ισοδυναμεί με την σειρά $\sum_{i=0}^{g-1}(g+1-i)b^i+1$

Ενώ ο μεγαλύτερος αριθμός κόμβων είναι ο τύπος (I) δηλαδή στην χειρότερη περίπτωση : $(g+1)+gb+(g-1)b^2+\cdots+2b^{g-1}+1b^g$

Που ισοδυναμεί με την σειρά: $\sum_{i=0}^{g}(g+1-i)b^i$.

Πρόβλημα 3:

Α) Χρησιμοποιούμε αντιπαράδειγμα για το κόμβο ts. Η ευρετική συνάρτηση υπερεκτιμά το πραγματικό κόστος εύρεσης αφού h(ts) = 23 ενώ το κόστος από το o103 είναι 8. Άρα ισχύει h(ts)>κόστους οπότε δεν είναι παραδεκτή.

Μια συνάρτηση για να είναι συνεπείς πρέπει αναγκαστικά να είναι και παραδεκτή οπότε δεν είναι ούτε και συνεπείς.

Β) Αναζήτηση πρώτα σε πλάτος: $o103 \rightarrow b3 \rightarrow o109 \rightarrow ts \rightarrow b1 \rightarrow b4 \rightarrow o111 \rightarrow o119 \rightarrow mail \rightarrow b2 \rightarrow c2 \rightarrow o123 \rightarrow storage \rightarrow c1 \rightarrow c3 \rightarrow o125 \rightarrow r123.$

Αναζήτηση πρώτα σε βάθος: $o103 \rightarrow ts \rightarrow mail \rightarrow o109 \rightarrow o119 \rightarrow storage \rightarrow o123 \rightarrow r123$

Αναζήτηση πρώτα σε βάθος με επαναληπτική εκβάθυνση:

1η επανάληψη: $o103$

2η επανάληψη: $o103 \rightarrow ts \rightarrow o109 \rightarrow b3$

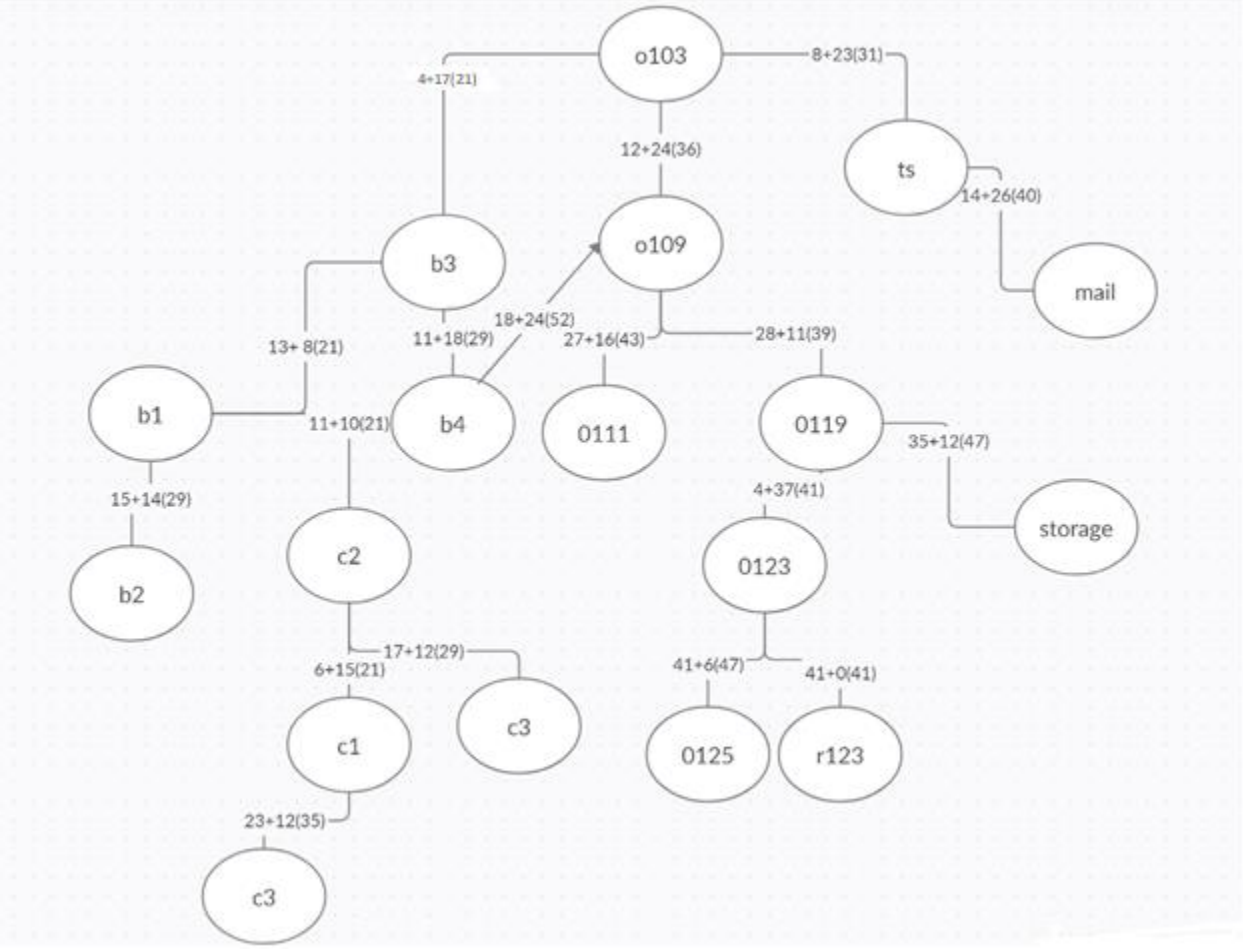
3η επανάληψη: $o103 \rightarrow ts \rightarrow mail \rightarrow o109 \rightarrow o119 \rightarrow o111 \rightarrow b3 \rightarrow b4 \rightarrow b1$

4η επανάληψη: $o103 \rightarrow ts \rightarrow mail \rightarrow o109 \rightarrow o119 \rightarrow storage \rightarrow o123 \rightarrow o111 \rightarrow b3 \rightarrow b4 \rightarrow b1 \rightarrow c2 \rightarrow b2$

5η επανάληψη: : $o103 \rightarrow ts \rightarrow mail \rightarrow o109 \rightarrow o119 \rightarrow storage \rightarrow o123 \rightarrow r123$

Άπληστη αναζήτηση πρώτα στον καλύτερο με ευρετική συνάρτηση h: $o123 \rightarrow b3 \rightarrow b1 \rightarrow c2 \rightarrow c1 \rightarrow c3 \rightarrow b2 \rightarrow b4 \rightarrow ts \rightarrow o109 \rightarrow o119 \rightarrow o123 \rightarrow r123$

A*με ευρετική συνάρτηση h: $o103 \rightarrow b3 \rightarrow b1 \rightarrow c2 \rightarrow c1 \rightarrow b2 \rightarrow b4 \rightarrow c3 \rightarrow ts \rightarrow o109 \rightarrow o119 \rightarrow mail \rightarrow o123 \rightarrow r123$



Πρόβλημα 4:

Α) Ένα πρόβλημα αναζήτησης ορίζεται από 4 μέρη:

- Ένα σύνολο καταστάσεων που στην δικιά μας περίπτωση θα είναι όλοι οι κόμβοι-δωμάτια και τα κόστη τους.
- Μια κατάσταση εκκίνησης που θα είναι μια σειρά από πακέτα και την αρχική θέση, δηλαδή το mail.
- Μια κατάσταση στόχου, η οποία θα ισχύει όταν το ρομπότ βρίσκεται στο δωμάτιο mail και όλα τα πακέτα έχουν παραδοθεί.
- Τέλος, μια συνάρτηση απαρίθμησης που θα επιστρέφει τους επόμενους κόμβους, τις κινήσεις και τα κόστη.

  Όπου πακέτο θα περιέχει 2 στοιχεία, αρχική θέση και προορισμός.

Β) Για μια παραδεκτή ευρετική συνάρτηση πρέπει να έχουμε κάνει πρώτα αλγόριθμο Dijkstra για κάθε κόμβο ώστε να έχουμε τις ελάχιστες αποστάσεις από έναν κόμβο σε κάποιον άλλον. Με αυτόν τον τρόπο θα έχουμε ειδική γνώση για το πρόβλημα.

Η συνάρτηση θα υπολογίζει τις αποστάσεις ως προς κάθε πακέτο + την απόσταση από το πακέτο στον προορισμό του. Στην συνέχεια θα επιστρέφει την μέγιστη απόσταση. Η συνάρτηση θεωρείται παραδεκτή γιατί δεν υπερεκτιμά το πραγματικό κόστος του γράφου αφού ο Dijkstra υπολογίζει την ελάχιστη απόσταση για αυτόν τον κόμβο.

Πρόβλημα 5:

Η αμφίδρομη αναζήτηση για θεωρείται πλήρης πρέπει και οι αναζητήσεις της να είναι και αυτές, οπότε έχουμε:

Α) Δεν είναι πλήρης οπότε σίγουρα δεν είναι και βέλτιστος αφού η αναζήτηση περιορισμένου βάθους δεν είναι πλήρης ενώ ο πρώτα σε πλάτος είναι.

Β) Δεν είναι πλήρης/Δεν είναι βέλτιστος αφού η αναζήτηση με επαναληπτική εκβάθυνση είναι ενώ περιορισμένου βάθος όχι.

Γ) Δεν είναι πλήρης/Δεν είναι βέλτιστος. Η αναζήτηση περιορισμένου βάθους δεν είναι πλήρης ενώ η A* είναι.

Δ) Είναι πλήρης όταν το cost>0 και b είναι infinite αφού και οι 2 αλγόριθμοι είναι A* που υλοποιείται με UCS /Είναι βέλτιστος όταν είναι πλήρης και επιπλέον τα κόστη είναι ίδια.

Ένας αποδοτικός έλεγχος και για τις 4 περιπτώσεις είναι να συγκρίνουν τα 2 σύνορα των αλγορίθμων και αν υπάρχουν κοινοί κόμβοι μεταξύ τους τότε οι 2 αναζητήσεις συναντιούνται και ο αμφίδρομος αλγόριθμος τελειώνει. Ενώ είναι αποδοτικός όταν υπάρχουν παραπάνω από έναν κοινό κόμβο και πάρουμε εκείνο με το λιγότερο κόστος.