Student Name: Κωνσταντίνος Φράγκος
Student ID: sdi2000207

Τεχνητή Νοημοσύνη
Εργασία 1

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Πρόβλημα 1:

Υπάρχει στο φάκελο

Πρόβλημα 2:

Για την εύρεση του μικρότερου αριθμού κόμβων:

Θα γίνει αναζήτηση bfs πρώτα για $d = 0$, μετά για $d = 0$ και $d = 1$ κλπ.. μέχρι $d = g$, όπου επειδή αναζητούμε τον μικρότερο αριθμό κόμβων θα τον βρούμε κατευθείαν στο βάθος $d = g$, δηλαδή σε αυτό το βάθος θα αναζητήσουμε έναν κόμβο μό νο.
Για $d = 0$, αναζητούμε μόνο τον κόμβο ρίζας

$\Rightarrow 1$

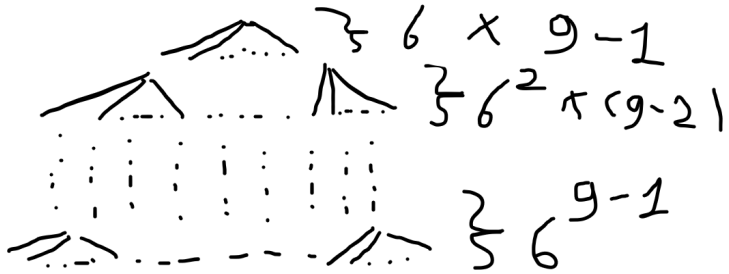
Για $d = 1$, 2 φορές κόμβος ρίζας +

$\Rightarrow 2 + b$

Για $d = 2$, 3 φορές κόμβος ρίζας +

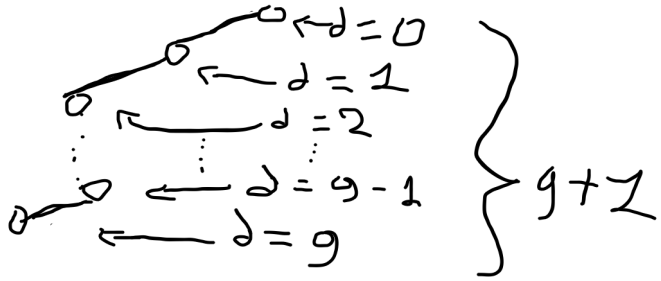
$\Rightarrow 3 + 2b + b^2$

Για $d = g - 1$, g φορές κόμβος ρίζας +



$\Rightarrow g + (g-1)b + (g-2)b^2 + ... + b^{(g-1)}$

Για $d = g$



$\Rightarrow g + 1 + g + (g-1)b + (g-2)b^2 + ... + b^{(g-1)} = 1 + 2g + (g-1)b + (g-2)b^2 + ... + b^{(g-1)}$

Άρα ο μικρότερος αριθμός κόμβων που μπορεί να δημιουργηθεί θα είναι:
$1 + 2g + (g-1)b + (g-2)b^2 + ... + b^{(g-1)}$

Για τον μεγαλύτερο αριθμό κόμβων η διαφορά θα είναι ότι στο g βάθος, θα αναζητήσουμε μέχρι:

$+ g + 1$ αναζητήσεις στον κόμβο ρίζα $\Rightarrow d + 1 + db + (d-1)b^2 + ... + b^d$

Άρα ο μεγαλύτερος αριθμός κόμβων που μπορεί να δημιουργηθεί θα είναι: $d+1+db+(d-1)b^2+...+b^d$

Πρόβλημα 3:

Για να είναι μία heuristic συνάρτηση συνεπής πρέπει $\forall n, n'$ να ισχύει $\mathrm{h(n)} \leq c(n, a, n') + h(n')$,όπου $n \xrightarrow{a} n'$
Εφαρμόζουμε την παραπάνω σχέση σε όλους τους κόμβους μας:

Για $n = mail \not\exists n'$
Για $n = ts, n' = mail, 23 \leq 6 + 26$

$$\text{Για } n = 103, n' = \begin{cases} ts & 21 \leq 8 + 23 \\ b3 & 21 \leq 4 + 17 \\ o109 & 21 \leq 12 + 24 \end{cases}$$

$$\text{Για } n = 109, n' = \begin{cases} 019 & 24 \leq 16 + 11 \\ 111 & 24 \leq 4 + 27 \end{cases}$$

Για $n = o111 \not\exists n'$

$$\text{Για } n = o119, n' = \begin{cases} 123 & 11 \leq 9 + 4 \\ storage & 11 \leq 7 + 12 \end{cases}$$

Για $n = o123, n' = \begin{cases} 125 & 4 \leq 4+16 \\ r123 & 4 \leq 4+0 \end{cases}$
Για $n = o125 \nexists n'$
Για $n = r123 \nexists n'$
Για $n = b1, n' = \begin{cases} c2 & 13 \leq 3+10 \\ b2 & 13 \leq 6+15 \end{cases}$
Για $n = b2, n' = b4, 15 \leq 3+18$
Για $n = b3, n' = \begin{cases} b1 & 17 \leq 4+13 \\ b4 & 17 \leq 7+18 \end{cases}$
Για $n = b4, n' = o109, 18 \leq 7+24$
Για $n = c1, n' = c3, 6 \leq 8+12$
Για $n = c2, n' = \begin{cases} c1 & 10 \leq 4+6 \\ c3 & 10 \leq 6+12 \end{cases}$
Για $n = c3 \nexists n'$
Για $n = storage \nexists n'$
Επομένως ισχύει η σχέση $\forall n, n'$
Άρα η h συνεπής, και παραδεκτή ως συνεπής.

β) Με $I(n_1, n_2, ..., n_n)$ συμβολίζουμε την ουρά/στίβα μας μετά από κάποια inputs
$O(n_1, n_2, ..., n_n)n_i$ συμβολίζουμε την ουρά/στίβα μας μετά από κάποια outputs και με n_i το output

Αναζήτηση πρώτα σε πλάτος

$I(o103), O()o103, I(b3, o109, ts), O(o109, ts)b3, I(o109, ts, b1, b4), O(ts, b1, b4)o109, I(ts, b1, b4, o111, o119),$
$O(b1, b4, o111, o119)ts, I(b1, b4, o111, o119, mail), O(b4, o111, o119, mail)b1, I(b4, o111, o119, mail, b2, c2),$
$O(o111, o119, mail, b2, c2)b4, O(o119, mail, b2, c2)o111, 0(mail, b2, c2)o119,$
$I(mail, b2, c2, o123, storage), O(b2, c2, o123, storage)mail, O(c2, o123, storage)b2, O(o123, storage)c2,$
$I(o123, storage, c1, c3), O(storage, c1, c3)o123, I(storage, c1, c3, o125, r123), O(c1, c3, o125, r123)storage,$
$O(c3, o125, r123)c1, O(o125, r123)c3, O(r123)o125, O()r123 \Rightarrow$Goal state

Επομένως η σειρά που βγαίνουν οι κόμβοι από το σύνορο:
$o103 \Rightarrow b3 \Rightarrow o109 \Rightarrow ts \Rightarrow b1 \Rightarrow b4 \Rightarrow o111 \Rightarrow 0119 \Rightarrow mail \Rightarrow b2 \Rightarrow c2 \Rightarrow o123 \Rightarrow storage \Rightarrow$
$c1 \Rightarrow c3 \Rightarrow o125 \Rightarrow r123$

Αναζήτηση πρώτα σε βάθος

$I(o103), O()o103, I(b3, o109, ts), O(b3, o109)ts, I(b3, o109, mail), O(b3, o109)mail, O(b3)o109, I(b3, o111, o119),$
$O(b3, o111)o119, I(b3, o111, o123, storage), O(b3, o111, o123)storage, O(b3, o111)o123, I(b3, o111, o125, r123),$
$O(b3, o111, o125), r123 \Rightarrow$Goal state

Επομένως η σειρά που βγαίνουν οι κόμβοι από το σύνορο:
$o103 \Rightarrow ts \Rightarrow mail \Rightarrow o109 \Rightarrow o119 \Rightarrow storage \Rightarrow o1123 \Rightarrow r124$

Αναζήτηση πρώτα σε βάθος με επαναληπτική εκβάθυνση

Για βάθος = 0:
$I(o103), O()103$

Για βάθος = 1:
$I(o103), O()103, I(b3, o109, ts), O(b3, o109)ts, O(b3)o109, O()b3$

Για βάθος = 2:
$I(o103), O()103, I(b3, o109, ts), O(b3, o109)ts, I(b3, o109, mail), O(b3, o109)mail, O(b3)o109, I(b3, o111, o119)$,
$O(b3, o111)o119, O(b3)o111, O()b3, I(b1, b4), O(b1)b4, O()b1$

Για βάθος = 3:
$I(o103), O()o103, I(b3, o109, ts), O(b3, o109)ts, I(b3, o109, mail), O(b3, o109)mail, O(b3)o109, I(b3, o123, storage)$,
$O(b3, o123)storage, O(b3)o123, O()b3, I(b1, b4), O(b1)b4, I(b1, o109), O(b1)o109, O()b1, I(b2, c2), O(b2)c2, O()b2$

Για βάθος = 4:
$I(o103), O()o103, I(b3, o109, ts), O(b3, o109)ts, I(b3, o109, mail), O(b3, o109)mail, O(b3)o109, I(b3, o111, o119)$,
$O(b3, o111)o119, I(b3, o111, o123, storage), O(b3, o111, o123)storage, O(b3, o111)o123, I(b3, o125, r123)$,
$O(b3, o125)r123 \Rightarrow$Goal state

Επομένως η σειρά που βγαίνουν οι κόμβοι από το σύνορο:
$o103 \Rightarrow o103 \Rightarrow ts \Rightarrow o109 \Rightarrow b3 \Rightarrow o103 \Rightarrow ts \Rightarrow mail \Rightarrow o109 \Rightarrow o119 \Rightarrow o111 \Rightarrow b3 \Rightarrow b4 \Rightarrow$
$b1 \Rightarrow o103 \Rightarrow ts \Rightarrow o109 \Rightarrow storage \Rightarrow o123 \Rightarrow b3 \Rightarrow b4 \Rightarrow o109 \Rightarrow b1 \Rightarrow c2 \Rightarrow b2 \Rightarrow o103 \Rightarrow ts \Rightarrow$
$mail \Rightarrow o109 \Rightarrow o119 \Rightarrow storage \Rightarrow o123 \Rightarrow r123$

Άπληστη αναζήτηση πρώτα στον καλύτερο με ευρετική συνάρτηση

Σε κάθε εξαγωγή αφαιρείται από την priority queue το στοιχείο με την μικρότερη heuristic value και σε περίπτωση ισότητας μικρότερο αλφαβητικά στοιχείο
$I(o103), O()o103, I(ts, b3, o109), O(ts, o109)b3, I(ts, o109, b1, b4), O(ts, o109, b4)b1, I(ts, o109, b4, c2, b2)$,
$O(ts, o109, b4, b2)c2, I(ts, o109, b4, b2, c1, c3), O(ts, o109, b4, b2, c3)c1, O(ts, o109, b4, b2)c3, O(ts, o109, b4)b2$,
$O(ts, o109)b4, O(o109)ts, I(o109, mail), O(mail)o109, I(mail, o111, o119), O(mail, o111)o119$,
$I(mail, o111, storage, o123), O(mail, o111, storage)o123, I(mail, o111, storage, o125, r123)$,
$O(mail, o111, storage, o125)r123 \Rightarrow$ Goal state

Επομένως η σειρά που βγαίνουν οι κόμβοι από το σύνορο:
$o103 \Rightarrow b3 \Rightarrow b1 \Rightarrow c2 \Rightarrow c1 \Rightarrow c3 \Rightarrow b2 \Rightarrow ts \Rightarrow o109 \Rightarrow o119 \Rightarrow o123 \Rightarrow r123$

A*

$n_i, i =$το κόστος από τον κόμβο $o103$ μέχρι τον n
Προσθέτουμε το i με το heuristic value του n και με βάση αυτό το αποτέλεσμα εξάγουμε από την ουρά το ελάχιστο, σε περίπτωση ισότητας πάμε αλφαβητικά

$I(o103), O()o103, I(b3_4, o109_{12}, ts8), O(o109_{12}, ts_8)b3_4, I(o109_{12}, ts_8, b1_8, b4_{11}), O(o109_{12}, ts_8, b4_{11})b1_8$,
$I(o109_{12}, ts_8, b4_{11}, c2_{11}, b2_{14}), O(o109_{12}, ts_8, b4_{11}, b2_{14})c2_{11}, I(o109_{12}, ts_8, b4_{11}, b2_{14}, c1_{15}, c3_{17})$,
$O(o109_{12}, ts_8, b4_{11}, b2_{14}, c3_{17})c1_{15}, O(o109_{12}, ts_8, b4_{11}, c3_{17})b2_{14}$,

$O(o109_{12}, ts_8, c3_{17})b4_{11}, O(o109_{12}, ts_8)c3_{17}, O(o109_{12})ts_8, O()o109_{12}, I(o111_{16}, o119_{28}), O(o111_{16})o119_{28}$, $I(o111_{16}, storage_{35}, o123_{37}), O(o111_{16}, storage_{35})o123_{37}, I(o111_{16}, storage_{35}, o125_{41}, r123_{41})$, $O(o111_{16}, storage_{35}, o125_{41})r123_{41} \Rightarrow$ Goal state

$o103 \Rightarrow b3 \Rightarrow b1 \Rightarrow c2 \Rightarrow c1 \Rightarrow b2 \Rightarrow b4 \Rightarrow c3 \Rightarrow ts \Rightarrow o109 \Rightarrow o119 \Rightarrow o123 \Rightarrow r123$

Πρόβλημα 4:

- State
  Για το ρομπότ:
  θέση ρομπότ: θ , κρατάει το πακέτο: $\kappa = \begin{cases} \text{p} & \text{κρατάει το πακέτο p} \\ 0 & \text{δεν κρατάει πακέτο} \end{cases}$
  Για τα πακέτα: disct { θέσεις ⇒ Πακέτα}

  $State(\theta, \kappa, disct)$

- Goal State
  Το ρομπότ είναι στη θέση mail , δεν κρατάει πακέτο και τα πακέτα όλα βρίσκοντε στον προορισμό τους, $disct_f =$ όλα τα πακέτα βρίσκοντε στον προορισμό τους

  $GoalState(mail, 0, disct_f)$

- Initial State
  Το ρομπότ είναι στη θέση mail , δεν κρατάει πακέτο και τα πακέτα όλα βρίσκοντε στην αρχική τους θέση, $disct_s =$ όλα τα πακέτα βρίσκοντε στην αρχική τους θέση

- Actions
  - Μετακινείται το ρομπότ σε μία γειτονική θέση
  - Πιάνει πακέτο
  - Αφήνει πακέτο

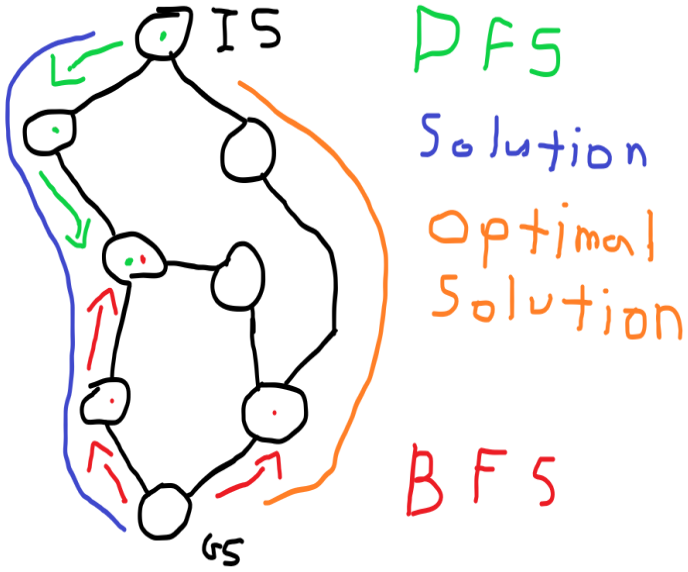
- Succesor function
  $\theta' =$ γειτονικός κόμβος κόμβου θ, Με τον όρο γειτονικός, εννούμε ότι υπάρχει μονοπάτι από τον θ στον θ'
  $P_S =$ Σύνολο όλων των πακέτων
  $Disct' = Disct[\theta'] \stackrel{+}{=} x, Disct[\theta] \stackrel{-}{=} x$, μετακίνηση πακέτου από τη θέση θ στη θέση θ'

If $\kappa == 0$ {
if $Disct[\theta] = empty$ then return $State(\theta', 0, Disct)$
else return { $State(\theta', 0, disct), State(\theta, x_i | i \in P_S, Disct)$ } }
Else $\{State(\theta', x, Disct'), State(\theta, 0, Disct)\}$)

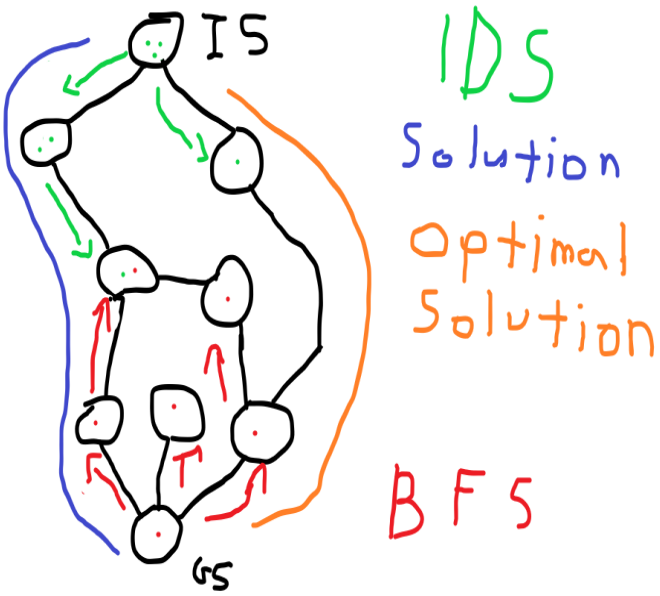
β) Για κάθε πακέτο το οποίο δεν είναι στη θέση του, βρίσκουμε την ελάχιστη απόστασή του από το σημείο προορισμού του, αθροίζουμε όλες αυτες τις αποστάσεις και επιστρέφουμε το αποτέλεσμα. Η συνάρτηση είναι παραδεκτή γιατί το κόστος ενός πακέτου για να φτάσει στον προορισμό του είναι τουλάχιστον η πιο σύντομη διαδρομή του για να πάει εκεί.

Πρόβλημα 5:

α) Ο αλγόριθμός μας είναι πλήρης με την προϋπόθεση ότι ο παράγωντας διακλάδωσης είναι πεπερασμένος, καθώς ο BLS είναι πλήρης δηλαδή σίγουρα θα επισκεφτεί τον κόμβο goal state εκτός αν επισκεφτεί νωρίτερα κάποιον κόμβο που ήδη έχει επισκεφτεί ο DLS . Δεν είναι βέλτιστος με αντιπαράδειγμα την παρακάτω εκτέλεση:



β) Ο αλγόριθμός μας είναι πλήρης με την προϋπόθεση ότι ο παράγωντας διακλάδωσης είναι πεπερασμένος, καθώς η αναζήτηση με επαναληπτική εκβάθηνση είναι πλήρης και θα επισκεφτεί σίγουρα τον κόμβο goal state εκτός αν νωρίτερα επισκεφτεί κάποιον κόμβο που ήδη έχει επισκεφτεί ο DLS . Δεν είναι βέλτιστος με αντιπαράδειγμα το παρακάτω:



γ) Ο αλγόριθμός μας είναι πλήρης με την προϋπόθεση ότι ο παράγωντας διακλάδωσης είναι πεπερασμένος, καθώς η αναζήτηση με επαναληπτική εκβάθηνση είναι πλήρης και θα επισκεφτεί σίγουρα τον κόμβο goal state εκτός αν νωρίτερα επισκεφτεί κάποιον κόμβο που ήδη έχει επισκεφτεί ο A* . Δεν είναι βέλτιστος καθώς και συνεπής heuristic function να έχουμε, ο A* μπορεί να βρει το goal state το οποίο να μην είναι από τη βέλτιστη διαδρομή, αλλά επειδή χρησιμοποιούμε αμφίδρομη αναζήτηση αυτό το μονοπάτι θα επιστραφεί.

δ) Ο αμφίδρομος διπλός A* είναι πλήρης καθώς ο απλός A* είναι πλήρης. Δεν είναι βέλτιστος γιατί μπορεί να συναντηθεί ο ένας A* με τον άλλο από μη βέλτιστο μονοπάτι αρχικά, και αυτό να επιστραφεί.

Ο κάθε αλγόριθμος έχει ένα visited set Το οποίο αποθηκεύει τους κόμβους που έχει επισκεφτεί ο αλγόριθμος. Μετά από κάθε επίσκεψη, ο κόμβος που επισκεύτεται ο ένας από τους 2 αλγορίθμους θα ελέγχεται αν υπάρχει μέσα στο set του άλλου. Αν το set είναι υλοποιημένο με δυαδικό δένδρο πχ η αναζήτηση θα γίνεται αποδοτικά με λογαριθμικό χρόνο