Project 1 - Τεχνητή Νοημοσύνη

Παναγιώτα Γύφτου , Α.Μ.: 1115201900318

Νοέμβριος 2022

Πρόβλημα 2

Ο μεγαλύτερος αριθμός κόμβων είναι:

$$1 + b + b^2 + b^3 + ... + b^d = \frac{b^{d+1} - 1}{b - 1}$$

ενώ ο μικρότερος αριθμός κόμβων είναι: d +1

Πιο αναλυτικά:
Ο αλγόριθμος Αναζήτηση Πρώτα κατά Βάθος (DFS), επεκτείνεται από τα αριστερά πρός τα δεξιά. Το μονοπάτι με το μικρρότερο πλήθος κόμβων θα είναι το συντομότερο, δηλαδή αυτό που ο στόχος θα είναι το αριστερότερο φύλλο του δέντρου.Αντίθετα το μονοπάτι με το μεγαλύτερο πλήθος κόμβων θα είναι αυτό που ο στόχος θα είναι το δεξιότερο φύλλο του δέντρου, δηλαδή σε αυτή την περίπτωση θα εξερευνηθούν όλοι οι κόμβοι του δέντρου αναζήτησης.

Πρόβλημα 3

α.
Μια ευρετική συνάρτηση λέγεται παραδεκτή αν δεν υπερκτιμά το πραγματικό κόστος του κόμβου, δηλαδή $h(v) \leq h*(v)$ για κάθε κόμβο ν του χώρου αναζήτησης.Μια ευρετική συνάρτηση λέγεται συνεπής εάν ισχύει: $h(n) \leq c(n,p) + h(p)$

Παρατηρούμε στον παρακάτω πίνακα ότι η συνάρτηση είναι παραδεκτή και συνεπής

με βάση τα παραπάνω.

Nodes	h	$h*$	$h \leq h*$
$o103$	21	$c(o103, ts) + h(ts)$	$21 \leq 8 + 23 = 31$
ts	23	$c(ts, mail) + h(mail)$	$23 \leq 6 + 26 = 32$
$o103$	21	$c(o103, b3) + h(b3)$	$214 + 17 = 21$
$b3$	17	$c(b3, b1) + h(b1)$	$17 \leq 4 + 13 = 17$
$b1$	13	$c(b1, c2) + h(c2)$	$13 \leq 3 + 10 = 13$
$c2$	10	$c(c2, c1) + h(c1)$	$10 \leq 4 + 6 = 10$
$c1$	6	$c(c1, c3) + h(c3)$	$6 \leq 8 + 12 = 20$
$c2$	10	$c(c2, c3) + h(c3)$	$10 \leq 6 + 12 = 18$
$b3$	17	$c(b3, b4) + h(b4)$	$17 \leq 7 + 18 = 25$
$b1$	13	$c(b1, b2) + h(b2)$	$13 \leq 6 + 15 = 21$
$b2$	15	$c(b2, b4) + h(b4)$	$15 \leq 3 + 18 = 21$
$b4$	18	$c(b4, o109) + h(o109)$	$18 \leq 7 + 24 = 31$
$o103$	21	$c(o103, o109) + h(o109)$	$21 \leq 12 + 24 = 36$
$o109$	24	$c(o109, o111) + h(o111)$	$24 \leq 4 + 27 = 31$
$o109$	24	$c(o103, o119) + h(o119)$	$24 \leq 16 + 11 = 27$
$o119$	11	$c(o119, storage) + h(storage)$	$11 \leq 7 + 12 = 19$
$o119$	11	$c(o119, o123) + h(o123)$	$11 \leq 9 + 4 = 13$
$o123$	4	$c(o123, o125) + h(o125)$	$4 \leq 4 + 6 = 10$
$o123$	4	$c(o123, r123) + h(r123)$	$4 \leq 4 + 0 = 4$

β.

- *Αναζήτηση πρώτα σε πλάτος (BFS)*

Η σειρά με την οποία εξάγωνται τα στοιχεία (δηλ. τα κωδικοποιημένα ονόματα των δωματίων), είναι η εξής:

o103, b3, o109, ts, b1, b4, o111, o119, mail, b2, c2, o123, storage, c1, c3, o125, r123

- *Αναζήτηση πρώτα σε βάθος (DFS)*

Η σειρά με την οποία εξάγωνται τα στοιχεία, είναι η εξής:

o103, ts, mail, o109, o119, storage, o123, r123

- *Αναζήτηση πρώτα σε βάθος με επαναληπτική εκβάθυνση (IDS)*

Η σειρά με την οποία εξάγωνται τα στοιχεία (δηλ. τα κωδικοποιημένα ονόματα των δωματίων), είναι η εξής:

Depth	*pop*
0	$o103$
1	$o103, b3, o109, ts$
2	$o103, b3, b1, b4, o109, o111, o119, ts, mail$
3	$o103, b3, b1, b2, c2, b4, o109, o109, o111, o119, o123, storage, ts, mail$
4	$o103, b3, b1, b2, b4, c2, c1, c3, b4, o109, o111, o109, o111, o119, o123, o125, r123$

- *Άπληστη αναζήτηση πρώτα στον καλύτερο (Greedy)*

Η σειρά με την οποία εξάγωνται τα στοιχεία (δηλ. τα κωδικοποιημένα ονόματα των δωματίων), είναι η εξής:

o103, b3, b1, c2, c1, c3, b2, b4, ts, o109, o119, o123, r123

- *A-σταρ*

Η σειρά με την οποία εξάγωνται τα στοιχεία (δηλ. τα κωδικοποιημένα ονόματα των δωματίων), είναι η εξής:

o103, b3, b1, c2, c1, b2, b4, c3, ts, o109, o119, mail, o123, r123

Πρόβλημα 4

(α)

◦ **Καταστάσεις** (*States*)

- Χώρος Καταστάσεων (*State Space*):
  State = (τρέχουσα θέση ρομπότ, κατάλογος ελέγχου παράδοσης πακέτων, σημαφόρος χρήσης βραχίονα)

• *τρέχουσα θέση ρομπότ*: οι συντεταγμένες του τρέχοντα κόμβου που βρίσκεται το ρομπότ

• *κατάλογος ελέγχου παράδοσης πακέτων*: (κωδικό όνομα πακέτου, συντεταγμένες σημείου παραλαβής, συντεταγμένες σημείου παράδοσης, σημαφόρος διανομής: σε εξέλιξη / αδράνεια, σημαφόρος παράδοσης: παραδοτέο / μη παραδοτέο)

• *σημαφόρος χρήσης βραχίονα*: μεταφέρει πακέτο/ δεν μεταφέρει πακέτο

◦ **Αρχική Κατάσταση** (*Start State*)

- Μαθηματικός τρόπος :

$$StartState = \left(mail, \left[\begin{array}{l} package1: \ (\ (x_{rcpt1}, y_{rcpt1}),\ \ (x_{del1}, y_{del1}),\ \ 0,\ \ 0\ ) \\ package2: \ (\ (x_{rcpt2}, y_{rcpt2}),\ \ (x_{del2}, y_{del2}),\ \ 0,\ \ 0\ ) \\ \qquad\qquad\qquad\qquad ... \\ packageN: (\ (x_{rcptN}, y_{rcptN}),\ (x_{delN}, y_{delN}),\ 0,\ \ 0\ ) \end{array} \right],\ 0 \right)$$

- Ανάλυση :

$StartState = ($

τρέχουσα θέση ρομπότ: $mail$,

κατάλογος ελέγχου παράδοσης πακέτων: ο κατάλογος περιέχει όλα τα ονόματα και τις πληροφορίες των πακέτων, με τους σημαφόρους διανομής και παράδοσης να είναι στην κατάσταση σε αδράνεια και μη παραδοτέο αντίστοιχα,

σημαφόρος χρήσης βραχίονα: δε μεταφέρει πακέτο $)$

◦ **Τελική Κατάσταση** (*Goal State*)

- Μαθηματικός τρόπος :

$$GoalState = \left(mail, \left[\begin{array}{l} package1: \ (\ (x_{rcpt1}, y_{rcpt1}),\ \ (x_{del1}, y_{del1}),\ \ 0,\ \ 1\ ) \\ package2: \ (\ (x_{rcpt2}, y_{rcpt2}),\ \ (x_{del2}, y_{del2}),\ \ 0,\ \ 1\ ) \\ \qquad\qquad\qquad ... \\ packageN: (\ (x_{rcptN}, y_{rcptN}),\ (x_{delN}, y_{delN}),\ 0,\ \ 1\ ) \end{array} \right],\ 0 \right)$$

- Ανάλυση :

$GoalState = ($

τρέχουσα θέση ρομπότ: $mail$,

κατάλογος ελέγχου παράδοσης πακέτων: ο κατάλογος περιέχει όλα τα ονόματα και τις πληροφορίες των πακέτων, με τους σημαφόρους διανομής και παράδοσης να είναι στην κατάσταση σε αδράνεια και παραδοτέο αντίστοιχα,

σημαφόρος χρήσης βραχίονα: δε μεταφέρει πακέτο $)$

◦ **Συνάρτηση Διαδόχου** (*Successor Function*)

Η συνάρτηση διαδόχου δουλειά της είναι να ανανεώνει, αν χρειάζεται, τον κατάλογο

ελέγχου παράδοσης πακέτων (δηλαδή αλλάζει τους σημαφόρους του πακέτου της τρέχουσας κατάστασης) και να επιστρέφει μια λίστα από έγκυρες επόμενες καταστάσεις, με την αντίστοιχη ενέργεια κίνησης τους.Πιο συγκεκριμένα στο πρόβλημα μας θα επιστραφεί μια λίστα από επόμενα προσβάσιμα δωμάτια που μπορεί να επισκεφτεί το ρομπότ, συνοδευόμενα με την πληροφορία οδηγίας μετακίνησης του ρομπότ, δηλ. το ρομποτ για να πάει (από το o109) στο o119 δωμάτιο πρέπει να πάει βόρεια. Οι επιτρεπτές κινήσεις του ρομπότ είναι (βόρεια, νότια, δυτικά, ανατολικά)

(β)

Χρειαζόμαστε μια συνάρτηση που να μην υπερεκτιμά τον αριθμό των βημάτων για τον στόχο.

Από την υπόθεση γνωρίζουμε ότι το ρομπότ μπορεί να μεταφέρει ένα πακέτο τη φορά, αυτό σημαίνει ότι από την στιγμή που θα πραγματοποιηθεί η παραλαβή, πρέπει το δέμα να παραδοθεί στον προορισμό του.Το κόστος διαδρομής για μια παραλαβή και παράδοση πακέτου περιλαμβάνει δύο κόστη διαδρομών.Το πρώτο είναι η διαδρομή που κάνει το ρομπότ από το σημείο που βρίσκεται μέχρι να παραλάβει ένα πακέτο και το δεύτερο είναι η διαδρομή που κάνει το ρομπότ από το σημείο παραλαβής μέχρι το σημείο παράδοσης.

Η ιδέα είναι η εξης:
Με τη βοήθεια της μεθόδου $Manhattan$ να υπολογιστούν όλες οι αποστάσεις **(τρέχουσα θεση ρομπότ- σημείο παραλαβής) + (σημείο παραλαβής-σημείο παράδοσης)** και να επιλεχθεί η πιο βέλτιστη διαδρομή.Η επιλογή του υπολογισμού απόστασης με μέθοδο $Manhattan$, έγινε με το κριτήριο ότι επειδή δεν δίνεται η δυνατότητα μετακίνησης του ρομποτ διαγώνια, η απόσταση που θα υπολογίζεται να είναι το άθροισμα των ενεργειών (μία ενέργεια τη φορά).Τέλος αφού έχουν αθροιστεί τα κόστη τρέχονΘεση-παράδοσης-παραλαβής όλων των μεταφερόμενων πακέτων, προσθέτουμε και το κόστος διαδρομής που διασχίζει το ρομπότ ώστε να επιστρέψει στην αρχική του θέση, δηλαδή την θέση του δωματίου $mail$.Στην περίπτωση που βρίσκεται σε αυτήν, το κόστος για την επιστροφή είναι 0.

Η παραπάνω ευρετική είναι ένας παραδεκτός ευρετικός μηχανισμός, επειδή κάθε κίνηση του ρομπότ το φέρνει σε μία κατάσταση πιο κοντά στον στόχο.Κάθε βήμα που κάνει το ρομπότ δεν είναι επιλεγμένη αυθαίρετα.Η απόσταση $Manhattan$ επιλέγει διαδρομές μη εξερευνημένες και με κατεύθυνση προς τον στόχο, και όχι σε οποιαδήποτε κατεύθυνση.

Πρόβλημα 5

(α) ***Αναζήτηση πρώτα σε πλάτος*** & ***Αναζήτηση περιορισμένου βάθους***

• *Αποτίμηση BFS*

Πληρότητα: Ο BFS είναι πλήρης, εάν ο παράγοντας διακλάδωσης (δηλαδή το πλήθος των νέων καταστάσεων-διαδόχων) είναι πεπερασμένος.

Βέλτιστη Συμπεριφορά: Ο BFS είναι βέλτιστος, εάν όλες οι ενέργειες που χρησιμοποιούνται στον κώδικα έχουν το ίδιο κόστος και ο κόμβος στόχου επιτευχθεί εγκαίρως.

• *Αποτίμηση DSL*

Πληρότητα: Ο DLS είναι πλήρης, εάν ισχύει η περίπτωση ο κόμβος στόχου να μη βρίσκεται πέρα από το όριο βάθους $d \leq l$.Διαφορετικά εάν $l < d$, δηλαδή ο κόμβος στόχου είναι σε βαθύτερο σημείο από το δεσμευμένο σύνολο βάθους το DLS θα επιστρέψει $False$ ακόμα και αν υπάρχει λύση.

Βέλτιστη Συμπεριφορά: Ο DSL δεν είναι βέλτιστος, διότι το όριο βάθους l που μπορεί να επιλεγεί μπορεί να είναι μεγαλύτερο από το $d(l > d)$, και έτσι υπάρχει περίπτωση σημαντικής αύξησης του κόστους αναζήτησης.

Συμπεραίνουμε ότι ο αλγόριθμός αμφίδρομης αναζήτησης είναι πλήρης, διότι η συνθήκη ότι τουλάχιστον ένας από τους δύο αλγόριθμους αναζήτησης είναι πλήρης είναι αληθής σε αυτόν το συνδυασμό, καθώς και οι δύο αλγόριθμοι είναι πλήρης όπως γνωρίζουμε από την θεωρία (οι λόγοι αναπτύχθηκαν πιο πάνω).Ωστόσο αυτός ο αλγόριθμός αμφίδρομης αναζήτησης δεν είναι βέλτιστος διότι ο DLS δεν είναι βέλτιστος,παρόλο που ο BFS είναι.

(β) ***Αναζήτηση με επαναληπτική εκβάθυνση*** & ***Αναζήτηση περιορισμένου βάθους***

• *Αποτίμηση IDS*

Πληρότητα: Ο IDS είναι πλήρης, υπό τις προϋποθέσεις του BFS.

Βέλτιστη Συμπεριφορά: Ο IDS είναι βέλτιστος, υπό τις προϋποθέσεις του BFS.

• *Αποτίμηση DSL*

Πληρότητα και Βέλτιστη Συμπεριφορά: Έχουν αναλυθεί στο (α)

Συμπεραίνουμε ότι ο αλγόριθμός αμφίδρομης αναζήτησης είναι πλήρης, διότι η συνθήκη ότι τουλάχιστον ένας από τους δύο αλγόριθμους αναζήτησης είναι πλήρης είναι αληθής σε αυτόν το συνδυασμό, καθώς και οι δύο αλγόριθμοι είναι πλήρης όπως γνωρίζουμε από την θεωρία (οι λόγοι αναπτύχθηκαν πιο πάνω).Ωστόσο αυτός ο αλγόριθμός αμφίδρομης αναζήτησης δεν είναι βέλτιστος διότι ο DLS δεν είναι βέλτιστος,παρόλο που ο IDS είναι.

(γ) A^* & ***Αναζήτηση περιορισμένου βάθους***

• *Αποτίμηση* A^*

Πληρότητα: Ο A^* είναι πλήρης

Βέλτιστη Συμπεριφορά: Ο A^* είναι βέλτιστος

* Ανάλυση διαφάνειες $https://cgi.di.uoa.gr/\ ys02/dialekseis2020/heuristics1spp.pdf\ \ p.19, 20, 21, 22$

• *Αποτίμηση DSL*

Πληρότητα και Βέλτιστη Συμπεριφορά: Έχουν αναλυθεί στο (α)

Συμπεραίνουμε ότι ο αλγόριθμός αμφίδρομης αναζήτησης είναι πλήρης, διότι η συνθήκη ότι τουλάχιστον ένας από τους δύο αλγόριθμους αναζήτησης είναι πλήρης είναι αληθής σε αυτόν το συνδυασμό, καθώς και οι δύο αλγόριθμοι είναι πλήρης όπως γνωρίζουμε από την θεωρία (οι λόγοι αναπτύχθηκαν πιο πάνω).Ωστόσο αυτός ο αλγόριθμός αμφίδρομης αναζήτησης δεν είναι βέλτιστος διότι ο DLS δεν είναι βέλτιστος,παρόλο που ο A^* είναι.

(δ) A^* & A^*

• *Αποτίμηση* A^*

Πληρότητα και Βέλτιστη Συμπεριφορά: Έχουν αναλυθεί στο (γ)

Συμπεραίνουμε ότι ο αλγόριθμός αμφίδρομης αναζήτησης είναι πλήρης, διότι η συνθήκη ότι τουλάχιστον ένας από τους δύο αλγόριθμους αναζήτησης είναι πλήρης είναι αληθής σε αυτόν το συνδυασμό, καθώς και οι δύο αλγόριθμοι είναι πλήρης όπως γνωρίζουμε από την θεωρία (οι λόγοι αναπτύχθηκαν πιο πάνω).Τέλος αυτός ο αλγόριθμός αμφίδρομης αναζήτησης είναι και βέλτιστος διότι και οι δύο αλγόριθμοι είναι βέλτιστοι