email at: sdi2000055@di.uoa.gr

Περιεχομενα

1. Ασκηση 2

Για τον **μέγιστο αριθμό κόμβων**: Η περίπτωση κατά την οποία ο αλγόριθμος θα δημιουργήσει τους περισσότερους κόμβους είναι στην περίπτωση κατά την οποία ο κόμβος στόχος βρίσκεται δεξιότερα στο τελευταίο επίπεδο g. Ο αλγόριθμος ξεκινάει από τον πρώτο κόμβο στην κορυφή, ο οποίος συνολικά θα εξερευνηθεί $g+1$ φορές (μια φορά για κάθε επανάληψη του αλγορίθμου). Αφού έπειτα έχουμε παράγωντα διακλάσωδης b, οι κόμβοι στο επίπεδο 1 θα είναι b, και θα εξερευνηθούν g φορές (για όλες τις επαναλήψεις του αλγορίθμου εκτός της πρώτης), κ.ο.κ. Φτάνοντας έτσι στο τελευταίο επίπεδο που όλοι οι κόμβοι θα εξευρενηθούν από 1 φορά μέχρι να βρεθεί ο κόμβος στόχος στο δεξιότερο μέρος. Άρα
(πόσες φορές εξερευνούνται) $\cdot$ (κόμβοι) =

$$(g+1)\cdot 1 + gb + (g-1)b^2 + \cdots + 1\cdot b^g = \sum_{d=0}^{g}(g-d+1)b^d$$

Για τον **μικρότερο αριθμό κόμβων**: Ο αλγόριθμος λόγω της λειτουργίας του θα έχει ελέγξει όλους τους κόμβους στο επίπεδο $(g-1)$, δηλαδή στο επίπεδο πάνω από το επίπεδο στο οποίο βρίσκεται ο στόχος. Ο αριθμός αυτών των κόμβων δίνεται από τον παραπάνω τύπο που βρήκαμε, αν θέσουμε ως g το $g-1$. Επομένως, μέχρι το επίπεδο $g-1$ θα έχουνε εξερευνηθεί: $g+(g-1)b+\cdot+2b^{g-2}+b^{g-1} = \sum_{d=0}^{g-1}(g-d)b^d$
Όταν φτάσουμε στο τελευταίο επίπεδο, θέλουμε τον κόμβο να είναι πρώτος, δηλαδή αριστερότερα. Ξεκινώντας από τον κόμβο ρίζας, κάθε φορά αναλύουμε έναν κόμβο ανά επίπεδο, άρα συνολικά για τα g επίπεδα θα δημιουργήσουμε b κόμβους, άρα $\underbrace{b+b+\cdots+b}_{g\text{ φορές}} = gb$ κόμβους , χωρίς να υπολογίζουμε την ρίζα. Άρα στην τελική επανάληψη έχουμε $gb+1$ κόμβους. Επομένως **συνολικά** ο ελάχιστος αριθμός κόμβων που μπορούν να δημιουργηθούν είναι:

$$\sum_{d=0}^{g-1}(g-d)\cdot b^d + (gb+1)$$

2. Ασκηση 3

α)
Αρχικά, θα δείξω ότι η παραπάνω συνάρτηση είναι συνεπής.
Για να είναι **συνεπής**, θα πρέπει να ισχύει: Για όλους τους κόμβους n, n' που είναι τέτοιοι ώστε ο n' να είναι απόγονος του n που παράγεται από μια ενέργεια a έχουμε $h(n) \leq c(n, a, n') + h(n')$

Συνέπεια			
$h(n)$	$h(n')$	$c(n, a, n')$	$h(n) \leq c(n, a, n') + h(n')$
$h(o103)$	$h(b3)$	4	$21 \leq 4 + 17$
$h(o103)$	$h(ts)$	8	$21 \leq 8 + 23$
$h(o103)$	$h(o109)$	12	$21 \leq 12 + 24$
$h(ts)$	$h(mail)$	6	$23 \leq 6 + 26$
$h(o109)$	$h(o111)$	4	$24 \leq 4 + 27$
$h(o109)$	$h(o119)$	16	$24 \leq 16 + 11$
$h(b3)$	$h(b4)$	7	$17 \leq 7 + 18$
$h(b3)$	$h(b1)$	4	$17 \leq 4 + 13$
$h(b1)$	$h(b2)$	6	$13 \leq 6 + 15$
$h(b2)$	$h(b4)$	3	$15 \leq 3 + 18$
$h(b1)$	$h(c2)$	3	$13 \leq 3 + 10$
$h(c2)$	$h(c3)$	6	$10 \leq 6 + 12$
$h(c2)$	$h(c1)$	4	$10 \leq 4 + 6$
$h(c1)$	$h(c3)$	8	$6 \leq 8 + 12$
$h(o119)$	$h(storage)$	7	$11 \leq 7 + 12$
$h(o119)$	$h(o123)$	9	$11 \leq 9 + 4$
$h(o123)$	$h(r123)$	4	$4 \leq 4 + 0$
$h(o123)$	$h(o125)$	4	$4 \leq 4 + 6$

Άρα αφού ισχύει για κάθε κόμβο και τους απογόνους του, είναι συνεπής. Άρα είναι και παραδεκτή.
β)
· **Αναζήτηση κατά πλάτος**

Η σειρά εξόδου από το σύνορο είναι:

$o103, b3, o109, ts, b1, b4, o111, o119, mail, b2, c2, o123$

Υποσημείωση: Ακολούθησα τον αλγόριθμο που βρίσκεται στις σημειώσεις εδώ , ο οποίος ελέγχει για τον κόμβο goal καθώς κοιτάζει τους κόμβους παιδιά του κόμβου που γίνεται expanded.

· **Αναζήτηση πρώτα κατά βάθος**

Η σειρά εξόδου από το σύνορο είναι:

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

Υποσημείωση: Ακολούθησα τον αλγόριθμο που βρίσκεται στις σημειώσεις εδώ. Οι κόμβοι μπαίνουν στο σύνορο (στοίβα) με αλφαβητική σειρά, άρα βγαίνουν αντίθετα.

· **Αναζήτηση πρώτα σε βάθος με επαναληπτική εκβάθυνση**

Για $d = 0$ η σειρά εξόδου είναι:
$o103$

Για $d = 1$ η σειρά εξόδου είναι:
$o103, ts, o109, b3$

Για $d = 2$ η σειρά εξόδου είναι:
$o103, ts, mail, o109, o119, o111, b3, b4, b1$

Για $d = 3$ η σειρά εξόδου είναι:
$o103, ts, mail, o109, o119, storage, o123, o111, b3, b4, o109, b1, b2$

Για $d = 4$ η σειρά εξόδου είναι:
$o103, ts, mail, o109, o119, storage, o123, r123$

Υποσημείωση: Ομοίως με τον DFS οι κόμβοι μπαίνουν με αλφαβητική σειρά εντός της στοίβας και για αυτό βγαίνουν αντίθετα.

· **Άπλειστη αναζήτηση πρώτα στον καλύτερο με ευρετική** h

Η σειρά εξόδου είναι:
$o103, b3, b1, c2, c1, c3, b2, b4, ts, o109, o11, o123, r123$

· **A***

Η σειρά εξόδου είναι:
$o103, b3, b1, c2, c1, b2, b4, c3, ts, o109, o119, mail, o123, r123$

3. Ασκηση 4

α)

Initial State

Όπως αναφέρεται στην εκφώνηση το ρομπότ ξεκινάει από το δωμάτιο του μαιλ χωρίς να κουβαλάει κάποιο πακέτο καθώς επίσης χωρίς να έχει παραδώσει κάποιο πακέτο ακόμα, και με μία αρχική λίστα η οποία δέχεται μέσα tuples της μορφής (αποστολέας, παραλήπτης, πακέτο)

initPosition = mail
packetBeingCarried = NULL
packets = [(sender, receiver, packet)]

Goal State

Θέλουμε το ρομπότ να επιστρέφει στην αρχική θεση χωρίς να μεταφέρει πακέτο και να μην έχει κάποιο πακέτο προς παράδωση, δηλαδή η λίστα παραδοτέων πακέτων να είναι άδεια.

finalPosition = mail
packetBeingCarried = NULL

packetsDelivered = []

Συνάρτηση διαδοχής / Successor Function
Η συνάρτηση διαδοχής επιστρέφει μια λίστα από πιθανές επόμενες κινήσεις της μορφής: (nextState, action) με το nextState να περιέχει πληροφορία σχετικά με την νέα θέση που βρίσκεται το ρομπότ, την επικαιρωμένη λίστα σχετικά με την κατάσταση των πακέτων και πληροφορία για το πακέτο που μεταφέρεται τώρα (αν μεταφέρεται), ενώ το action πρόκειται για την κίνηση που επιδέχεται το ρομπότ στις συντεταγμένες του. Οι κινήσεις αυτές πρέπει φυσικά να είναι δεκτές (δηλαδή μια κίνηση που οδηγεί το ρομπότ πάνω σε τοίχο να μην γίνεται δεκτή κ.ο.κ.)

Cost Function
Όρισα το κάθε βήμα, που όπως είπα πριν να είναι δεκτό και ένα των πάνω, κατώ, δεξία αριστερά, να έχει κόστος 1. Άρα
cost(state, action, nextState) = 1, nextState $\in SuccessorFunction(state).nextState$
β)
Για την ευρυστική αρχικά θα πρέπει να χρησιμοποιήσουμε μια συνάρτηση υπολογισμού απόστασης. Αν υποθέσουμε ότι επιστρέπονται μόνο οι 4 παραπάνω κατευθύνσεις, και όχι κάποιος συνδυασμός τους (π.χ. διαγώνιος), τότε μπορούμε να θεωρήσουμε ότι η συνάρτηση απόστασης είναι απλά μια συνάρτηση Μανχάταν Απόστασης. Αν ωστόσο επιτρέπονται και συνδυασμοί, για παράδειγμα έχουμε και διαγώνιες κατευθύνσεις τότε θα πρέπει να λάβουμε μία καλύτερη ευρεστική. Αυτή μπορεί να είναι η διαγώνια απόσταση ή ακόμα και η ευκλείδια απόσταση (αυτή μπορεί να χρησιμοποιηθεί για οποιαδήποτε ωστόσο περίπτωση). Σε κάθε περίπτωση θέλουμε να λάβουμε μια απόφαση η οποία δεν υπερεκτιμά την απόσταση.
Μια ευρεστική συνάρτηση, σχετικά απλή, είναι:
Το άθροισμα των αποστάσεων μεταξύ των πακέτων που πρέπει να παραδωθούν και της απόστασης του κοντινότερου δωματίου από το mail.
Δηλαδή:
cost = 0
for packet in packetsToBeDelivered:
    cost = cost + distance(sender,receiver)
    minDist = min{ distance(mail, sender), minDist}

Η ευρετική αυτή είναι σίγουρα παραδεκτή, αφού δεν υπερκτιμά ποτέ την απόσταση που θα πρέπει να διανύσει το ρομπότ, αφού την συνολική απόσταση μεταξύ πακέτων σίγουρα θα πρέπει να την διανύσει, καθώς και σίγουρα θα διανύσει ίση ή μεγαλύτερη απόσταση για να βγει από το mail για να πάει στο πρώτο άλλο δωμάτιο.

4. Ασκηση 5

· (α) Αναζήτηση πρώτα σε πλάτος και αναζήτηση περιορισμένου βάθους

Για την πληρότητα της αναζήτησης πρέπει:

i) Ο παράγοντας διακλάδωσης να είναι πεπερασμένος, ως προϋπόθεση για την πληρότητα της αναζήτησης πρώτα σε πλάτος.

ii) Το όριο βάθους, ℓ, να είναι μεγαλύτερο ή ίσο από το βάθος λύσης d, ως προϋπόθεση πληρότητας της αναζήτησης περιορισμένου βάθους ($d \leq \ell$)
Προφανώς ο αλγόριθμος δεν είναι βέλτιστος. Αυτό συμβαίνει λόγω της αναποτελεσματικότητας της αναζήτησης περιορισμένου βάθους η οποία ανεξαρτήτως του επιπέδου στόχου, θα παράγει πολύ περισσότερους κόμβους από το βέλτιστο.

· (β) Αναζήτηση με επαναληπτική εκβάθυνση και αναζήτηση περιορισμένου βάθους

Για την πληρότητα της αναζήτησης πρέπει:

i) Ο παράγοντας διακλάδωσης να είναι πεπερασμένος, ως προϋπόθεση για την πληρότητα της αναζήτησης πρώτα σε βάθος, στην οποία βασίζεται η αναζήτηση με επαναληπτική εκβάθυνση.

ii) Το όριο βάθους, ℓ, να είναι μεγαλύτερο ή ίσο από το βάθος λύσης d, ως προϋπόθεση πληρότητας της αναζήτησης περιορισμένου βάθους ($d \leq \ell$)

Προφανώς ο αλγόριθμος δεν είναι βέλτιστος. Ομοίως με τον (α), η αναζήτηση περιορισμένου βάθους καθιστά τον αλγόριθμο μη βέλτιστο, λόγω της δικιάς της μη βέλτιστης φύσης.

· (γ) A* και αναζήτηση περιορισμένου βάθους

Για την πληρότητα της αναζήτησης πρέπει:

i) Η ευρετική συνάρτηση να μην υπερεκτιμά το κόστος που έχουμε για να φτάσουμε στο στόχο (παραδεκτή), ο παράγοντας διακλάδωσης να είναι πεπερασμένος, κάθε ενέργεια να κοστίζει τουλάχιστον $\delta > 0$.

ii) Το όριο βάθους, ℓ, να είναι μεγαλύτερο ή ίσο από το βάθος λύσης d, ως προϋπόθεση πληρότητας της αναζήτησης περιορισμένου βάθους ($d \leq \ell$)

Ομοίως με τα (α), και (β), ο αλγόριθμος δεν είναι βέλτιστος, λόγω της χρήσης της μη βέλτιστης αναζήτησης περιορισμένου βάθους.

· (δ) A* και A*

Για την πληρότητα της αναζήτησης πρέπει:

i) Η ευρετική συνάρτηση να μην υπερεκτιμά το κόστος που έχουμε για να φτάσουμε στο στόχο (παραδεκτή), ο παράγοντας διακλάδωσης να είναι πεπερασμένος, κάθε ενέργεια να κοστίζει τουλάχιστον $\delta > 0$. (αποκλειστικά δηλαδή οι προϋποθέσεις για να είναι ο A* βέλτιστος)

Για να είναι ο αλγόριθμος βέλτιστος χρειάζεται η ευρετική συνάρτηση να είναι συνεπής.

Για τον έλεγχο συνάντησης των αλγορίθμων

· Για τον (α) μπορούμε να χρησιμοποιήσουμε ένα σύνολο για να κρατάει τους εξερευνημένους κόμβους και να ελέγχουμε αν αυτοί συνδέονται (αν υπάρχουν δηλαδή) με τον συνολο εξερευνημένων κόμβων της αναζήτησης κατά πλάτος.
· Για τον (β), καθώς κανένας από τους δυο υποαλγορίθμουςδεν κρατάει τους εξερευνημένους κόμβους, μπορεί να υλοποιηθεί ένα σύνολο που να κρατάει τους κόμβους (του τελευταίου κάθε φορά επιπέδου) που η αναζήτηση με επαναληπτική εκβάθυνση εξερευνεί. Τελικώς τώρα είμαστε σε θέση να ελέγχουμε αν οι κόμβοι που παράγει η αναζήτηση περιορισμένου βάθους ανήκουν στο σύνολο εξερευνημένων της επαναληπτικής εκβάθυνσης, άρα να ελέγξουμε αν υπάρχει συνάντηση.
· Για τον (γ) θα μπορούσαμε ξανά να κρατάμε ένα σύνολο εξερευνημένων κόμβων για την αναζήτηση πειρορισμένου βάθους, την οποία να εκμεταλλεύεται ο A* για να ελέγχει αν οι απόγονοι ανήκουν στο σύνολο.
· Για τον (δ), αφού ο A* κρατάει σύνολο εξερευνημένων κόμβων, αρκεί να ελέγξουμε αν υπάρχει ένα κοινός κόμβος στα δύο σύνολα εξερευνημένων κόμβων των A*.