Read Me Για Τον Κώδικα Του Pac man

Άγγελος Τσιτσόλη sdi2000200

Νοέμβριος 13, 2022

Ερώτημα 2

Στο ερώτημα αυτό χρησιμοποιείται ο αλγόριθμος πρώτα σε βάθος επαναληπτικής εκβάθυνσης άρα θα χρησιμοποιείται ο γνωστός αλγόριθμος IDS.

Ο αλγόριθμος IDS ουσιαστικά αυτό που θα κάνει είναι ξεκινώντας με βάθος ίσο με το μηδέν ως όριο να εκτελεί τον αλγόριθμο DLS και κάθε φορά που τον εκτελεί να αυξάνει το βάθος κατά ένα .Συγκεκριμένα κάθε φορά που θα εκτελεί τον DLS δεν θα διατηρεί αποθηκευμένους κάπου τους κόμβους που εξερεύνησε πριν(την προηγούμενη φορά που εκτελέσθηκε δηλαδή ο DLS) , αλλά αντιθέτως ο αλγόριθμος DLS θα γίνεται συνεχώς απο την αρχή δημιουργώντας δηλαδή ένα δέντρο προσθέτωντας νέους κόμβους κάθε φορά λόγω του αυξημένου κατά ένα βάθους μέχρι να βρεθεί η κατάσταση στόχου ή μέχρι να μην μπορεί να προχωρήσει λόγω του ορίου του βάθους.

Ως χειρότερη θεωρείται η περίπτωση όπου για να βρεθεί η κατάσταση στόχου θα πρέπει να ελεγχθούν όλες οι πιθανές κατάστασεις .Σε ένα δέντρο καταστάσεων η χειρότερη περίπτωση θα ήταν η κατάσταση στόχου να βρίσκοταν σε πολύ μεγάλο βάθος και να χρειαζόταν να ελέγξουμε όλους του κόμβους του δέντρου μέχρι να φτάσουμε στον τελευταίο που θα ήταν η κατάσταση στόχου .Επίσης εφόσον ο παράγοντας διακλάδωσης είναι ίσος με b τότε η ρίζα θα είναι $b^0 = 1$ εφόσον είναι μοναδική στο επίπεδο 0 ,ο αριθμός των παιδιών της ρίζας θα είναι b^1 στο επίπεδο 1 , επομένως, τα παιδιά των παιδιών της ρίζας θα είναι b^2 κ.τ.λ για τα επόμενα επίπεδα . Στην χειρότερη περίπτωση όσον αφορά τον αριθμό των κόμβων προκύπτουν τα εξής παρακάτω:

1. Όπως αναφέρθηκε και πιο πάνω στον αλγόριθμο IDS θα εκτελείται ο DLS και θα δημιουργείται κάθε φορά ένα δέντρο , μέχρι να φτάσει το όριο βάθους που του επιτρέπεται ή σε κατάσταση στόχου.Αρχικά στ επίπεδο 0 έχουμε μόνο ένα κόμβο την ρίζα, άρα ο παράγοντας διακλάδωσης θα είναι $b^0 = 1$. Ξεκινώντας για βάθος d=0 έχουμε ότι θα δημιουργηθεί μόνο η ρίζα , οπότε κρατάμε την περίπτωση αυτή . Στην συνέχεια για βάθος d=1 και έπειτα για μεγαλύτερα βάθη θα δημιουργείται ομοίως πάντοτε η ρίζα , άρα εφόσον η κατάσταση στόχου βρίσκεται σε βάθος d η ρίζα θα δημιουργηθεί d φορές μέχρι να βρεθούμε στην κατάσταση στόχου, συν μία φορά όπου δημιουργέιται για την περίπτωση όπου d=0. Οπότε συνολικά έχουμε (d+1)·1 φορές για τη ρίζα.

2. Εφόσον θα χρειαστεί να φτάσουμε μέχρι το βάθος d προκειμένου να βρούμε την κατάσταση στόχου θα χρειαστεί να δημιουργήσουμε τα παιδιά της ρίζας d φορές . Στο επίπεδο αυτό έχουμε b^1 κόμβους άρα συνολικά θα δημιουργηθούν οι κόμβοι αυτοί d·b φορές.

3. Απο τα παιδιά των παιδιών της ρίζας και έπειτα βρισκόμαστε κάθε φορά ένα βήμα πιο κοντά στο βάθος d οπότε θα δημιθουργούμε κόμβους κάθε φορά και λιγότερες φορές διότι βρισκόμαστε πιο κοντά στον να βρούμε τον στόχο.Επομένως τα παιδιά των παιδιών της ρίζας θα δημιουργηθούν d-1 φορές καθώς το επίπεδο τους βρισκεται πιο κοντά στο βάθος d κατά ένα επίπεδο σε σχέση με τους αμέσως προηγούμενους κόμβους.Επίσης στο επίπεδο αυτό έχουμε b^2 κόμβους άρα συνολικά θα δημιουργηθούν (d-1)·b^2 φορές οι κόμβοι αυτοί.

4. Έπειτα τα παιδιά των προηγούμενων κόμβων θα βρίσκονται και αυτά κατά ένα επίπεδο πιο κοντά στο βάθος d άρα με την σειρά τους θα εμφανιστούν d-2 φορές και για τους επόμενους κόμβους θα ισχύει ότι θα δημιουργηθούν d-2,d-3,d-4 φορές κ.τ.λ.π.Όσο θα μειώνεται το βάθος διότι φτάνουμε

πιο κοντά στην κατάσταση στόχου θα αυξάνεται ο αριθμός των κόμβων ανά επίπεδο μέχρι το βάθος που βρίσκεται η κατάσταση στόχου $b^3, b^4....b^d$.

5. Οι κόμβοι που βρίσκονται στον προτελευταίο επίπεδο πριν φτάσουμε στο βάθος d θα δημιουργηθούν 2 φορές και συνολικά στο επίπεδο αυτό θα έχουμε b^{d-1} κόμβους, δηλαδή μία φορά που θα φτάσει μέχρι εκεί ο αλγόριθμος IDS και άλλη μια φορά όπου θα φτάσουμε στο επόμενο επίπεδο στον τελικό στόχο άρα συνολικά θα δημιουργηθούν $2 \cdot b^{d-1}$ φορές.

6. Οι κόμβοι που βρίσκονται σε βάθος d θα δημιουργηθούν μόνο μία φορά , ενώ στο επίπεδο εκείνο θα έχουμε b^d κόμβους άρα συνολικά θα δημιουργηθούν $1 \cdot b^d$ φορές..Εκεί θα τελειώσει η διαδικασία εύρεσης του τελικού στόχου.

Εν τέλει προσθέτουμε αυτές τις τιμές και προκύπτει πως στην χειρότερη περίπτωση ο αριθμός των κόμβων θα είναι ο εξής: $(\mathrm{d}+1) + d \cdot b + (d-1) \cdot b^2 + ... + 2 \cdot b^{d-1} + 1 \cdot b^d$

Στην καλύτερη περίπτωση ουσιαστικά βρίσκουμε με την πρώτη προσπάθεια τον κόμβο στόχου , δηλαδή ο πρώτος κόμβος που εξετάζουμε θα είναι και ο κόμβος στόχου , επομένως προκύπτει το εξής : Εφόσον το μικρότερο βάθος που μπορεί να είναι ένας κόμβος είναι g τότε για d=g προκύπτει number of nodes=$(\mathrm{g}+1) + g \cdot b + (g-1) \cdot b^2 + (g-2) \cdot b^3 + ... + 2 \cdot b^{g-1} + 1 \cdot b^g$ και καθώς κοιτάμε ένα μόνο κόμβο δηλαδή φτάνουμε μέχρι το επίπεδο μηδέν g=0 και στο επίπεδο μηδέν ο παράγοντας διακλάδωσης είναι ίσος με 1 άρα number of nodes=$(0+1) + 0 \cdot 1 + (0-1) \cdot 1^2 + (0-2) \cdot b^3 + ... + 2 \cdot 1^{0-1} + 1 \cdot 1^0$ <=> number of nodes=$1 + 0 + (-1) + (-2) + ...2 + 1 = 1$

Ερώτημα 3

Αποδεικνύουμε ότι η συνάρτηση αυτή είναι συνεπής , επομένως αυτόματα αποδεικνύεται ότι η συνάρτηση είναι παραδεκτή :

Αρχικά ισχύει ότι h(r123)=0.

Έπειτα αποδεικνύουμε για κάθε κόμβο , για κάθε γείτονα του ότι ισχύει η σχέση $\mathrm{h(n)} \leq c(n, n_1) + h(n_1)$ όπου n ένας κόμβος και n ο γείτονας του . Με λίγα λόγια θα αποδείξουμε έτσι ότι απο την τριγωνική ανισότητα που προκύπτει απο αυτή τη σχέση κάθε πλευρά του τριγώνου είναι μικρότερη απο το άθροισμα των άλλων δύο.

1. $\mathrm{h(o103)} \leq c(o103, o109) + h(o109)$ <=> $21 \leq 12 + 24$ <=> $21 \leq 36$
2. $\mathrm{h(o103)} \leq c(o103, b3) + h(b3)$ <=> $21 \leq 4 + 17$ <=> $21 \leq 21$
3. $\mathrm{h(o103)} \leq c(o103, ts) + h(ts)$ <=> $21 \leq 8 + 23$ <=> $21 \leq 31$
4. $\mathrm{h(o109)} \leq c(o109, o119) + h(o119)$ <=> $24 \leq 16 + 11$ <=> $24 \leq 27$
5. $\mathrm{h(o109)} \leq c(o109, o111) + h(o111)$ <=> $24 \leq 4 + 27$ <=> $24 \leq 31$
6. $\mathrm{h(o119)} \leq c(o119, storage) + h(storage)$ <=> $11 \leq 7 + 12$ <=> $11 \leq 19$
7. $\mathrm{h(o119)} \leq c(o119, o123) + h(o123)$ <=> $11 \leq 9 + 4$ <=> $11 \leq 13$
8. $\mathrm{h(o123)} \leq c(o123, r123) + h(r123)$ <=> $4 \leq 4 + 0$ <=> $4 \leq 4$
9. $\mathrm{h(o123)} \leq c(o123, o125) + h(o125)$ <=> $4 \leq 4 + 6$ <=> $4 \leq 10$
10. $\mathrm{h(b3)} \leq c(b3, b4) + h(b4)$ <=> $17 \leq 7 + 18$ <=> $17 \leq 25$
11. $\mathrm{h(b3)} \leq c(b3, b1) + h(b1)$ <=> $17 \leq 4 + 13$ <=> $17 \leq 17$
12. $\mathrm{h(b1)} \leq c(b1, b2) + h(b2)$ <=> $13 \leq 6 + 15$ <=> $13 \leq 21$
13. $\mathrm{h(b1)} \leq c(b1, c2) + h(c2)$ <=> $13 \leq 3 + 10$ <=> $13 \leq 13$
14. $\mathrm{h(c2)} \leq c(c2, c3) + h(c3)$ <=> $10 \leq 6 + 12$ <=> $10 \leq 18$
15. $\mathrm{h(c2)} \leq c(c2, c1) + h(c1)$ <=> $10 \leq 4 + 6$ <=> $10 \leq 10$

16. h(b2)$\leq c(b2,b4)+h(b4)$ <=> 15$\leq 3+18$ <=> 15 ≤ 21
17. h(c1)$\leq c(c1,c3)+h(c3)$ <=> 6$\leq 8+12$ <=> 6 ≤ 20
18. h(b4)$\leq c(b4,o109)+h(o109)$ <=> 18$\leq 7+24$ <=> 18 ≤ 31

Breadth First Search :
Χρησιμοποιείται ο αλγόριθμος Graph Search, με fringe να αποτελεί μια ουρά (Queue).
Η σειρά με την οποία βγαίνουν οι κόμβοι απο την λίστα σύνορο είναι η εξής :
o103,b3,o109,ts,b1,b4,o111,o119,mail,b2,c2,o109,o123,storage,b4,c1,c3,o125,r123

Depth First Search : Χρησιμοποιείται ο αλγόριθμος Graph Search, με fringe να αποτελεί μια στοίβα (Stack).
Η σειρά με την οποία βγαίνουν οι κόμβοι απο την λίστα σύνορο είναι η εξής :
o103,ts,mail,o109,o119,storage,o123,r123

Iterative Deepening Search: Χρησιμοποιείται ο αλγόριθμος Iterative Deepening Search , με το fringe κάθε φορά να είναι μια στοίβα.
d=0 o103
d=1 o103,ts,o109,b3
d=2 o103,ts,mail,o109,o119,o111,b3,b4,b1
d=3 o103,ts,mail,o109,o119,storage,o123,o111,b3,b4,o109,b1,c2,b2
d=4 o103,ts,mail,o109,o119,storage,o123,r123

Best First Search:
o103,b3,b1,c2,c1,c3,c3,b2,b4,b4,ts,mail,o109,o119,o123,r123

A* με ευρετική:
o103,b3,b1,c2,c1,b2,b4,c3,ts,c3,b4,o109,o119,mail,o123,r123

Ερώτημα 4

α)

- Καταστάσεις: (State,Holds-packages,Destination,Number of packages that have to be delivered) x=0,1 (το 0 συμβολίζει ότι δεν κουβαλάει πακέτο ενώ το 1 συμβολίζει ότι κουβαλάει πακέτο)
- Αρχική κατάσταση: (mail,x,None,N) (αρχικά ξεκινάει απο το mail χωρίς να έχει κάποιο προορισμό και N είναι ο αριθμός όλων των πακέτων που πρέπει να μεταφερθούν)
- Ενέργειες: i,j=mail,ts,o103... (state$_i$,1,state$_j$,N)=¿αυτή η κατάσταση σημαίνει ότι παρέλαβε ένα πακέτο.Επόμενη κατάσταση να φτάσει στον τελικό προορισμό δηλαδή το State και το Destination να έχουν ίδια τιμή οπότε : (state,0,state,N-1) . Δηλαδή όταν φτάσει στον προορισμό του κρατώντας ένα πακέτο τότε θα μειωθεί ο αριθμός των πακέτων που πρέπει να παραδώσει.Σε περίπτωση που δεν κουβαλάει κάποιο πακέτο θα έχει τη μορφή (state$_i$,0,state$_j$,N).
- Κατάσταση Στόχου:(mail,x,None,0) (τελικά φτάνει στο mail πάλι έχοντας παραδώσει όλα τα πακέτα)

β)
Για το πρόβλημα αυτό προκύπτει μια βέλτιστη διαδρομή (που θα έχει το μικρότερο κόστος) , την οποία μπορεί να ακολουθήσει το ρομπότ ώστε να φτάσει στον τελικό στόχο με τον καλύτερο τρόπο τον πιο 'κερδοφόρο'.Το σκεπτικό είναι το εξής (όπως και σχεδόν σε κάθε ευρετική): απο την κατάσταση που βρισκόμαστε πιο είναι το κόστος της διαδρομής που μένει μέχρι την τελική κατάσταση (να έχουν παραδωθεί δηλαδή όλα τα πακέτα εκεί που πρέπει και να έχει επιστρέψει στην θέση που ξεκόνησε το ρομπότ);Γιαυτό λοιπόν η ευρετική μας θα μας δώσει την επιπλέον πληροφορία προκειμένου να διαθέτει μια κατεύθυνση ο αλγόριθμος που θα χρησιμοποιεί την ευρετική.Η ευρετική για κάθε κατάσταση θα πρέπει να βγάζει ένα κόστος που να είναι όσο το δυνατό πιο κοντά στο μέγιστο (αλλά να είναι πάντοτε μικρότερο ή ίσο , για να είναι παραδεκτή η ευρετική)ώστε , ο αλγόριθμος να εξερευνεί όσο το δυνατό λιγότερους κόμβους. Για κάθε κατάσταση η ευρετική θα βρίσκει το συνολικό κόστος που προκύπτει, απο τα κόστη

των διαδρομών που θα πρέπει να κάνει το ρομπότ (απο την στιγμή που παραλάβει το πακέτο) για κάθε πακέτο προκειμένου να το παραδώσει συν το κόστος για την απόσταση απο το κοντινότερο πακέτο όταν τελειώνει μια παράδοση . Ουσιαστικά η ευρετική μπορεί εύκολα να επιστρέψει για μια κατάσταση το κόστος διαδρομής που μπορεί να ακολουθήσει το ρομπότ για να φτάσει στην τελική κατάσταση , ωστόσο δεν μπορούμε να γνωρίζουμε απο τις τόσες διαδρομές που έχει στην επιλογή, του ποια διαδρομή θα διαλέξει, που σημαίνει ότι το κόστος της διαδρομής που θα διαλέξει υπάρχει μεγάλη πιθανότητα να είναι μεγαλύτερο απο το πραγματικό κόστος άρα να μην είναι πλεον παραδεκτή η ευρετική.Επομένως προσθέτουμε στο συνολικό κόστος της ευρετικής και το κόστος της διαδρομής που κάνει το ρομπότ (κάθε φορά που τελειώνει μια παράδοση) για να πάει στο πιο κοντινό πακέτο , μαυτό τον τρόπο γνωρίζουμε ότι θα συνεχίσει την πορεία του το ρομπότ όπως πρέπει και δεν θα δοκιμάζει καινούργιες διαδρομές χρονοβόρες όπως και δαπανηρές σε σχέση με το πραγματικό κόστος της λύσης.Άρα με τον τρόπο αυτό το ρομπότ θα διανύει μια διαδρομή που θα προσεγγίζει κατά το δυνατόν την βέλτιστη λύση .Προκειμένου να θεωρείται παραδεκτή η ευρετική θα πρέπει να ισχύει ότι δεν υπερεκτιμά το κόστος για να φτάσουμε στον τελικό κόστος , δηλαδή ο στόχος μας είναι να παραδσωθούν όλα τα πακέτα και να επιστρέψει στο ίδιο σημείο απο όπου ξεκίνησε.Εφόσον η βέλτιστη λύση του προβλήματος είναι το ρομπότ να παραδόσει τα πακέτα κάνοντας το μικρότερο δυνατό δρόμο δηλδή προφανώς παραδίδοντας κάθε φορά το κοντινότερο τότε είναι προφανές ότι η ευρετική που βρήκαμε καθώς κάθε φορά μετράει το κόστος για τη διαδρομή που κάνει για να παραδοθεί ένα πακέτο και στην συνέχεια μετράει το κόστος για το κοντινότερο τότε θα είναι παραδεκτή διότι δεν ξεπερνάς το κόστος της βέλτιστης λύσης. Συγκεκριμένα δεν μπορούμε να ξέρουμε ότι θα είναι ίση με την βέλτιστη λύση (γιατί είναι πολύ πιαθνό να περνάει και απο δωμάτια που δεν χρειάζεται) αλλά σίγουρα θα την προσεγγίζει καθώς προσπαθεί να την μιμηθεί σε ένα βαθμό.

Η τελική κατάσταση του προβλήματος αυτού είναι το ρομπότ να έχει παραδώσει όλα τα πακέτα και να έχει επιστρέψει στο σημείο που ξεκίνησε . Επομένως μια ευρετική που θα μπορούσε να χρησιμοποιηθεί απο τον αλγόριθμο A* , ξεκινώντας απο ένα οποιοδήποτε σημείο (για παράδειγμα το mail, όπως δίνεται) , είναι η εξής : Η ευρετική συνάρτηση θα αποτελεί το άθροισμα των κόστων των απόστασεων απο το πιο κοντινο πακέτο προς παράδοση κάθε φορά , επίσης των κόστων των διαδρομών που θα διανύσει το ρομπότ κουβαλώντας ένα πακέτο κάθε φορά μέχρι να το αφήσει στον προορισμό του και της επιστροφής στο αρχικό σημείο που ξεκίνησε.

Ουσιαστικά η ευρετική θα υπολογίζει το κόστος διαδρομής του πλησιέστερου πακέτου κάθε φορά ώστε να κατευθυνθεί προς αυτό και στην συνέχεια θα υπολογίζει το κόστος της διαδρομής που θα διανύσει το ρομπότ μεταφέροντας το πακέτο . Αυτή η διαδικασία θα γίνει μέχρι να μεταφερθούν όλα τα πακέτα και στο στο τέλος θα συμπεριληφθεί και το κόστος μέχρι την επιστροφή στο αρχικό σημείο.

όσον αφορά την παραδεκτότητα της ευρετικής , η ευρετική θα είναι σίγουρα ίση με το κόστος της βέλτιστης διαδρομής που μπορεί να κάνει το ρομπότ.

Ερώτημα 5

Πληρότητα

1)

α)

BFS: Για πεπερασμένο παράγοντα διακλάδωσης b είναι πλήρης βρίσκει πάντοτε δηλαδή τη λύση .

DLS: Είναι πλήρης όταν το βάθος του δεν είναι άπειρο όταν δηλαδή υπάρχει ένα όριο στο βάθος ώστε να

σταματήσει . Για να είναι πλήρης δηλαδή έχει ένα $l \geq d$ όπου l είναι το όριο βάθος και d το βάθος της λύσης.

IDS: Είναι πλήρης υπο τις προυποθέσεις του BFS , δηλαδή για πεπρασμένο παράγοντα διακλάδωσης θα βρίσκει πάντοτε λύση.

A*:Είναι πλήρης εφόσον ο παράγοντας διακλάδωσης είναι πεπερασμένος . Θα βρίσκει πάντοτε τη λύση στο πρόβλημα .

Βελτιστότητα
BFS: Είναι βέλτιστος αλγόριθμος με την προυπόθεση ότι όλες οι ενέργειες έχουν το ίδιο κόστος , διότι σε περίπτωση που έχουν άλλο διαφορετικό κόστος εφόσον η bfs δεν διαλέγει με βάση το κόστος αλλά διαλέγει κάθε φορά τους κόστους του επιπέδου τότε είναι πολύ πιθανό να έχει διαλέξει μια λύση η οποία να μην είναι 'φθηνότερη' να είναι δηλαδή σπάταλη όσον αφορά το κόστος των ενεργειών.

DLS: Δεν είναι βέλτιστος καθώς όπως και ο DFS (βασίζεται στον dfs όπου εξ ορισμού δεν είναι βέλτιστος) .

IDS: Είναι βέλτιστος υπό τις προυποθέσεις του BFS δηλαδή όταν όλες οι ενέργειες έχουν το ίδιο κόστος θα βρίσκει πάντοτε τη βέλτιστη λύση.

A*:Ο A* είναι βέλτιστος όταν ο γράφος στον οποίο εκτελείται είναι πεπερασμένος δεν είναι άπειρος και σύμφωνα με τις διαφάνειες του μαθήματος έχει δύο επιλογές για να εξασφαλιστεί η βέλτιστη συμπεριφορά του 1) να επεκταθεί ο αλγόριθμος Graph Search για να απορρίπτει την πιο δαπανηρή απο δύο διαδρομές που βρίσκει προς τον ίδιο κόμβο.(Αυτό γίνεται σε περίπτωση που ο A* χρησιμοποιεί τον αλγόριθμο Graph Search ως κύρια υπορουτίνα διότι υπάρχουν περιπτώσεις κατά τις οποίες ο Graph Search θα απορρίψει την βέλτιστη διαδρομή). 2)Η ευρετική που χρησιμοποιεί ο A* να είναι συνεπής.

Σύμφωνα με τα παραπάνω στην ερώτηση," είναι ο αλγόριθμος αμφίδρομης αναζήτησης με υπορουτίνες όπως στα (α)-(δ) πλήρης;" . Έχουμε (συμπεριλεμβάνοντας για το καθένα και τις συνθήκες για τις οποίες μπορεί να είναι πλήρης οι οποίες αναφέρθηκαν παραπάνω για το κάθενα αλγόριθμο ξεχωριστά) :

α)Ο BFS είναι πλήρης αλγόριθμος για πεπερασμένο παράγοντα διακλάδωσης και ο DLS είναι πλήρης για πεπερασμένο βάθος. Άρα είναι πλήρης ο αλγόριθμος αμφίδρομης αναζήτησης υπο τις συνθήκες που αναφέραμε.

β)Ο IDS είναι πλήρης για πεπερασμένο παράγοντα διακλάδωσης και ο DLS πλήρης για πεπερασμένο βάθος.Άρα θα είναι πλήρης ο αλγόριθμος αμφίδρομης αναζήτησης υπο τις συνθήκες αυτές.

γ) Ο A* είναι πλήρης για πεπερασμένο παράγοντα διακλάδωσης και ο DLS , είναι πλήρης για πεπερασμένο βάθος . Άρα υπο τις συνθήκες αυτές είναι πλήρης ο αλγόριθμος αμφίδρομης αναζήτησης .

δ)Ο A* είναι πλήρης για πεπρασμένο παράγοντα διακλάδωσης , άρα είναι πλήρης ο αλγόριθμος αμφίδρομης αναζήτησης υπο τις συνθήκες αυτές.

Στην ερώτηση "Είναι βέλτιστος;" . Οπότε σύμφωνα με τα παραπάνω και τις συνθήκες για τις οποίες είναι ή όχι βέλτιστοι οι αλγόριθμοι έχουμε :

α)Ο BFS είναι βέλτιστος , ο DLS δεν είναι βέλτιστος. Άρα εν τέλει δεν είναι βέλτιστος ο αλγόριθμος αμφίδρομης αναζήτησης .

β)Ο IDS είναι βέλτιστος αλλά ο DLS δεν είναι βέλτιστος.Άρα εν τέλει δεν είναι βέλτιστος ο αλγόριθμος αμφίδρομης αναζήτησης .

γ)Ο A* είναι βέλτιστος για τις παραπάνω προυποθέσεις αλλά ο DLS δεν είναι βέλτιστος οπότε δεν θα είναι βέλτιστος ο αλγόριθμος αμφίδρομης αναζήτησης.

δ)Ο A* είναι βέλτιστος για τις παραπάνω προυποθέσεις , άρα είναι βέλτιστος ο αλγόριθμος αμφίδρομης

αναζήτησης.

β) Ουσιαστικά για έναν αποδοτικό έλεγχο θα μπορούσαμε να πούμε ότι για κάθε περίπτωση που έχουμε δύο βέλτιστους αλγορίθμους τότε οι αλγόριθμοι θα συναντηθούν σε κάποιο σημείο της βέλτιστης διαδρομής . Ουσιαστικά εφόσον και οι δύο είναι βέλτιστοι θα ακολουθήσουν τον βέλτιστο μονοπάτι και κάποια στιγμή όπως είναι προφανές θα συναντηθούν κατά την εκτέλεση τους . Σε περίπτωση που δεν είναι βέλτιστοι ή κάποιος απο αυτούς δεν είναι βέλτιστος δεν μπορούμε να είμαστε με τίποτα σίγουρα αν θα συναντηθούν μπορεί τυχαία να συναντηθούν το πιο πιθανό είναι μάλλον όχι.Φυσικά υπο τις συνθήκες που είπαμε παραπάνω προκειμένου να είναι βέλτιστοι οι αλγόριθμοι έχουμε τα εξής :(Έστω δηλαδή ότι ισχύουν όλες οι προυποθέσεις παραπάνω που χρειάζονται για να είναι πλήρης και βέλτιστοι κάποιοι απο τους παραπάνω)
α)Θα συναντηθούν
β)Δεν θα συναντηθούν
γ)Δεν θα συναντηθούν
δ)Θα συναντηθούν
(Προφανω΄ς όταν λέμε ότι δεν θα συναντηθούν εννοούμε ότι μπορεί και να συναντηθούν απλώς η πιθανότηταείναι πολύ μικρή)