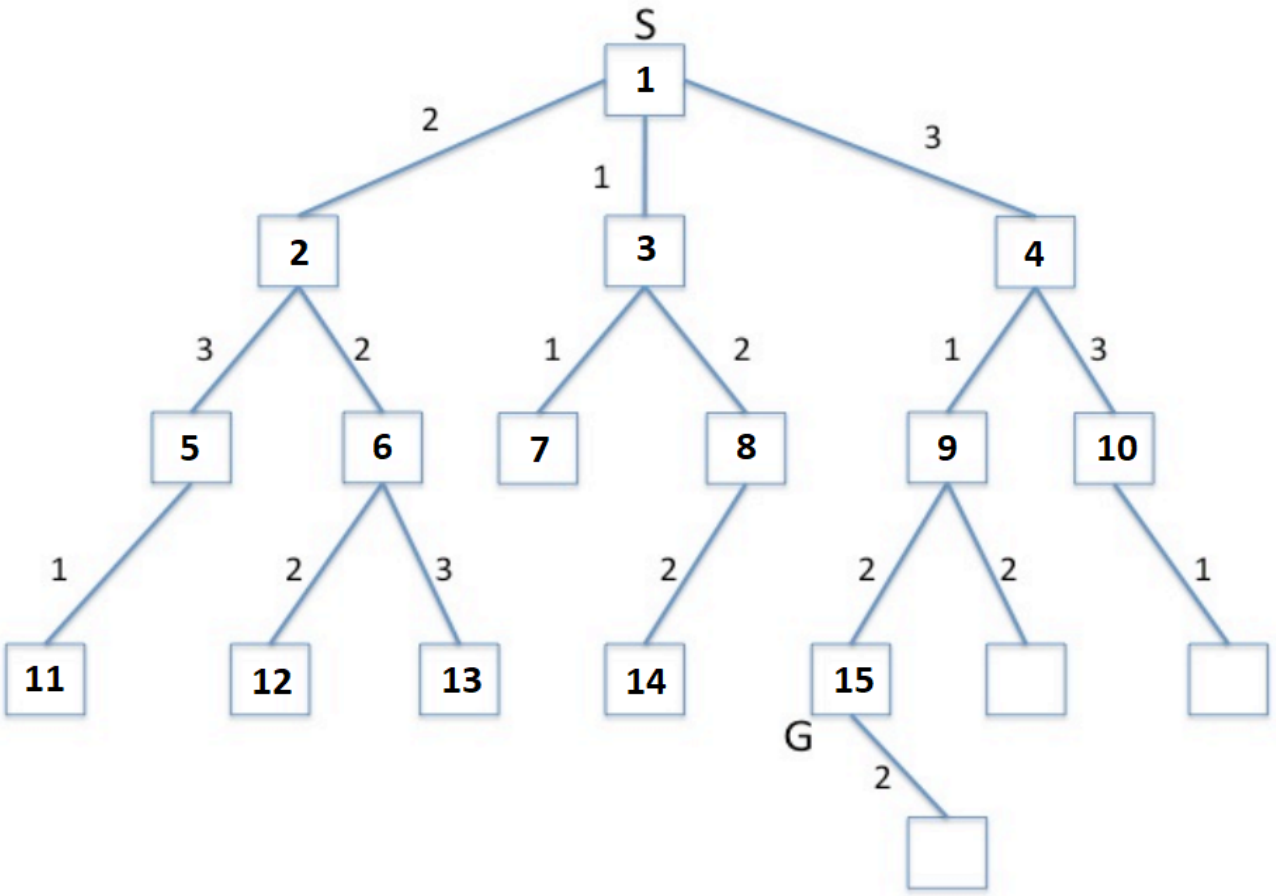
ΠΛΗ417 - Τεχνητή Νοημοσύνη
1η Σειρά Θεωρητικών Ασκήσεων

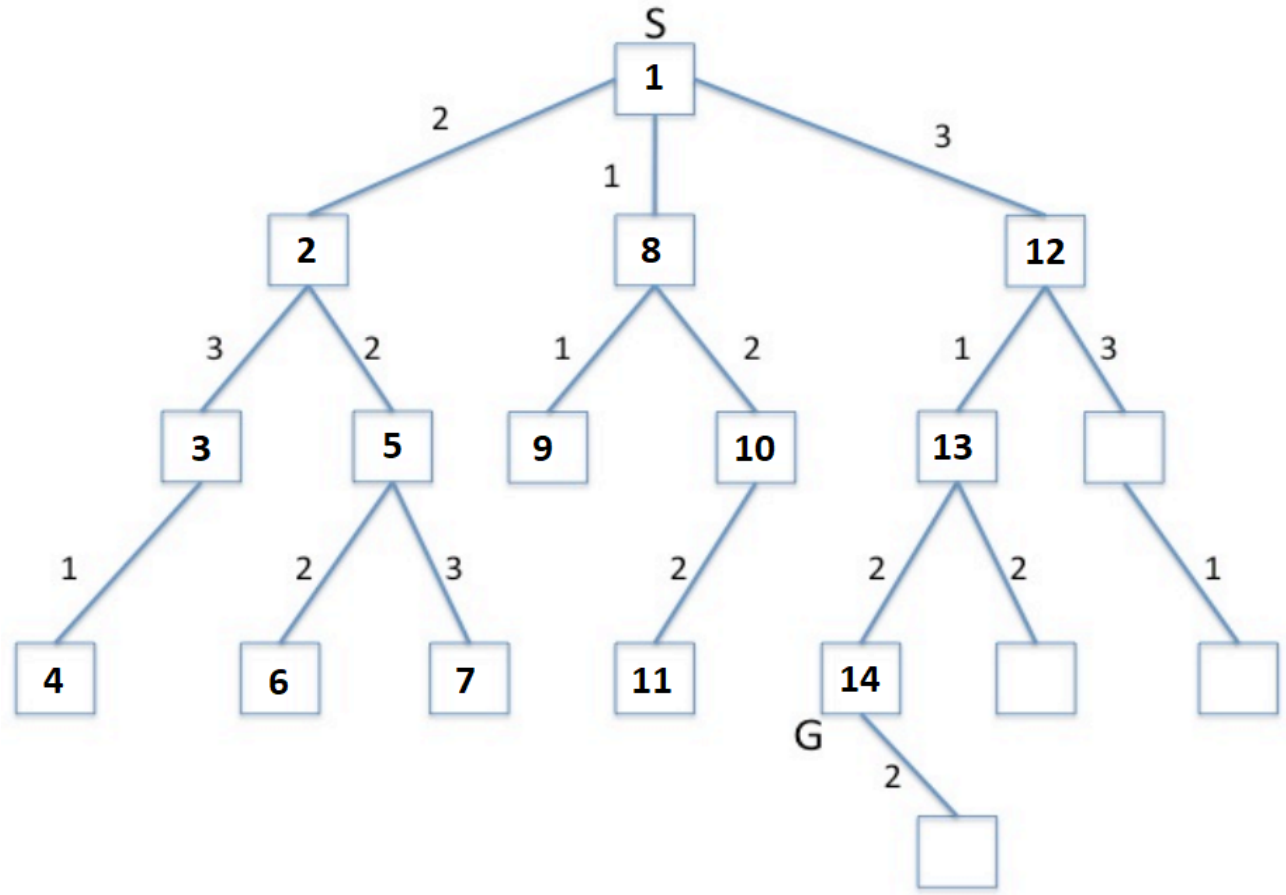
Μανώλης Πετράκος
ΑΜ 2014030009

Άσκηση 1)

a) Breadth-First Search

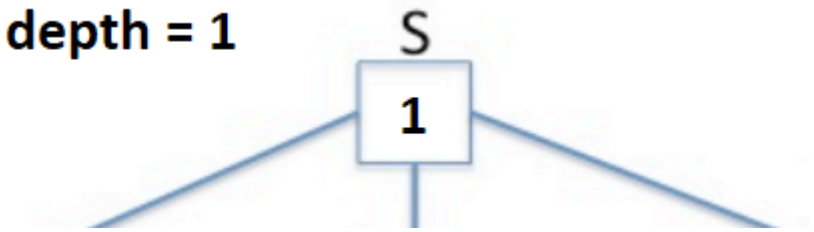


b) Depth-First Search

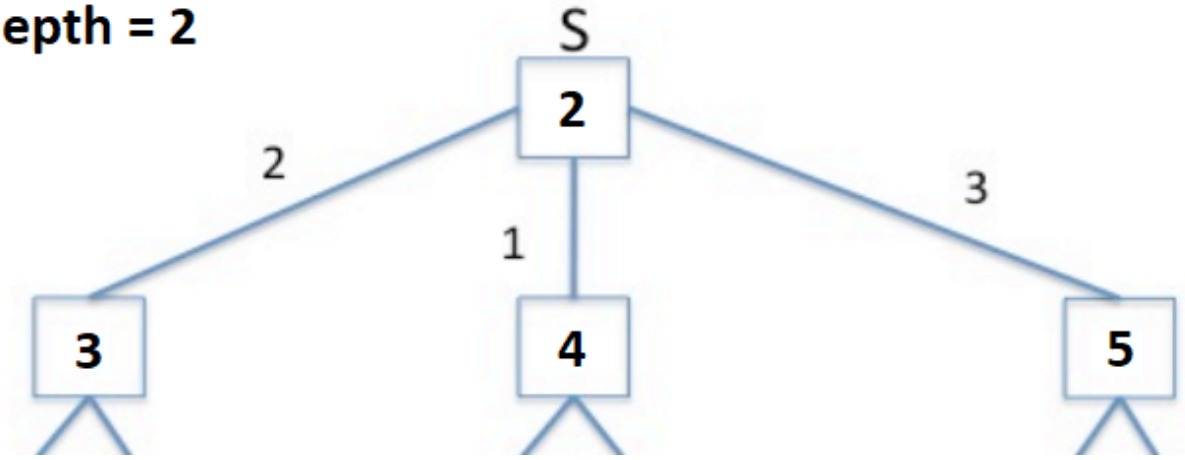


c) Iterative Deepening Search

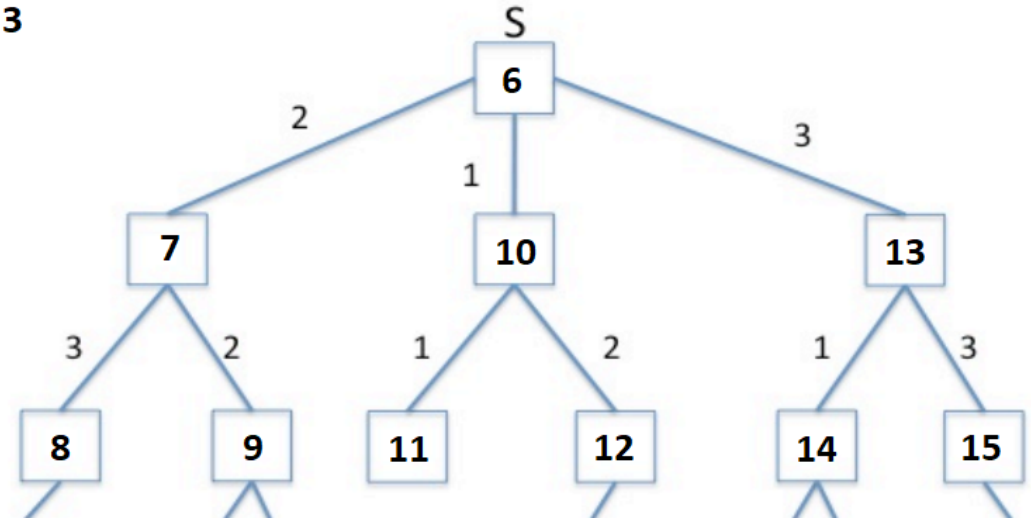
Έχω κάνει ξεχωριστό σχεδιάγραμμα για κάθε iteration για να είναι πιο ευανάγνωστο και προφανές πως δουλεύει ο αλγόριθμος.



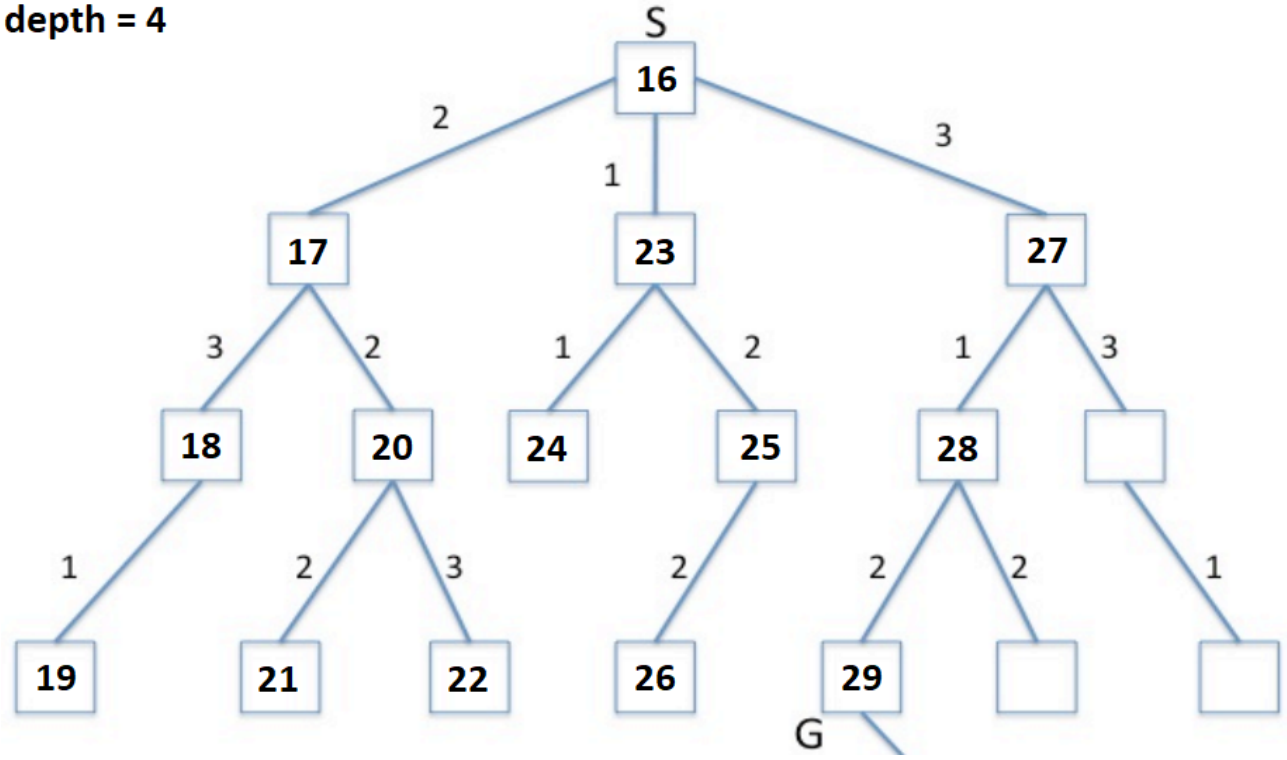
depth = 2



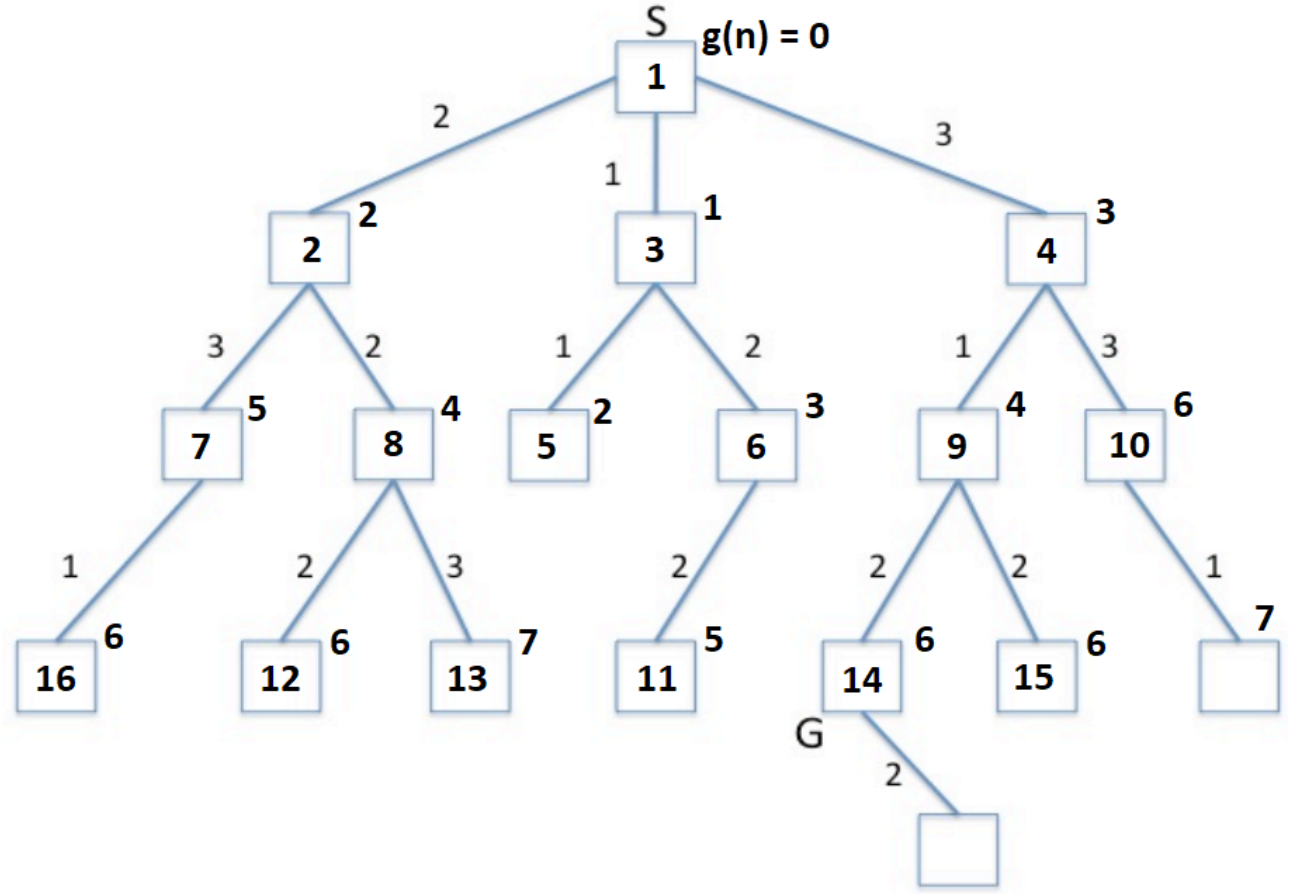
depth = 3



depth = 4



d) Uniform-Cost Search



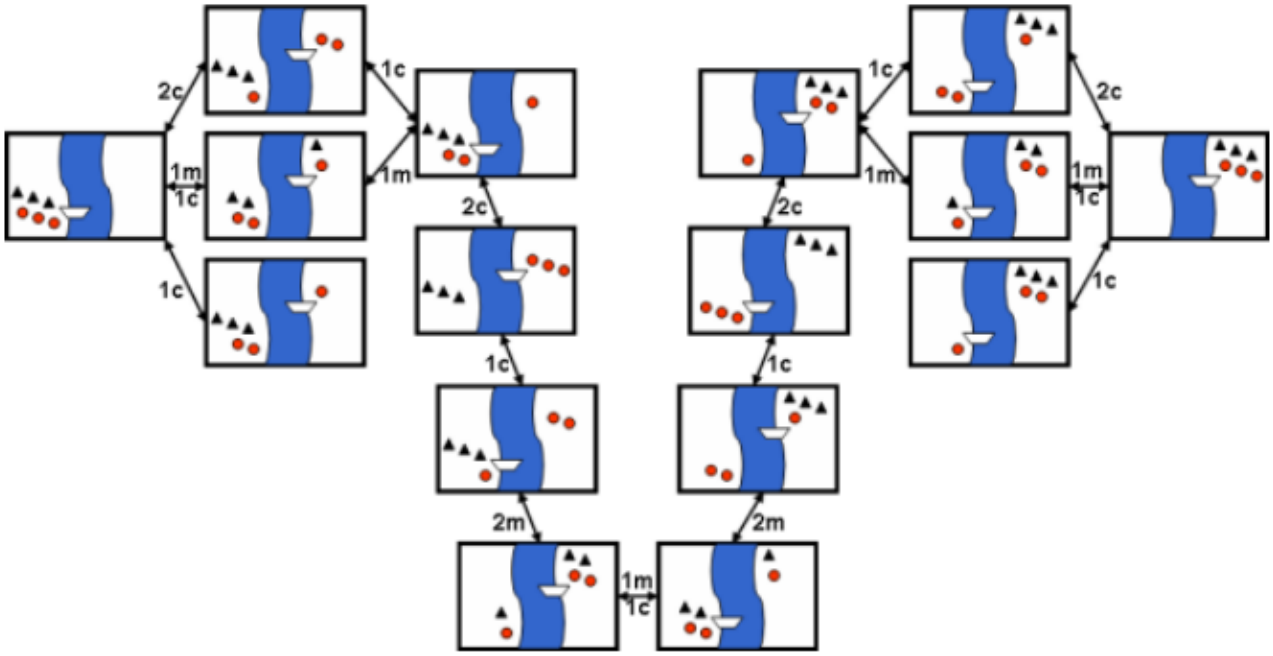
Άσκηση 2)

Το μέτρο απόδοσης (performance measure) δείχνει αντικειμενικά πόσο επιτυχής είναι ο πράκτορας και χρησιμοποιείται από κάποιον τρίτο, τυπικά από τον σχεδιαστή του.

Το Utility function αντιστοιχεί αριθμητική τιμή σε καταστάσεις ή ακολουθίες καταστάσεων και χρησιμοποιείται από τον πράκτορα για να παίρνει λογικές αποφάσεις όταν οι στόχοι είναι ανεπαρκείς.

Άσκηση 3)

a) Η κάθε κατάσταση αποτελείται από 5 μεταβλητές: Πόσοι κανίβαλοι είναι στα αριστερά, πόσοι ιεραπόστολοι είναι στα αριστερά, σε ποια μεριά είναι η βάρκα, πόσοι κανίβαλοι είναι στα δεξιά και πόσοι ιεραπόστολοι είναι στα δεξιά. Στην αρχική κατάσταση όλοι οι κανίβαλοι και οι ιεραπόστολοι είναι στα αριστερά και η στόχος είναι να μεταφερθούν όλοι στα δεξιά. Το κόστος όλων των καταστάσεων είναι το ίδιο. Επιτρέπονται κινήσεις όπου μεταφέρονται 1 ή 2 άτομα μαζί με την βάρκα και δεν καταλήγουν σε κατάσταση όπου οι κανίβαλοι υπερέχουν αριθμητικά των ιεραποστόλων σε οποιαδήποτε μεριά.



State-space του προβλήματος. [Source]

b) Το πρόγραμμα εκτυπώνει πρώτα τις κινήσεις του αλγορίθμου και μετά τις καταστάσεις που πέρασε για να φτάσει στην βέλτιστη λύση. Ο κώδικας βρίσκεται στο repository:

https://github.com/Manwlis/PLH417-theoretical-exercises

Αν χρησιμοποιηθεί ένας πλήρης αλγόριθμος, θα βρεθεί η βέλτιστη λύση ανεξαρτήτου αν εξερευνούνταν επαναλαμβανόμενες καταστάσεις. Όμως, αν δεν επιτρέπονται, θα μειωθούν η πολυπλοκότητα και οι αναγκαίοι υπολογισμοί. Στις περισσότερες καταστάσεις υπάρχει μόνο μια επόμενη έγκυρη κατάσταση πέρα από αυτήν που προήλθε. Για αυτό, αρκεί να αποτρέπεται η επαναφορά στην αμέσως προηγούμενη για να μην υπάρξουν επαναλαμβανόμενες καταστάσεις.

c) Δεν είναι προφανές ότι οι περισσότερες κινήσεις είναι μη επιτρεπτές και ότι είναι αδύνατο να βρεθούμε στις περισσότερες καταστάσεις.

Άσκηση 4)

Ο GRAPH-SEARCH αλγόριθμος, εάν βρει δύο μονοπάτια προς την ίδια κατάσταση, απορρίπτει το νεότερο. Αν αυτό είναι μικρότερο, ο αλγόριθμος θα χάσει την βέλτιστη λύση. Η μέθοδος αναζήτησης uniform-cost επεκτείνει τους κόμβους βάση κόστους, το νεότερο μονοπάτι είναι πάντα ακριβότερο και δεν χάνεται η βέλτιστη λύση. Η breadth-first αναζήτηση ισούται με την uniform-cost όταν το κόστος του βήματος είναι σταθερό.

Αν υπάρχουν δύο μονοπάτια προς τον στόχο, ένα μήκους 1 και κόστους 2 και ένα μήκους 2 και κόστους 1 η iterative deepening αναζήτηση θα προτιμήσει το πρώτο, καθώς η επέκταση γίνεται βάση μήκους.

Άσκηση 5)

a) Αφού ο χώρος είναι γραμμικός, υπάρχουν άπειρες καταστάσεις και μονοπάτια προς τον στόχο.

b) Το μικρότερο μονοπάτι μεταξύ δυο σημείων είναι μια ευθεία. Αν δεν μπορεί να σχηματιστεί μια ευθεία, τότε το μικρότερο μονοπάτι είναι μια γραμμή που αποτελείται από μικρότερες ευθείες (δεν υπάρχουν σχήματα με καμπύλες) και πλησιάζει περισσότερο σε αυτήν. Για να γίνει αυτό πρέπει οι ευθείες να είναι από το αρχικό σημείο σε γωνία, από γωνία σε γωνία και από γωνία στο τελικό σημείο. Υπάρχουν 33 γωνίες, το αρχικό και το τελικό σημείο, άρα σύνολο 35 καταστάσεις.

Άσκηση 6)

a) Τα μονοπάτια που δεν έχουν εξερευνηθεί πλήρως μπορεί να είναι φθηνότερα από το κομμάτι τους που έχει εξερευνηθεί. Τότε είναι άγνωστο αν είναι καλύτερα ή χειρότερα από μια λύση που έχει βρεθεί ήδη.

b) Στη περίπτωση των δέντρων, αν γνωρίζουμε το υπολειπόμενο βάθος, μπορούμε να βρούμε ότι το μέγιστο δυνατό αρνητικό κόστος ισούται με βάθος * c. Αν αυτό δεν μπορεί να κάνει το ανεξερεύνητο μονοπάτι καλύτερο από την ήδη υπάρχουσα λύση, το μονοπάτι δεν χρειάζεται να εξερευνηθεί. Αν δεν είναι γνωστό το βάθος, η σταθερά c δεν βοηθάει.

Στη περίπτωση των γράφων, μπορεί να υπάρχουν βρόγχοι με αρνητικό κόστος όπου θα κάνουν τα μονοπάτια όλο και φθηνότερα. Η σταθερά c δεν βοηθάει.

c) Η βέλτιστη συμπεριφορά είναι αυτή που ελαχιστοποιεί το κόστος. Ο πράκτορας θα επαναλαμβάνει τον βρόγχο συνέχεια.

d) Οι άνθρωποι βαριούνται να οδηγούν συνέχεια από το ίδιο τοπίο. Δηλαδή η αξία της ομορφιάς του μειώνεται κάθε φορά που το επισκέπτονται. Ο πράκτορας, σε κάθε κατάσταση πρέπει να ξέρει ποιες τοποθεσίες έχει επισκεφθεί και πόσες φορές. Το κέρδος από κάθε τοπίο μειώνεται αναλογικά με τις φορές που έχει επισκεφθεί.

e) Ο ύπνος. Η ξεκούραση (αρνητικό κόστος) που προσφέρει προκαλεί τους ανθρώπους να το κάνουν κάθε βράδυ.

Άσκηση 7)

Iasi – Sibiu. Αν ξεκινήσει από το Iasi θα πάει στο Neamt και θα γυρίσει πίσω στην αρχή. Αυτό θα επαναλαμβάνεται επ’ άπειρον. Αν ξεκινήσει από το Sibiu θα πάει μέσω Fagaras-Bucharest-κτλ, ενώ μέσο Rimnicu-Pitesti-Bucharest-κτλ η διαδρομή είναι μικρότερη.

Άσκηση 8)

a) Το πρόβλημα μπορεί να γίνει offline αναζήτηση αν το belief state είναι οι πιθανοί συνδυασμοί εσωτερικών τοίχων.

Το initial belief state ισούται με το set όλων των συνδυασμών εσωτερικών τοίχων, όπου είναι 2^{12} = 4096 αφού υπάρχουν 12 διαφορετικές θέσεις που μπορεί να έχουν ή να μην έχουν τοίχο.

Το μέγεθος του χώρου των belief state είναι 3^{12} = 531.441 γιατί κάθε θέση τοίχου μπορεί να έχει επιβεβαιωθεί ότι έχει ή δεν έχει τοίχο ή να παραμένει ακόμα άγνωστη.

b) Στην αρχική κατάσταση μπορεί να δει 2 θέσεις εσωτερικών τοίχων, άρα υπάρχουν 2^2 = 4 συνδυασμοί.

Άσκηση 9)

Έστω h(n) συνεπής ευρετική συνάρτηση. Τότε:

$h(N) \leq c(N, P) + h(P)$

όπου N ένας κόμβος, P ένας απόγονος του και και c(N, P) το πραγματικό κόστος μεταξύ N και P.

$h(G) = 0$, όπου G ο στόχος.

Έστω ότι το συντομότερο μονοπάτι είναι το $x_0, x_1, \ldots, x_n, g$. Τότε:

$$h(x_n) \leq c(x_n, g) + h(g) = c(x_n, g)$$

Ομοίως

$$h(x_{n-1}) \leq c(x_{n-1}, x_n) + h(x_n) \leq c(x_{n-1}, x_n) + c(x_n, g)$$

Με επαγωγή ως προς το n

$$h(x_k) \leq c(x_k, x_{k+1}) + \ldots + c(x_{n-1}, x_n) + c(x_n, g) \quad k \in (1, n-1)$$

Ο δεύτερος όρος της ανισότητας ισούται με $c(x_k, g)$, άρα

$$h(x_k) \leq c(x_k, g)$$

Δηλαδή h(n) παραδεκτή.

Άσκηση 10)

Ο αλγόριθμος θα έχει παρόμοια συμπεριφορά με την A* αναζήτηση. Το βάρος *α* είναι κοινό και για την ευρετική συνάρτηση *ε(n)* και για το μερικό κόστος *g(n)*. Σαν αποτέλεσμα δεν έχει καμία επίδραση.

π.χ. $g(n_1) + h(n_1) \geq g(n_2) + h(n_2) \Rightarrow a * (g(n_1) + h(n_1)) \geq a * (g(n_2) + h(n_2))$

Άσκηση 11)

Αφού h1, h2, h3 παραδεκτές:

$$\begin{array}{l} h1(n) \leq f(n) \\ h2(n) \leq f(n) \\ h3(n) \leq f(n) \end{array} \quad (1)$$

Για κάθε n.

a) Τότε: $j1(n)=max(h1(n),h2(n),h3(n))\leq f(n)$

άρα j1 παραδεκτή.

b) Παρομοίως: $j2(n)=min(h1(n),h2(n),h3(n))\leq f(n)$

άρα j2 παραδεκτή.

c) Από (1):

$$j3(n)=\Sigma_i w_i h_i(n)\leq \Sigma_i w_i f(n)=w_1 f(n)+w_2 f(n)+w_3 f(n)=(w_1+w_2+w_3) f(n)=f(n)$$
$$\Rightarrow j3(n)\leq f(n)$$

άρα j3 παραδεκτή.

Η καλύτερη εκτίμηση είναι αυτή που είναι πιο κοντά στην πραγματικότητα. Αφού και οι τρεις ευρετικές συναρτήσεις είναι παραδεκτές, οι εκτιμήσεις τους είναι μικρότερες από την πραγματικότητα, με την μεγαλύτερη από αυτές να είναι πιο κοντά σε αυτήν. Για αυτό προτιμάται η j1.