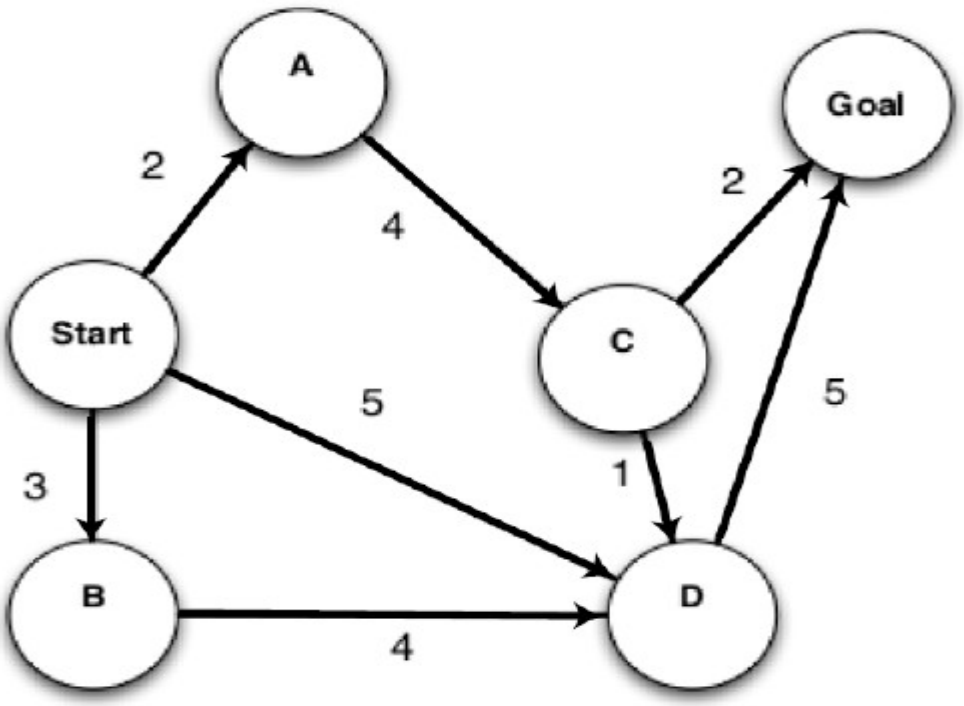
Τεχνητη Νοημοσυνη
1ο Συνολο Ασκησεων:Αναζητηση

Ασκηση 1.1:



1. Αναζήτηση πρώτα σε βάθος (Depth-First Search)

Εφόσον το graph δεν έχει nodes τα οποία επαναλαμβάνονται η nodes που καταλήγουν σε Dead-ends το Depth-First Search θα γίνει ως εξης:

TOTAL_COST	12	12	8	10
NODE		GOAL		
NODE	GOAL	D	GOAL	
NODE	D	C	C	GOAL
NODE	B	A	A	D
START	START	START	START	START

Αρα παρατηρούμε άμεσα οτι το καλύτερο path που μπορεί να λάβει ειναι το: START-A-C-GOAL που εχει συνολικο κοστος 8. Το depth first search ψάχνει έναν-έναν κόμβο μέχρι ήτε να φτάσει στο τέλος του ήτε μέχρι να χρειαστεί να κανει backtrack . Το depth First Search εχει O(V+E) πολυπλόκοτητα οπου V,E είναι Vertices και Edges δηλαδη κορυφες και Ακρες

2. Αναζήτηση πρώτα σε πλάτος (Breadth-First Search)

Το Breadth First Search ειναι πολυ πιο χρήσιμο στο να βρεί το γρηγορότερο μονοπάτι σε σχέση με το Depth First Search καθώς παίρνει ολα τα γειτονικά nodes και δεν χρειάζεται να κανει backtrack, παρόλο που και το Bread First Search έχει O(V+E) πολυπλόκοτητα.
Αρα ξεκινώντας απο το START θα έχουμε το εξής:

NODE	D
NODE	B
NODE	A
NODE	START

Στην συνέχεια θα πάρουμε όλα τα γειτονικά nodes του A,το οποιο παρατηρούμε γρήγορα είναι μονο το C.
Στην συνέχεια παίρνουμε το node B και βλέπουμε οτι το μονο γειτονικο του node ειναι το D.
Στην συνέχεια παίρνουμε το node D και βλέπουμε οτι το μονο γειτονικο του node ειναι το Goal. Αρα συμπληρωνουμε λοιπον το πρωτο ολόκληρο path :
START-D-GOAL με κόστος 10

Δεν εχει τελειωσει ομως εδω ο αλγοριθμος , έχουμε ακομα να επεκτείνουμε τα path τα οποία δέν έχουν ολοκληρωθεί.

Συνεχίζοντας απο το START-A-C path: Παίρνουμε ολα τα γειτονικα node του C.
τα οποία είναι D,GOAL.Βρηκαμε δηλαδη αλλο ενα ολόκληρο path :
START-A-C-GOAL με κόστος 8

Επιπλέον βρήκαμε και το D node που ξέρουμε οτι καταλήγει στο Goal , αρα έχουμε και αλλο ενα ολόκληρο path:
START-A-C-D-GOAL με κόστος 12

Συνεχίζοντας απο το START-B-D path: Παίρνουμε ολα τα γειτονικα node του D.
το οποία είναι GOAL.Βρηκαμε δηλαδη αλλο ενα ολόκληρο path :
START-B-D-GOAL με κόστος 12

‘Εχωντας βρει λοιπον ολα τα nodes και τα ολόκληρα paths, βλέπουμε οτι το καλύτερο path με το χαμηλότερο κοστος ειναι το START-A-C-GOAL .

3. Αναζήτηση ομοιόμορφου κόστους (Uniform Cost Search)

Το Uniform Cost Search έχει μια παρόμοια λογικη με το Breadth-First Search, καθώς στην αρχή παίρνει ολα τα γειτονικά nodes απο το START. Μονο που το Uniform Cost Search εχει ενα subarray που λέγεται priority queue. Δηλαδή μετα που έχει διαλέξει ολα τα γειτονικά nodes απο το START διαλέγει nodes τα οποία έχουν το χαμηλότερο κόστος.

‘Αμεσα παρατηρεί το προγραμμα οτι το χαμηλότερο κοστος απο την αρχή ειναι προς το node A. Απο εκει και περα το Uniform Cost Search δεν εχει επιλογή και συνεχίζει στο C. Εκει βλέπει οτι έχει παλι 2 επιλογές τα nodes D,GOAL.Εφόσον παίρνει το χαμηλότερο κόστος θα δοκιμάσει το node D, απο το node D έχει μονο το GOAL,που ‘εχει κόστος 5 αρα σύνολο 6 που ειναι αρκετα μεγαλύτερο απο πριν. Τότε το προγραμμα θα γυρίσει στο node C και θα παει στο Goal δίνωντας μας το optimal path:
START-A-C-GOAL με κόστος 8.
Αλλα το path που ακολούθησε το προγραμμα ηταν :
START-A-C-D-*-GOAL
* Ξαναγύρισε στο node C αλλα εφόσον το έχουμε βάλει ηδη στην λίστα δεν θα το ξαναγράψουμε.

Ασκηση 1.1:

Τα heuristic cost του γραφου ειναι η εκτιμιση για το ποσο περιπου απεχουν απο το GOAL node που στην περιπτωση μας ειναι το G.

	A	B	C	G
h(1)	1	1	1	0
h(2)	5	4	3	0
h(3)	5	3	0	0
h(4)	6	2	1	0
h(5)	3	2	1	0

h(1):
Για να είναι το heuristic cost (1) σε κάθε node τότε πρέπει να απέχουν μόνο ένα node απο το GOAL. Επίσης θα πρέπει το κόστος της διαδρομής να ειναι 1 πράγμα που βλέπουμε οτι δεν ισχυει . Επομένως το h(1) είναι αποδεκτό εφόσον ολα απέχουν παρα μονο ενα node μακρια απο το GOAL, αλλα δεν ειναι συνεπής καθώς το κόστος διαδρομης τους ειναι πάνω απο 1.

h(2):
A:
Το heuristic cost (5) του A , δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 6 (A-B-C-GOAL) .

B:
Το heuristic cost (5) του B, δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 4 (B- GOAL). Δέν είναι ομως συνέπης εφόσον υπάρχει και αλλη μία πιο φτήνη διαδρομη.

C:
Το heuristic cost (3) του C, δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 1 (C-GOAL) . Επομένως δέν είναι και συνεπής.

H(3):

A:
Το heuristic cost (5) του A , δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 6 (A-B-C-GOAL) .

B:
Το heuristic cost (4) του B, είναι αποδεκτό. Δέν είναι ομως συνέπης εφόσον υπάρχει και αλλη μία πιο φτήνη διαδρομη. (B-C-GOAL)

C:
Δέν γίνεται ενα node να έχει heuristic cost (0), εκτος απο όταν είναι ηδη το GOAL state.

H(4):

A:
Το heuristic cost (6) του A , είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 6. Είναι και συνεπής καθώς είναι η καλύτερη διαδρομή που μπορεί να παρει.

B:
Το heuristic cost (3) του B, είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 3. Είναι και συνεπής καθώς είναι η καλύτερη διαδρομή που μπορεί να παρει.

C:
Το heuristic cost (1) του C είναι συνεπής και αποδέκτο καθως είναι η πιο φτηνή διαδρομη και ταυτόχρονα η καλύτερη.

H(5):

A:
Το heuristic cost (3) του A , δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 6. Αρα δέν ειναι και συνεπής.

B:
Το heuristic cost (2) του B, δεν είναι αποδεκτό καθως το μικρότερο κόστος διαδρομής του ειναι 3. Αρα δέν ειναι και συνεπής.

C:
Το heuristic cost (1) του C είναι συνεπής και αποδέκτο καθως είναι η πιο φτηνή διαδρομη και ταυτόχρονα η καλύτερη.

Ασκηση 1.3:

Το ρομπότ για ψώνια έχει μόνο 2 καταστάσεις , μεταφορα και αγορα.
Επομένως ο χώρος καταστάσεων είναι ουσιαστικα το S.

2.
(α′):
h = min s ′= s t(s, s′), δεδομένου οτι το ρομπότ παίρνει ηδη την καλύτερη απόσταση , ο συντομότερος χρόνος από το τρέχον κατάστημα προς οποιοδήποτε άλλο κατάστημα είναι αποδεκτό και συνεπης.

(β′):
Ο χρόνος για το σπίτι από την τρέχουσα τοποθεσία h = t(s, s1),
Έιναι αποδεκτος , αλλα όχι συνεπης εφόσον η τρέχουσα τοποθεσια(s) αλλαζει συνεχως.

(γ′):
Ο συντομότερος χρόνος προς οποιοδήποτε κατάστημα που πουλάει ένα αντικείμενο που έχει απομείνει στη λίστα,
h = min l ∈ u (min s′ : l ∈ s′ t(s, s′))
Εφόσον το ρομότ ηδη παίρνει το συντομοτερό δρόμο, ο συντομότερος χρόνος προς οποιοδήποτε κατάστημα που πουλάει ένα αντικείμενο που έχει απομείνει στη λίστα είναι αποδεκτος. Δέν μπορει ομως να γινει συνεπης

εφοσον όταν αφαίρει αντικείμενα απο την λίστα η απόσταση και ο χρόνος αλλάζουν.

(δ′)
Ο συνολικός χρόνος που ∑ που θα χρειαστούμε ώστε να αγοράσουμε χωριστά το κάθε αντικείμενο που απομένει στη λίστα
$h = l \in u\ mins' : l \in s'\ t(s, s')$

Ο συνολικός χρόνος που θα χρειαστεί το ρομπότ για να αγοράσει ενα αντικείμενο είναι t(si, sj). Επομένως για να αγοράσει χωριστα το κάθε αντικείμενο που απομένει στη λίστα ο τύπος που μας δόθηκε είναι αποδεκτος καθώς μετάει χωριστά τα αντικείμενα ανα μαγαζί. Είναι και συνεπής αφου το ρομπότ παραμένει στο ιδιο μαγάζι για να αγοράσει τα αντικείμενα.

(ε′)
Το γινόμενο του πλήθους των αντικειμένων που δεν έχουν αγοραστεί με τον συντομότερο χρόνο από κατάστημα σε κατάστημα
$h = |u|\ min si\ ,sj \neq si\ t(si\ ,\ sj\ )$

Επειδή λοιπόν, το ρομπότ παίρνει απο μόνο του το συντομότερο μονοπατι απο μαγαζί σε μαγαζί , ο συνολικός χρόνος του εξαρτάτε απο την λίστα αντικειμένων ανα μαγαζί . Έτσι δεν μπόρει να θεωρειθεί συνεπής η συνάρτηση. Ουτε μπόρει να θεωρειθεί αποδεκτό γιατι το κάθε μαγαζί εχει δικιά του λίστα με αντικείμενα και για αυτόν τον λόγο δεν γίνετε να έχουμε σώστα τον συντομότερο χρόνο των αντικειμένων που δεν έχουν αγοραστεί