INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

CSE643 – Artificial Intelligence

Monsoon 2022 session

Quiz#1, Quiz#2 and Mid-sem makeup exam

11-Nov-2022

Time: **For Quiz #1 and Quiz #2 - 3:00 PM to 3:30 PM** **Max mark:10 (will be scaled to 5)**

For Midsem - 3:00 PM to 5:00 PM **Max mark:50 (will be scaled to 25)**

INSTRUCTIONS:

1. **Laptops, mobiles, ipads and notebooks are not allowed. Closed book quiz/exam.**
2. **If you are giving Quiz#1 makeup then answer questions 1, 2, and 3.**
3. **If you are giving Quiz#2 makeup then answer questions 4 & 5.**
4. **If you are giving Midsem makeup then answer all questions from 1 to 10.**
5. **Ensure that in the answer sheet you write your name and roll number clearly. If you have additional sheets then write your name and roll number on that too. Tie up the answer sheets or put one inside the other and mention number of additional answer sheets. Submit the hard-copy answer sheets to the TAs by 5PM.**

Q1: Represent the following statements in (a) and (b) as propositional logic statements and then using clausal form show the logical equivalence of (a) with the statement in (b). (2 marks)

a) When a person takes AIcourse then he has fun, or when he takes MLcourse then he has fun.
b) When a person takes AIcourse and MLcourse then he has fun.

Answers:

Propositions are:

1) P: Person takes AIcourse
2) Q: Person has fun
3) R: Person takes MLcourse

Statements can be written as:

4) Statement (a) can be written as (P → Q) v (R → Q)
5) Statement (b) can be written as (P ^ R → Q)
6) Proposition in (4) can be written in clausal form as (¬P v Q) v (¬R v Q)
7) Clause in (6) can be written as (¬P v ¬R v Q)
8) Clause in (7) is the logical equivalence of (5) since ≡ (P ^ R → Q)

Thus, we have shown logical equivalence of statement (a) with statement (b).

Q2: Examine the short Prolog program below and answer the questions (3 marks)

(i) Explain in words what the following Prolog program does.
(ii) Show its output with some values for all its arguments.
(iii) Where all does backtracking occur? Where all does recursion occur?

```
oc(_, [], 0).
oc(X, [Y|L], N) :- not(X=Y), oc(X, L, N).
oc(X, [X|L], N) :- oc(X, L, M), N is M+1.
```

Answers:

(i) The Prolog program finds if an element X occurs N times in a given list L. If X occurs in List L N times then the program succeeds and returns True, else False.
(ii) ?-oc(3, [2, 3, 3, 4, 5, 6], 2). Returns true. ?-oc(3, [2, 3, 4, 4, 5, 6], 2). Returns false.
(iii) Backtracking occurs in the second clause where not(X=Y) is checked wherein if the check fails, that is X=Y then it backtracks and goes to the next clause to count how many times that element has occurred. Recursion occurs in clause 2 and clause 3 when the program goes to check in the tail of the list.

Q3: Assume that the spacecraft Chandrayaan-5 is going to be launched to the Moon and its rover Pragyan-5 is being designed to drive around the Moon's surface, collect rock samples and then come back to the Vikram-5 lander. It has the following aspects:

The rover's batteries can be charged by the solar cells and it needs to do that. It can move around. It can pick up a rock. The rover has a map that indicates the type of rock expected in a location and the expected weight of rocks at that location.

Formulate the PEAS for the Pragyan-5 rover. What kind of Agent will you design for this rover? Specify the Agent details. Specify all the characteristics of the environment that the rover will encounter.
(5 marks)

Answers:

Agent: Pragyan-5 rover

Kind of Agent: Goal-based agent

Performance Measure:

i) Types of rocks collected
ii) Number of rocks collected
iii) Locations visited
iv) Able to charge its batteries before it runs out of charge
v) Able to plan its moving based on amount of charge left

Environment:

i) Rocky surface mixed with mounds
ii) Solar light power
iii) Different types of rocks
iv) Estimated weight of rocks
v) Distinguish between rocks and mounds

Actuators

i) Robotic arm and robotic hands to grab/lift rocks
ii) Location identification through cameras
iii) Charging through solar cells
iv) Wheels to move around

Sensors

i) Cameras at various points of the rover
ii) Infrared sensors for distance measurement
iii) Battery charge detection sensor
iv) Sensor for Location with respect to lander
v) Map correlation sensor
vi) Sensors to identify between rocks and mounds

Environment type

i) Fully observable
ii) Deterministic
iii) Episodic
iv) Static (nothing changes on the moon)
v) Discrete (actions like pick up, move, charge)
vi) Single Agent

Q4: There are three parts to this question (5 marks)

i) Express the following as FOPL sentences.

Pahul was a man and he was a gambler. All men are people. Viput Singh was a don. All gamblers were either loyal to Viput Singh or hated him. Everyone is loyal to someone. People only try to assassinate dons they are not loyal to. Pahul tried to assassinate Viput Singh.

ii) Represent the FOPL sentences in CNF showing all steps.

iii) Using resolution refutation (and also draw the graph) determine whether Pahul hated Viput Singh. <Note: in the graph do not use abbreviations for the CNF>

Answers:

Predicates used

i) man(x) /* x is a man */
ii) gambler(x) /*x is a gambler */
iii) people(x) /* x is of type people */
iv) don(x) /* x is a don */
v) loyal(x, y) /* x is loyal to y */
vi) hates(x, y) /* x hates y */
vii) assassinate(x,y) /* x tries to assassinate y */

(i) Representing given sentences in FOPL

1) man(Pahul) Λ gambler(Pahul)
2) ∀(x) man(x) → people(x)
3) don(Viput Singh)
4) ∀(x) gambler(x) → loyal(x, Viput Singh) v hates(x, Viput Singh)
5) ∀(x) people(x) → ∃(y) loyal(x, y)
6) ∀(x) ∀(y) people(x) ^ don(y) ^ assassinate(x, y) → ¬loyal(x, y)
7) assassinate(Pahul, Viput Singh)

(ii) Representing FOPL sentences in CNF

8) man(Pahul)
9) gambler(Pahul)
10) ¬man(x) v people(x) /* rewriting (2) in clausal form
11) don(Viput Singh)
12) ¬gambler(x) v loyal(x, Viput Singh) v hates(x, Viput Singh) /* rewriting (4) in clausal form
13) ∀(x) people(x) → loyal(x, g(x)) /* introducing a skolem function g(x)
14) ¬people(x) v loyal(x, g(x)) /* rewriting the above in clausal form

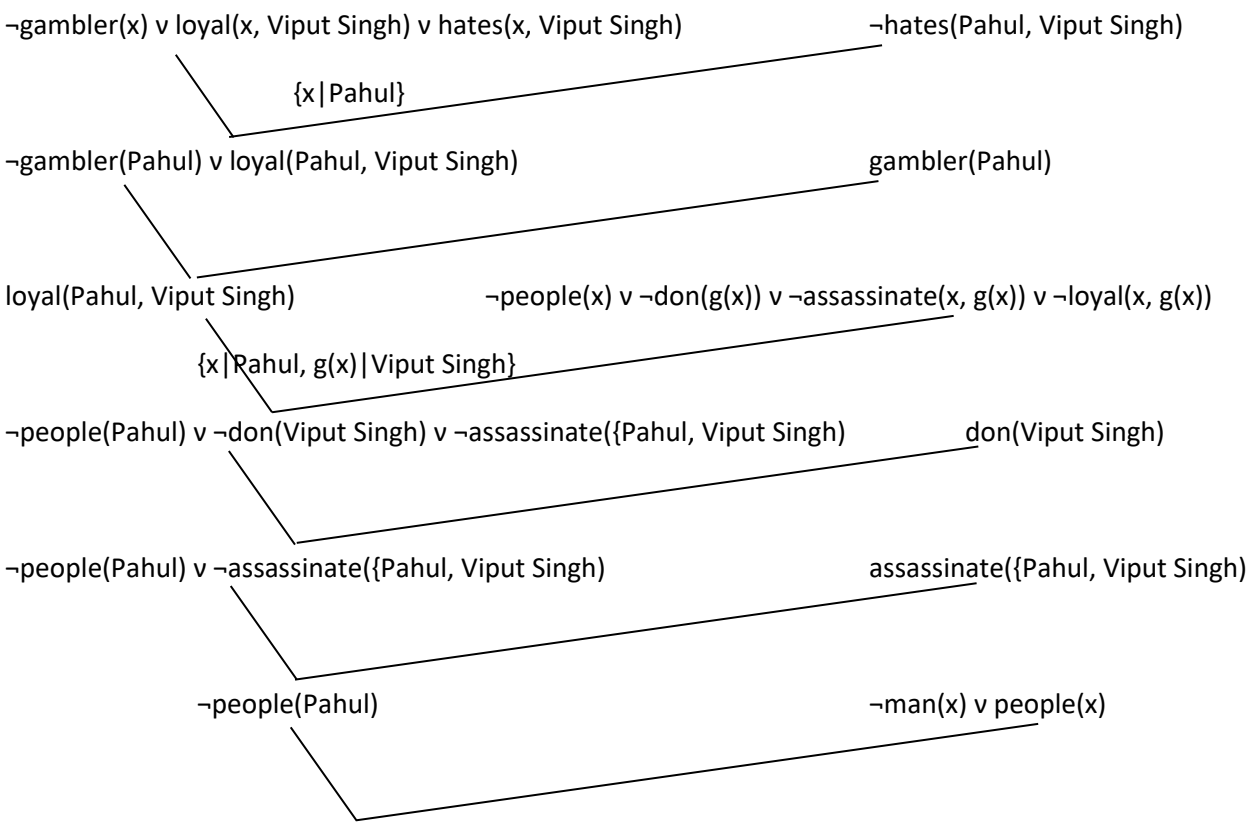
15) ¬people(x) ∨ ¬don(y) ∨ ¬assassinate(x, y) ∨ ¬loyal(x, y) /* rewriting (6) in clausal form
16) ¬people(x) ∨ ¬don(g(x)) ∨ ¬assassinate(x, g(x)) ∨ ¬loyal(x, g(x)) /* using skolem function for y
17) assassinate(Pahul, Viput Singh)

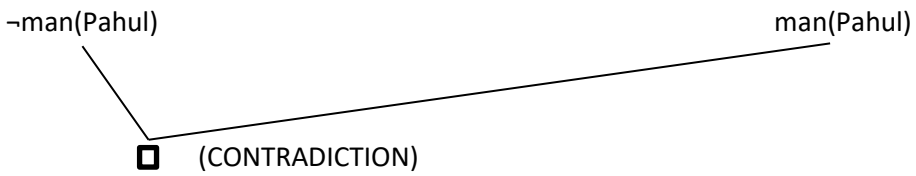
(iii) Resolution refutation

To determine the truth value of the clause hates(Pahul, Viput Singh).

18) Assume negation of the clause ¬hates(Pahul, Viput Singh) and add it to the set of clauses
19) Unifying clause (12) with (18) with substitution {x|Pahul} and resolution of hates predicate we get ¬gambler(Pahul) ∨ loyal(Pahul, Viput Singh)
20) Clause (9) resolves with (19) with resolution of gambler predicate to give loyal(Pahul, Viput Singh)
21) Clause (16) and clause (20) unify with the substitution {x|Pahul, g(x)|Viput Singh} and with the resolution of loyal predicate we get ¬people(Pahul) ∨ ¬don(Viput Singh) ∨ ¬assassinate({Pahul, Viput Singh)
22) Clause (21) resolves with (11) of don predicate and then resolves with (17) of assassinate predicate to give ¬people(Pahul)
23) Clause (22) unifies with (10) with substitution {x|Pahul} and resolves with people predicate to give ¬man(Pahul).
24) Clause (22) and (8) unify with substitution {x|Pahul} and resolve with man predicate to give a CONTRADICTION. Thus, the negation of our hypothesis ¬hates(Pahul, Viput Singh) is FALSE which implies that hates(Pahul, Viput Singh) is TRUE.

Resolution graph





Q5: Answer the following precisely and to the point: (5 marks)

(i) Explain Branch-and-Bound search. What is bound in the search? How does it help?

(ii) Explain Best-first search method. How is it useful?

(iii) Explain A* search method. What is the difference between A* search and Best-first search? Which one is better and why?

(iv) When Dijkstra's search algorithm is available for discovering the shortest path in a graph, why do we need any other search method? Justify.

Answers:

(i)

a. Branch-and-bound search is a refinement over depth-first and breadth-first search where the nodes in the OPEN list for expansion are sorted in terms of the cost of reaching that node from the start node. The cheapest one is picked up for expansion and the search algorithm ignores all nodes that are more expensive than the current cheapest node.

b. The node expansion is bound by the current cheapest cost of reaching a node. This helps by reducing the nodes to consider for expansion.

(ii)

a. Best-first search is a method that evaluates the estimated cost of reaching the goal node from the current child nodes and selects that child node that has the least cost from the heuristic estimates. Thus Best-first search focuses on expanding that node that is estimated closest to the goal. It evaluates nodes by using just the heuristic cost function; that is, f(n) = h(n).

b. Best-first search is useful when we have many child nodes and need to find a quick path to the goal node without evaluating the cost of reaching a node.

(iii)

a. A* search considers the cost of reaching a child node, represented as g(n), and the estimated cost of reaching the goal node from that child node, represented

as h(n) in order to compute the overall cost f(n) as f(n) = g(n) + h(n) to a goal node via that node. A* then selects that child node which has the lowest cost f(n).

b. The difference between A* search and Best-first search is that A* considers the cost of reaching a child node in addition to the estimated cost of reaching the goal node from that child node, whereas Best-first does not consider the cost of reaching a particular node but only focuses on selecting that node that is estimated to reach the goal faster.
c. A* search is better as it will return the optimal cost path from the start node to a goal node.

(iv) Dijkstra's algorithm focuses on finding the path that has the cheapest cost of reaching the goal node, assuming that all the actual costs are known upfront. It does not consider the estimated cost of reaching a goal that can be computed from heuristics. In real-world, since the environment is dynamic, the estimated cost can be computed from heuristics and this helps select the most appropriate path for a goal.

Q6: Answer the following: (6 marks)

(i) Describe Modus Tollens and explain the rationale of its derivation.
(ii) What is Resolution? Show in logic form.
(iii) What is Hypothetical Syllogism? Explain its derivation using resolution.
(iv) What knowledge does the following logic statement express? Is it a valid statement?

$$\exists x \, (\text{StudiesAt}(x, \text{IIITD})) \rightarrow \text{Smart}(x))$$

Answers:

(i) Modus Tollens states that given a proposition $P \rightarrow Q$ and given the fact that $\neg Q$ is TRUE then we can derive that $\neg P$ is TRUE. This is because if P was TRUE then Q would have been derived, but since $\neg Q$ is given then $\neg P$ has to be TRUE else the proposition $P \rightarrow Q$ would be violated.

(ii) In Resolution when we have a clause of the form $(\neg P \vee Q)$ and we have a fact P then we can derive Q from that through the resolution of P with $\neg P$. In logical form we have P and $(\neg P \vee Q)$, which can be written as $(P \wedge \neg P) \vee Q$ which is equivalent to $(F) \vee Q$ which is equivalent to Q. Thus, we can derive Q.

(iii) Hypothetical Syllogism states that given two statements, S1: $P \rightarrow Q$ and S2: $Q \rightarrow R$ then we can derive $P \rightarrow R$. This is possible using resolution as follows: S1 can be

rewritten as ($\neg$P v Q) , and S2 can be rewritten as ($\neg$Q v R). Thus, we have ($\neg$P v Q) Λ ($\neg$Q v R) which can be rewritten as $\neg$P v (Q Λ $\neg$Q) v R where (Q Λ $\neg$Q) resolve to give $\neg$P v (F) v R which gives us ($\neg$P v R) and that is equivalent to P $\rightarrow$ R.

(iv) This statement means that "there exists someone who if he/she studies in IIITD then that x is smart". This is an invalid logical statement.

Q7: Describe Hill-climbing method. What is the problem with Hill-climbing? Why is Hill climbing method required when we have Best-first search? (6 marks)

Answers:

a. The Hill climbing method evaluates all the child nodes of the current node and selects that child node that give the highest gradient or 'gain' (or steepest-ascent) towards the goal. Hill climbing terminates when it reaches a "peak" where no unexplored child node gives a higher value or gain as compared to the current state.
b. The problem with hill climbing is that since it does not look ahead beyond the current set of child nodes it can get stuck at the local maxima, especially when the immediate child nodes do not give a 'gain' or when the child nodes are giving a loss.
c. Hill climbing search is required when we do not have all the heuristic information of all the nodes except for the immediate child nodes. This is useful in various practical scenarios when it is not possible to evaluate the heuristic cost for achieving the goal state from the current set of states except the local gains/ losses.

Q8: You are given the road distance between a few cities below. Describe A* search of finding a route from Calicut to Amritsar in the data given below. Write out the initial state, actions, state-space, path, goal-test for the search. Create your own reasonable heuristic. (10 marks)

Distance in Kilometres	Ahmedabad	Bangalore	Bhubaneshwar	Bombay	Calcutta	Chandigarh
Ahmedabad	-	1490	1697	552	2068	1157
Amritsar	1356	2496	2224	1849	1919	239
Bangalore	1490	-	1538	1013	1961	2296
Bombay	552	1013	1678	-	2012	1645
Calcutta	2068	1461	423	2012	-	1721
Calicut	1648	520	1923	1171	2346	2741

Answers

Assuming heuristic h(n) as (given distance – 200) between city to Amritsar. For cities not directly specified we take an intermediate city and compute the distance from city to Amritsar. Thus, we have h(n) as follows:

City (from Calicut)	g(n)	h(n) (to Amritsar)	f(n)	f*(n)
Ahmedabad	1648	1156	2804	3004
Amritsar	2980 (via Chandigarh)	0	2938	2938
Bangalore	520	2296	2816	3016
Bhubaneshwar	1923	2024	3947	4147
Bombay	1171	1649	2820	3020
Calcutta	2346	1719	4065	4265
Calicut	0	2780	2780	2980
Chandigarh	2741	39	2780	2980

We can see that h(n) is admissible and consistent.

Cycle 1:

Initial State: OPEN: {Calicut}; CLOSED: {}; Goal-test: {Amritsar} fails

Actions: Select and remove node from OPEN – Calicut. Expand Calicut with g(n), h(n) and f(n) for each calculated. Now OPEN is { Ahmedabad (1648,1156,2804) , Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Chandigarh(2741,39,2780) }. Add Calicut to CLOSED.

State space: {Ahmedabad, Bangalore, Bhubhaneshwar, Bombay, Calcutta, Chandigarh}

Goal test: CLOSED {Calicut} not the Goal state

Cycle 2:

Initial State: OPEN is { Ahmedabad (1648,1156,2804) , Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Chandigarh(2741,39,2780) }; CLOSED: {Calicut}; Goal-test: {Amritsar} fails

Actions: Select lowest f value from OPEN and remove node from OPEN – Chandigarh. Expand Chandigarh with g(n), h(n) and f(n) for each calculated. So we get { Ahmedabad(3893,1156,5054), Amritsar(2980,0,2980), Bangalore(5037,2296,7333), Bombay(4386,1649,6035), Calcutta(4462,1719,6181) }.

We note that the parent of Amritsar is Chandigarh while for the rest the parent node is Calicut.

Since Calicut is already in CLOSED list, it is not considered. Now OPEN already has { Ahmedabad (1648,1156,2804) , Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065) } and each of these nodes already have a lower g(n) cost that the ones expanded from Chandigarh these nodes are not updated as the route via that city through Chandigarh is more expensive to what already exists. Thus, only Amritsar(2980,0,2980) is added to OPEN list. Thus the OPEN list now is { Ahmedabad (1648,1156,2804) , Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) } and the parents of all these nodes except Amritsar is Calicut.

Add Chandigarh to CLOSED list.

State space: {Ahmedabad, Bangalore, Bhubhaneshwar, Bombay, Calcutta, Amritsar}.

Goal test: CLOSED {Calicut, Chandigarh} not the Goal state

Cycle 3:

Initial State: OPEN { Ahmedabad (1648,1156,2804) , Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) }. CLOSED: {Calicut, Chandigarh}; Goal-test: {Amritsar} fails

Actions: Select lowest f value from OPEN and remove node from OPEN – Ahmedabad. This is not the goal state. Expand Ahmedabad with g(n), h(n) and f(n) for each calculated. So we get { Amritsar(3004,0,3004), Bangalore(3138,2296,5434), Bhubhaneshwar(3345,2024,5369), Bombay(2200,1649,3849), Calcutta(4414,1719,6133) }. Now OPEN has { Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) }. CLOSED has {Calicut, Chandigarh, Ahmedabad}

State space: {Amritsar, Bangalore, Bhubhaneshwar, Bombay, Calcutta}. Goal-test fails.

Cycle 4:

Initial State: OPEN has { Bangalore (520,2296,2816), Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) }. Choose cheapest, Bangalore, remove from OPEN and add to closed. CLOSED {Calicut, Chandigarh, Ahmedabad, Bangalore}. Goal-test: fails

Actions: Choose lowest f(n) node, so we choose Bangalore and expand. We get {Bhubhaneshwar(2058,2024,4082), Bombay(1533,1649,3182), Calcutta(1981,1719,3700), Amritsar(3016,0,3016)}. Now we see that in OPEN list none of the nodes will be updated in the cost or their parents changed.

Cycle 5:

OPEN has { Bhubhaneshwar (1923,2024,3947), Bombay (1171,1649,2820), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) }. Choose cheapest – Bombay and remove from OPEN and add to CLOSED and expand that node as it is not the goal node.

Actions: Expand Bombay we get {Bhubhaneshwar(2849,2024,4873), Calcutta(3183,1719,4902), Amritsar(3020,0,3020)}. Now we see that none of the nodes in OPEN has a greater f(n) value than these and thus not updated.

Cycle 6:

OPEN has { Bhubhaneshwar (1923,2024,3947), Calcutta (2346,1719,4065), Amritsar(2980,0,2980) }.

We pick the cheapest node that is Amritsar. This is the goal-node and its parent is Chandigarh. And Chandigarh node parent is Calicut. So the goal-test succeeds and we return the path {Calicut, Chandigarh, Amritsar}. The search terminates successfully.

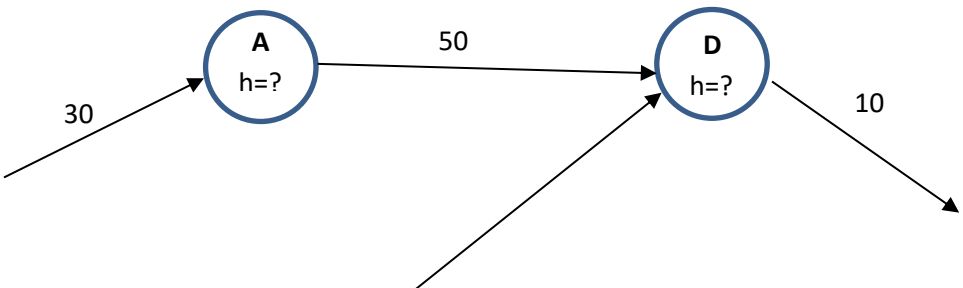
We can see that h(n) is admissible and consistent from all the values as for any node h(s) ≤ h*(s). It is consistent as h(n) ≤ h(n) ≤ c(n, n') + h(n').

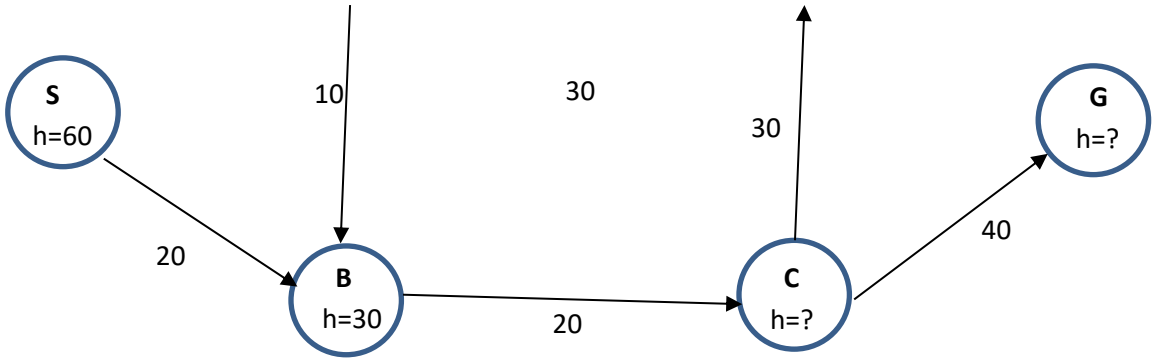
Search cost: Examined 6 nodes. In general, it can be $O(b^d)$ where b is the branching factor and d the depth.

Q9: There are two parts to this question.

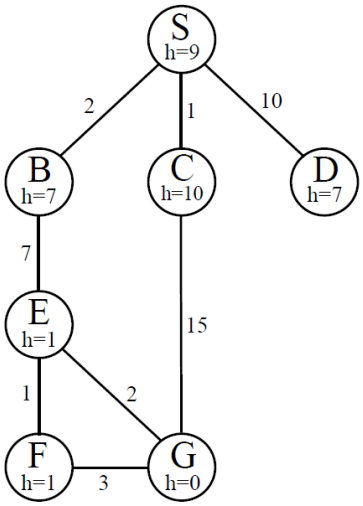
(a) Consider the following graph. (3 marks)

i) Create heuristics such that in A* search the heuristic is admissible.
ii) Can you show that your heuristics is consistent?
iii) If incidentally your heuristics is not consistent then what should be done to make it consistent?





(b) Assume the following search graph is given. The path cost and heuristics are given with the nodes. S is the start node and G is the goal node. What are the paths returned for DFS, BFS, Best-first and A* search. (2 marks)



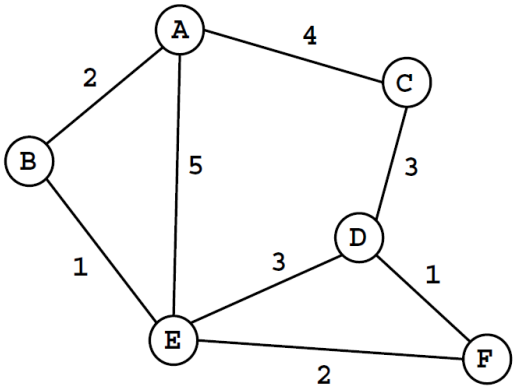
Answers:

a) Heuristics are:
   i. h(G) = 0, h(D) = 10, h(C) = 40, h(A) = 40. Since each heuristic is less than equal to the true cost to reach the goal, it never overestimates the cost to the goal and thus this heuristic is admissible.
   ii. This heuristic is not consistent since the h(S) = 60 and it is greater than 20 (the cost to the next node B) plus 30 (the heuristic at node B).
   iii. In order to make this heuristic consistent we need to change h(S) to 50. Otherwise this heuristic cannot be made consistent.

b) Paths returned are
   i. Depth-first search: S-B-E-F-G
   ii. Breadth-first search: S-C-G
   iii. Best-first search: S-B-E-G

iv. A* search: S-B-E-G

Q10: For the given graph below in which A is the start node and F is the goal node, the cost of traversing to a node is given. Let the heuristic function be the minimum number of arcs between node n and the goal node. Is this an admissible heuristic? Justify. Also show in what order does A* visit the nodes and what are the estimated values when they are visited? (3 marks)



Answers:

The heuristic values are h(A) = 2, h(B) = 2, h(C) = 2, h(D) = 1, h(E) = 1, h(F) = 0.

The heuristic is admissible since the heuristic value at each node is less than or equal to the optimal value to the goal node.

A* visits the nodes and gives the path as A-B-E-F and the estimated values are at f(A) = 0+2 = 2, f(B) = 2+2 = 4, f(E from A) = 5+1 = 6, f(C) = 4+2 =6. First node B is expanded and then f(E from B) = 3+1 =4 is expanded since it is the cheapest.