**Chapter 5**

**Informed Search**

**The chapter consist of Short type Questions & Answers , Descriptive Question & Answer and MCQs & answers**

Contents

[Short type Questions & Answers 2](#_Toc14943507)

[Define greedy best-first search. 2](#_Toc14943508)

[In branch and bound (B&B), how is the upper bound (UB) calculated? 3](#_Toc14943509)

[How is the lower bound (LB) calculated for a path? 3](#_Toc14943510)

[With B&B, when do we prune a path? 3](#_Toc14943511)

[What is the distinction between informed and uninformed search? 3](#_Toc14943512)

[What is a heuristic? 3](#_Toc14943513)

[When is a heuristic admissible? 3](#_Toc14943514)

[A\* can be seen as a combination of what two search strategies? 3](#_Toc14943515)

[Define A\* search. 3](#_Toc14943516)

[Q With reference to the Travelling Salesman Problem explain what is meant by combinatorial explosion and what effect this has in finding an optimal solution? 4](#_Toc14943517)

[What do you mean by Recursive best-first search? 4](#_Toc14943518)

[Define Hill Climbing Search. 4](#_Toc14943519)

[Mention the types of hill-climbing search. 5](#_Toc14943520)

[Why a hill climbing search is called a greedy local search? 5](#_Toc14943521)

[Define genetic algorithm. 5](#_Toc14943522)

[Descriptive Question & Answer 5](#_Toc14943523)

[Q Using the A\* algorithm work out a route from town A to town M. Use the following cost functions. 5](#_Toc14943524)

[G(n) = The cost of each move as the distance between each town (shown on map). 5](#_Toc14943525)

[H(n) = The Straight Line Distance between any town and town M. These distances are given in the table below 5](#_Toc14943526)

[The Search Tree 6](#_Toc14943527)

[Q Would A\* be optimal with an admissible heuristic if we checked for goal states upon generation rather than expansion? Give an articulate, solid proof or counterexample, whichever is appropriate. 8](#_Toc14943528)

[Q Recall that for the A\* algorithm, the evaluation function, f, for a node n is such that f(n) = g(n) + h(n) where h(n) is a non-overestimating estimate of the cost to reach a goal from n. What interesting things can you say about the behavior of the algorithm when 9](#_Toc14943529)

[Q Explain Hill Climbing with its types and algorithm for each type 9](#_Toc14943530)

[**Hill climbing** 9](#_Toc14943531)

[Q Explain Branch and Bound Search with its Concept, Algorithm, Implementation, Advantages and Disadvantages 12](#_Toc14943532)

[Q Explain simulated annealing and write its algorithm . 18](#_Toc14943533)

[Consider the below search problem, and traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table. 20](#_Toc14943534)

[Traverse the given graph using the A\* algorithm. 21](#_Toc14943535)

[The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state. 21](#_Toc14943536)

[Q Consider the search problem represented in Figure ??, where a is the start node and e is the goal node. The pair [f, h] at each node indicates the value of the f and h functions for the path ending at that node. Given this information, what is the cost of each path? The cost < a, c >= 2 is given as a hint. Is the heuristic function h admissible? Explain why or why not. 23](#_Toc14943537)

[Q Consider the search problem represented in Figure ??, where a is the start node and there are goal nodes at f and j. For each node, the heuristic cost is indicated on the node, and for each arc, the arc cost is indicated along the arc. Neighbors are ordered according to the f function. What is the UB when only the start node has been explored? Which goal node is found first by B&B? What is the UB immediately after the first goal node is found? Is the second goal found by B&B? (upper bound (UB) , branch and bound (B&B)) 24](#_Toc14943538)

[MCQs & answers 24](#_Toc14943539)

# Short type Questions & Answers

## Define greedy best-first search.

Greedy best-first search expands the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using the heuristic function f(n) = h(n).

  Define Consistency.

A heuristic h(n) is consistent if, for every node n and every successor n’ of

 n generated by any action a, the estimated cost of reaching the goal n is no greater than the step cost of getting to n’ plus the estimated cost of reaching the goal from n’:

h(n) <= c(n, a, n’) | h(n’)

## In branch and bound (B&B), how is the upper bound (UB) calculated?

Answer: It’s the cost of the best solution found so far. If no solution has been found, the upper bound is infinite.

## How is the lower bound (LB) calculated for a path?

Answer: LB(p) = f(p) = cost(p) + h(p)

## With B&B, when do we prune a path?

Answer: We prune the path p if LB(p) ≥ UB.

## What is the distinction between informed and uninformed search?

Answer: Uninformed search doesn’t take into account any information about the goal (until the goal is reached). Informed search uses estimates of distance to the goal.

## What is a heuristic?

Answer: A heuristic is an estimate of the distance to the goal.

## When is a heuristic admissible?

Answer: A heuristic is admissible if it doesn’t overestimate the distance to the goal.

## A\* can be seen as a combination of what two search strategies?

Answer: It can be viewed as a combination of lowest-cost-first and best-first.

## Define A\* search.

 A\* search evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal.

## Q With reference to the Travelling Salesman Problem explain what is meant by combinatorial explosion and what effect this has in finding an optimal solution?

The number of solutions is n! (n factorial), where n is the number of cities. This results in an exponential rise in the number of solutions. For example, for 10 cities the number of possible routes is 3,628,800.  
This is known as combinatorial explosion where the number of solutions rises exponentially.  
The effect this has with regards to TSP is that is quickly becomes impossible to search the entire search space (i.e. enumerate all possible solutions and choose the best route).

Therefore, heuristic methods are often used to find solutions to these problems.

## What do you mean by Recursive best-first search?

 Recursive best-first search is a simple recursive algorithm that attempts to minimize the operation of standard best-first search, but using only linear space.

 What are the reasons that hill climbing often gets stuck?

Local maxima:

 A local maximum is a peak that is higher than each of its neighboring states, but lower than the global maximum.

 Ridges:

 Ridges results in a sequence of local maxima that is very difficult for greedy algorithms to navigate.

 Plateaux:

A Plateaux is an area of the state space landscape where the evaluatin function is flat.

## Define Hill Climbing Search.

The hill climbing search algorithm is simply a loop that continually moves in the direction of increasing value that is uphill. It terminates when it reaches a “peak”

where no neighbor has a higher value.

## Mention the types of hill-climbing search.

                                                              Stochastic hill climbing

                                                              First-choice hill climbing

                                                              Random-restart hill climbing

## Why a hill climbing search is called a greedy local search?

Hill climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next.

## Define genetic algorithm.

A genetic algorithm is a variant of stochastic beam in which successor states are generated by combining two parent states, rather than by modifying a single state.

# Descriptive Question & Answer

## Q Using the A\* algorithm work out a route from town A to town M. Use the following cost functions.

## G(n) = The cost of each move as the distance between each town (shown on map).

## H(n) = The Straight Line Distance between any town and town M. These distances are given in the table below

36

122

43

40

45

61

80

112

36

32

102

32

31

52

20

31

M

B

D

E

G

H

I

J

F

C

K

A

L

* **.**

**Provide the search tree for your solution and indicate the order in which you expanded the nodes. Finally, state the route you would take and the cost of that route.**

**Straight Line Distance to M**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **223** |  | **E** | **165** |  | **I** | **100** |  | **M** | **0** |
| **B** | **222** |  | **F** | **136** |  | **J** | **60** |  |  |  |
| **C** | **166** |  | **G** | **122** |  | **K** | **32** |  |  |  |
| **D** | **192** |  | **H** | **111** |  | **L** | **102** |  |  |  |

Using A\* Algorithm

## The Search Tree

The figures next to each node represent the G(n) and H(n) functions, where

G(n) = The cost of the search so far (i.e. distance travelled)

H(n) = The heuristic value (i.e. the straight line distance to the target town)

**1**

**A**

**3**

**C**

**2**

**B**

**7**

**L**

**6**

**F**

**5**

**D**

**4**

**A**

**10**

**K**

**9**

**J**

**8**

**C**

**13**

**M**

**12**

**J**

**11**

**F**

0+223=223

61+166=227

36+222=258

141+102=243

92+136=228

93+192=285

122+223=345

204+32=236

214+60=274

123+166=289

236+0=236

240+60=300

316+136=452

The route you would take is A, C, F, K, M at a cost of 236.

c) **The straight line distance heuristic used above is known to be an admissible heuristic. What does this mean and why is it important?**

An admissible heuristic is one which never over estimates the cost to the goal. This is obviously the case with the straight line distance between two towns.

Having admissible heuristics is important as it allows the A\* algorithm to be proved to be optimal (i.e. always find the best solution).

## Q Would A\* be optimal with an admissible heuristic if we checked for goal states upon generation rather than expansion? Give an articulate, solid proof or counterexample, whichever is appropriate.

Answer: No. Try it on this graph, where the only goal is D:

A

/ \

3/ \5

/ \

(h=1)B C(h=6)

\ /

9\ /6

\ /

D(h=0, goal)

First we expand a, generating B with f=3+1=4 and C with f=5+6=11.

So we expand B next, which generates D. D has cost f=12+0=12.

If we stop now, we return the path A-B-D, but this is not optimal.

The optimal path goes through C. The real A\* algorithm would expand

C first.

## Q Recall that for the A\* algorithm, the evaluation function, f, for a node n is such that f(n) = g(n) + h(n) where h(n) is a non-overestimating estimate of the cost to reach a goal from n. What interesting things can you say about the behavior of the algorithm when

1. **h is a perfect estimate**
2. **h(n) is zero for every n**

If h was a perfect estimate, A\* would directly expand the least cost path to the goal. If h(n) was always 0, A\* "degenerates" into uniform-cost search.

## Q Explain Hill Climbing with its types and algorithm for each type

**Hill climbing**

Hill climbing is a variant of generate-and-test in which feedback from the procedure is used to help the generator decide which direction to move in the search space. In a pure generate-and-test procedure, the test function responds with only a yes or no. But if the test function is augmented with a heuristic function that provides an estimate of how close a given state is to a goal state, the generate procedure can exploit it. This is particularly nice because often the computation of a heuristic function can be done at almost no cost at the same time that the test for a solution is being performed. Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available.

**FOR EXAMPLE:**

Suppose you are in an unfamiliar city without a map and you want to get downtown. You simply aim for the tall buildings. The heuristic function is just distance between the current location and the location of the tall buildings and the desirable states are those in which this distance is minimized. Getting downtown is an example of such a problem. For these problems, hill climbing can terminate whenever a goal state is reached.

For maximization (or minimization)problems, such as the traveling salesman problem. In these problems, there is no a priori goal state. For problems of this sort, it makes sense to terminate hill climbing when there is no reasonable alternative state to move to.

**SIMPLE HILL CLIMBING:**

The simplest way to implement hill climbing

**ALGORITHM: *SIMPLE HILL CLIMBING***

1.       Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

2.       Loop until a solution is found or until there are no new operators left to applied in the current state:

a)      Select an operator that has not yet been applied to the current state and apply it to produce a new state

b)      Evaluate the new state,

                                             i.            If it is a goal state, then return it and quit.

                                           ii.            If it is not a goal state but it is better than the current state, then make it the current state.

                                          iii.            If it is not better than the current state, then continue in the loop.

The key difference between this algorithm and the one we gave for generate-and-test is the use of an evaluation function as a way to inject task-specific knowledge into the control process.

**EXAMPLE FOR HOW HILL CLIMBING WORKS:**

                    The puzzle of the four colored blocks. To solve the problem, we first need to define a heuristic function that describes how close a particular configuration is to being a solution. One such function is simply the sum of the number of different colors on each of the four sides. A solution to the puzzle will have a value of 16. Next we need to define a set of rules that describe ways of transforming one configuration into another. Actually, one rule will suffice. It says simply pick block and rotate it 90degrees in any direction. Having provided these definitions, the next step is to generate a starting configuration. This can either be done at random or with the aid of heuristic function. Now hill climbing can begin. We generate a new state by selecting a block and rotating it. If the resulting state is better, then we keep it. If not, we return to previous state and try a different perturbation.

**STEEPEST-ASCENT HILL CLIMBING:**

 A useful variation on simple hill climbing considers all the moves from the current state and selects the best one as the next state. This method is called steepest-ascent hill climbing or gradient search. Notice that this contrasts with the basic method in which the first state that is better than the current state is selected.

**ALGORITHM: *STEEPEST-ASCENT HILL CLIMBING***

1.       Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

2.       Loop until a solution is found or until a complete iteration produces no change to current state:

a)      Let SUCC be a state such that any possible successor of the current state will be better then SUCC.

b)      For each operator that applies to the current state do:

                                             i.            Apply the operator and generate a new state.

                                    ii.            Evaluate the new state. If it is goal state, then return it and quit. If not, compare it to SUCC. If it is better, then set SUCC to this state. If it is not better, leave SUCC alone.

c)       If the SUCC is better than current state, then set current state to SUCC.

To apply steepest-ascent hill climbing to the colored blocks, we must consider all perturbations of the initial state and choose the best. For this problem, this is difficult since there are so many possible moves. There is a trade-off between the time required to select a move (usually longer for steepest-ascent hill climbing) and the number of moves required to get a solution (usually longer for basic hill climbing) that must be considered when deciding which method will work better for a particular problem.

      Both basic and steepest-ascent hill climbing may fail to find a solution. Either algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated. This will happen if the program has reached either a local maximum, a plateau, or a ridge.

**LOCAL MAXIMUM:**

 A local maximum is a state that is better than all its neighbors but is not better than some other states farther away. At a local maximum, all moves appear to make things worse. Local maxima are particularly frustrating because they often occur almost within sight of a solution. In this case, they called foothills.

**PLATEAU:**

A plateau is a flat area of the search space in which a whole set of neighboring states have the same value. On a plateau, it is not possible to determine the best direction in which to move by making local comparisons.

**RIDGE:**

A ridge is a special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope. But the orientation of the high region, compared to the set of available moves and the directions in which they move, makes it impossible to traverse a ridge by single moves.

There are some ways of dealing with these problems, although these methods are by no means guaranteed:

         Apply two or more rules before doing the test. This corresponds to moving in several directions at once. This is particularly good strategy for dealing with ridges.

Consider the blocks world problem. Assume the same operators (i.e., pick up one block and put it on the table; pick up one block and put it on another one) suppose we use the following heuristic function:

Local: Add one point for every block that is resting on the thing it is supposed to resting on. Subtract one point for every block that is sitting on the wrong thing.

Using this function, the goal state has a score of 8. The initial state has a score of 4 (since it gets one point added for blocks C,D,E,F,G and H and one point subtracted for blocks A and B). There is only one move from the initial state, namely to move block A to the table. That produces a state with a score of 6. The hill-climbing procedure will accept that move. From the new state, there are three possible moves, leading to the three states. These states have the score: (a) 4, (b) 4, and (c) 4. Hill climbing will halt because all these states have lower scores than the current state. The process has reached a local maximum that is not the global maximum.

## Q Explain Branch and Bound Search with its Concept, Algorithm, Implementation, Advantages and Disadvantages

**Branch and Bound Search**

Branch and Bound is an algorithmic technique which finds the optimal solution by keeping the best solution found so far. If partial solution can’t improve on the best it is abandoned, by this method the number of nodes which are explored can also be reduced. It also deals with the optimization problems over a search that can be presented as the leaves of the search tree. The usual technique for eliminating the sub trees from the search tree is called pruning. For Branch and Bound algorithm we will use stack data structure.

**Concept:**

**Step 1:**Traverse the root node.

**Step 2:**Traverse any neighbour of the root node that is maintaining least distance from the root node.

**Step 3:**Traverse any neighbour of the neighbour of the root node that is maintaining least distance fromthe root node.

**Step 4:**This process will continue until we are getting the goal node.

**Algorithm:**

**Step 1:**PUSH the root node into the stack.

**Step 2:**If stack is empty, then stop and return failure.

**Step 3:**If the top node of the stack is a goal node, then stop and return success.

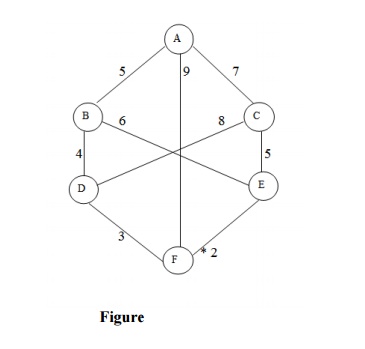
**Step 4:**Else POP the node from the stack. Process it and find all its successors. Find out the pathcontaining all its successors as well as predecessors and then PUSH the successors which are belonging to the minimum or shortest path.

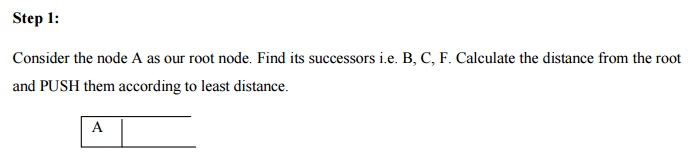
**Step 5:**Go to step 5.

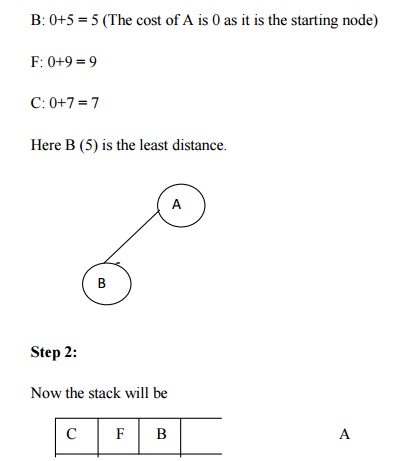
**Step 6:**Exit.

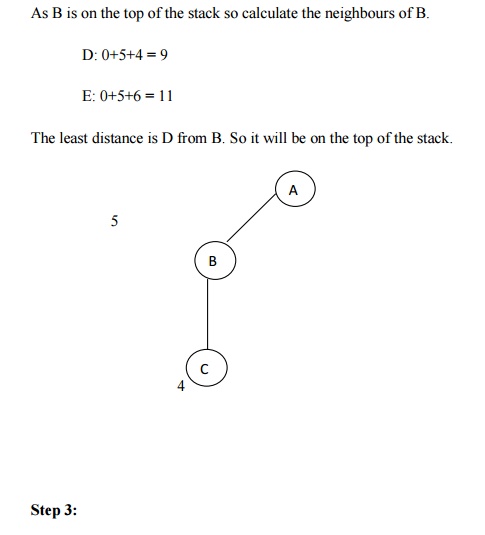
**Implementation:**

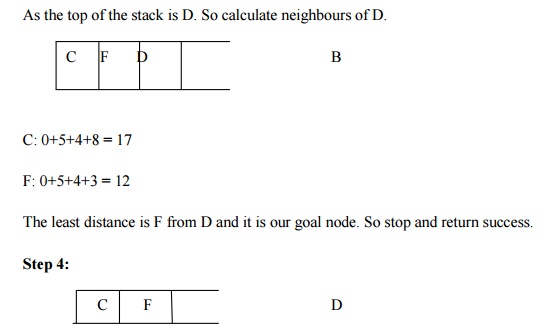
Let us take the following example for implementing the Branch and Bound algorithm.











Hence the searching path will be A-B -D-F

**Advantages:**

As it finds the minimum path instead of finding the minimum successor so there should not be any repetition.

The time complexity is less compared to other algorithms.

**Disadvantages:**

The load balancing aspects for Branch and Bound algorithm make it parallelization difficult.

The Branch and Bound algorithm is limited to small size network. In the problem of large networks, where the solution search space grows exponentially with the scale of the network, the approach becomes relatively prohibitive.

## Q Explain simulated annealing and write its algorithm .

**simulated annealing:**

Simulated annealing is a variation of hill climbing in which, at the beginning of the process, some downhill moves may be made. The idea is to do enough exploration of the whole space early on so that the final solution is relatively insensitive to the starting state. This should lower the chances of getting caught at a local maximum, a plateau, or a ridge.

In order to be compatible with standard usage in discussions of simulated annealing. We make two notational changes for the duration of this section. We use the term objective function in place of the term heuristic function.

And we attempt to minimize rather than maximize the value of the objective function. Thus we actually describe a process of valley descending rather than hill climbing.

**three differences for simulated annealing from the simple hill-climbing procedure:**

         The annealing schedule must be maintained.

         Moves to worse states may be accepted.

         It is a good idea to maintain, in addition to the current state, the best state found so far. Then, if the final state is worse than that earlier state, the earlier state is still available.

**algorithm: *simulated annealing***

1.       Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

2.       Initialize BEST-SO-FAR to the current state.

3.       Initialize T according to the annealing schedule.

4.       Loop until a solution is found or until there are no new operators left to be applied in the current state.

a)      Select an operator that has not yet been applied to the current state and apply it to produce a new state.

b)      Evaluate the new state. Compare

                      ∆E= (value of current)—(value of new state)

         If the new state is a goal state, then return it and quit.

         If it is not a goal state but is better than the current state, then make it the current state. Also set BEST-SO-FAR to this new state.

         If it is not better than the current state, then make it the current state with probability p’ as defined above. This step is usually implemented by invoking a random number generator to produce a number in the range [0, 1]. If that number is less than p’, then the move is accepted. Otherwise, do nothing.

         Revise T as necessary according to the annealing schedule.

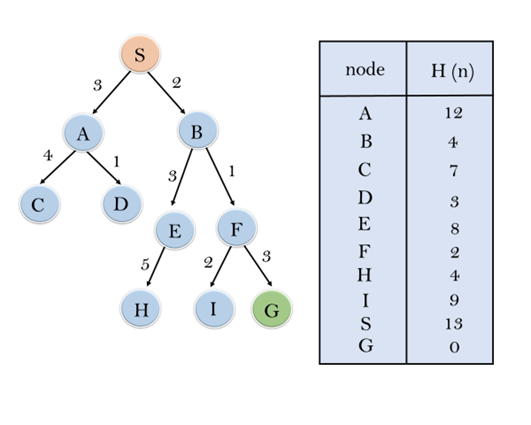
c)       Revise T as necessary according to the annealing schedule.

5.       Return BEST-SO-FAR, as the answer.

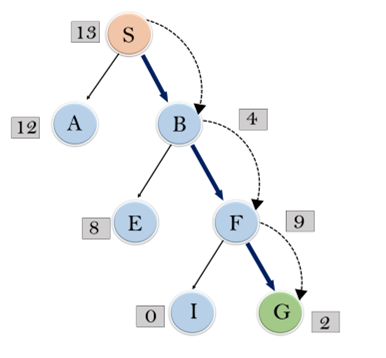
To implement this revised algorithm, it is necessary to select an annealing schedule, which has three components. The first is the initial value to be used for temperature. The second is the criteria that will be used to decide when the temperature of the system should be reduced. The third is the amount by which the temperature will be reduced each time it is changed. There may also be a fourth component of the schedule, namely, when to quit. Simulated annealing is often used to solve problems in which the number of moves from a given state is very large.

For such problem, it may not make sense to try all possible moves. Instead, it may be useful to exploit some criterion involving the number of moves that have been tried since an improvement was found.

## Consider the below search problem, and traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



**Expand the nodes of S and put in the CLOSED list**

**Initialization:** Open [A, B], Closed [S]

**Iteration 1:** Open [A], Closed [S, B]

**Iteration 2:** Open [E, F, A], Closed [S, B]  
                  : Open [E, A], Closed [S, B, F]

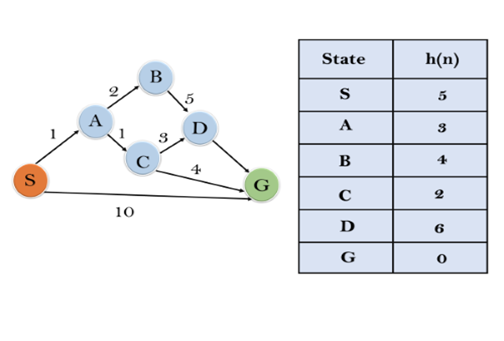
**Iteration 3:** Open [I, G, E, A], Closed [S, B, F]  
                  : Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

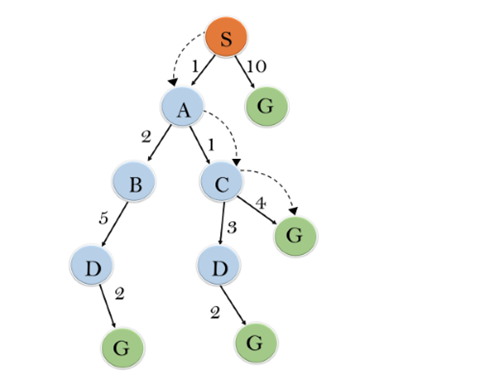
## Traverse the given graph using the A\* algorithm.

## The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.

Here we will use OPEN and CLOSED list.



**Solution:**



**Initialization:** {(S, 5)}

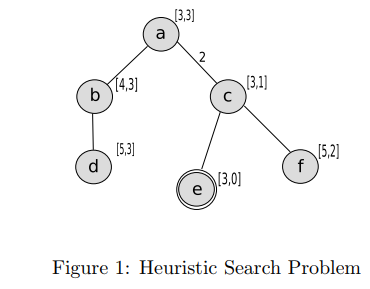
**Iteration1:** {(S--> A, 4), (S-->G, 10)}

**Iteration2:** {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

**Iteration3:** {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

**Iteration 4** will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

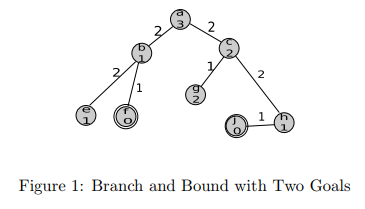
## Q Consider the search problem represented in Figure ??, where a is the start node and e is the goal node. The pair [f, h] at each node indicates the value of the f and h functions for the path ending at that node. Given this information, what is the cost of each path? The cost < a, c >= 2 is given as a hint. Is the heuristic function h admissible? Explain why or why not.



Answer: < a, b >= 1, < b, c >= 1, < a, c >= 2, < c, e >= 1, < c, f >= 1

Yes, it is admissible because it never overestimates the distance to the goal.

## Q Consider the search problem represented in Figure ??, where a is the start node and there are goal nodes at f and j. For each node, the heuristic cost is indicated on the node, and for each arc, the arc cost is indicated along the arc. Neighbors are ordered according to the f function. What is the UB when only the start node has been explored? Which goal node is found first by B&B? What is the UB immediately after the first goal node is found? Is the second goal found by B&B? (upper bound (UB) , branch and bound (B&B))



Answer: The UB is ∞ when only the start node has been explored. The goal node at f is found first, and the UB immediately after is 3. The second goal is not found, as its path is pruned

# MCQs & answers

**1.** What is the other name of informed search strategy?

(a) Simple search

(b) Heuristic search

(c) Online search

(d) None of the above

**2.** How many types of informed search method are in artificial intelligence?

(a) 1

(b) 2

(c) 3

(d) 4

**3.** Which search uses the problem specific knowledge beyond the definition of the problem?

(a) Informed search

(b) Depth-first search

(c) Breadth-first search

(d) Uninformed search

**4.** What is the heuristic function of greedy BFS?

(a) f (n) != h(n)

(b) f (n) <h(n)

(c) f (n) = h(n)

(d) f (n) >h(n)

**5.** Which is used to improve the performance of heuristic search?

(a) Quality of nodes

(b) Quality of heuristic function

(c) Simple form of nodes

(d) None of the above

6. Which search method will expand the node that is closest to the goal?

(a) Best-first search

(b) Greedy best-first search

(c) A\* search

(d) None of the mentioned

7. Which search is complete and optimal when h(n) is consistent?

(a) Best-first search

(b) Depth-first search

(c) Both Best-first & Depth-first search

(d) A\* search

8. Which method is used to search better by learning?

(a) Best-first search

(b) Depth-first search

(c) Metalevel state space

(d) None of the mentioned

9. Which search uses only the linear space for searching?

(a) Best-first search

(b) Recursive best-first search

(c) Depth-first search

(d) None of the mentioned

**Answers**

**1. (b) 2. (d) 3. (a) 4. (c) 5. (b) 6. (b) 7. (d) 8. (c) 9. (b)**