Heuristic Methods:

Intuitive search methods guided by evidence.

1st: Hill Climbing Search (HCS):

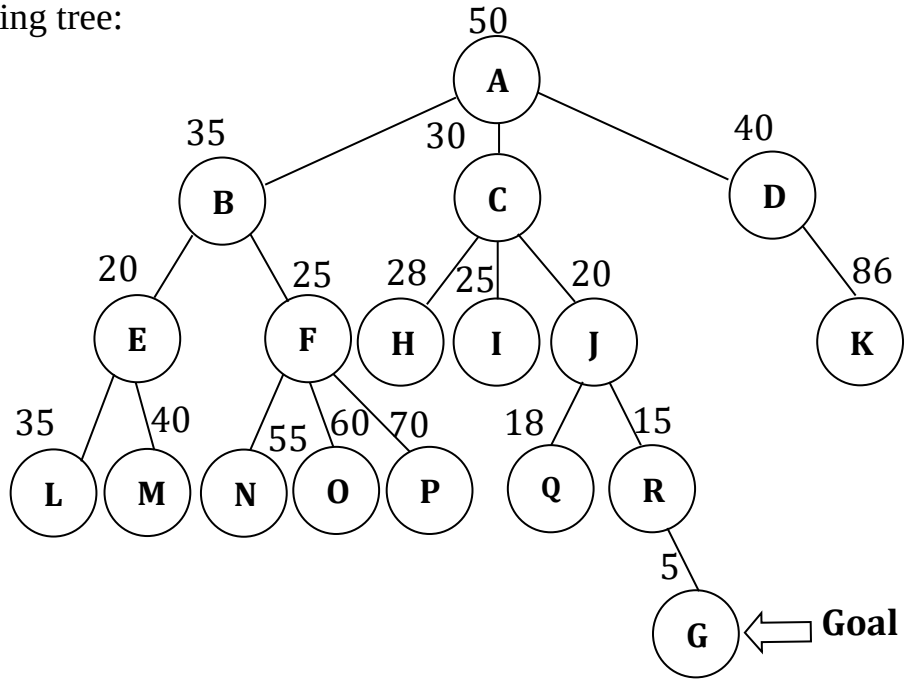
Aggregative search or hill climbing method.

- In this type of search, a new method is relied upon that depends on a specific function placed in the search process, then each node is taken and we study how far it is from the goal.
- That is, the cost is calculated by accumulating the path costs until reaching the goal.
- Also, the resulting nodes are compared with all previous and new previously unbranched nodes.

HCS Algorithm:

```
cs=start, open=[start], stop=false;
path=[start];
while(not stop) do
      begin
           if(cs=goal) then return(path);
           else
              generate children of cs and put them in open;
           if open is empty then stop=true;
           else
              begin
                   x=cs;
                   for each state y in open do
                   begin
                        compute the heuristic value of y, h(y);
                        if(y better than x) then x=y;
                   end
                   if(x better than cs) then cs=x, add cs to path;
                   else
                      stop=true;
              end
      end
end
```

Example 1: Find the goal (G) with a minimum cost by using the HCS method to the following tree:



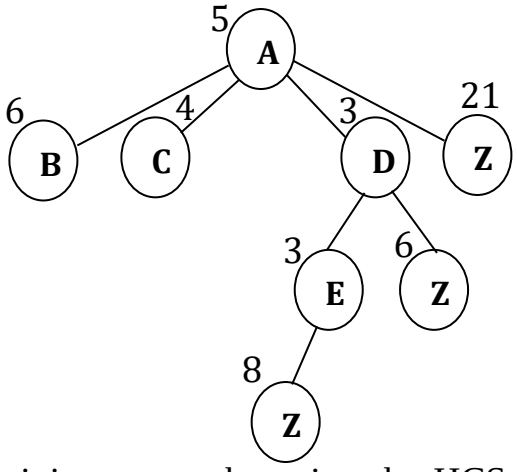
Where: Current State(CS)=A, Open=A, Stop=False, Path=[A].

Solution of 1:

#	CS	Open	Path
0	A	[A]	[A]
1	C We take the lowest value among the children	[B,C,D] (A's children) At each step, it is emptied, then the process of generating children takes place.	[A,C] X=A 50 X=B 35 X=C 30 X=D 40
2	J	[H,I,J] (C's children)	[A,C,J] X=C 30 X=H 28 X=I 25 X=J 20
3	R	[Q,R] (J's children)	[A,C,J,R] X=J 20 X=Q 18 X=R 15
4	G	[G] (R's children)	[A,C,J,R,G] X=R 15 X=G 5 The path that reaches the solution with the lowest cost.

Solution Path: A --- C --- J --- R --- G
Search Space: A, B, C, D, H, I, J, Q, R, G
No. of nodes: 18, No. of links: 17

***H.W.*:** Find the goal (Z) with a minimum cost by using the HCS method to the following tree:

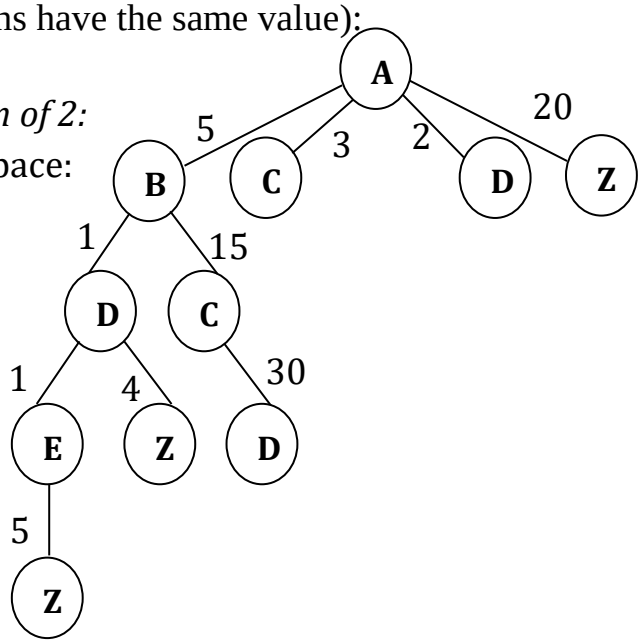


Solution Path: ?
No. of nodes: ?
No. of links: ?

Example 2: Find the goal (Z) with a minimum cost by using the HCS method to the following table, where A is initial state (all paths have the same value):

Node	Cost
$(A \to B)$	5
$(A \to C)$	3
$(A \to D)$	2
$(A \to Z)$	20
$(B \to D)$	1
$(B \to C)$	15
$(C \to D)$	30
$(D \to E)$	1
$(D \to Z)$	4
$(E \to Z)$	5

Solution of 2:
State Space:

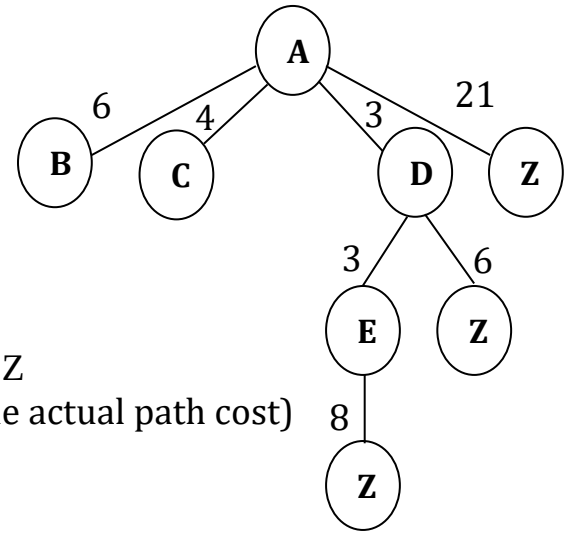


Search Space: $f(n) = g(n) + h(n)$

$$g(1): \begin{cases} f(B) = 5+1 = 6 \\ f(C) = 3+1 = 4 \\ f(D) = 2+1 = 3 \Leftarrow \\ f(Z) = 20+1 = 21 \end{cases}$$

$$g(2): \begin{cases} f(E) = 1+2 = 3 \Leftarrow \\ f(Z) = 4+2 = 6 \end{cases}$$

$$g(3): \quad f(Z) = 5+3 = 8 \Leftarrow$$

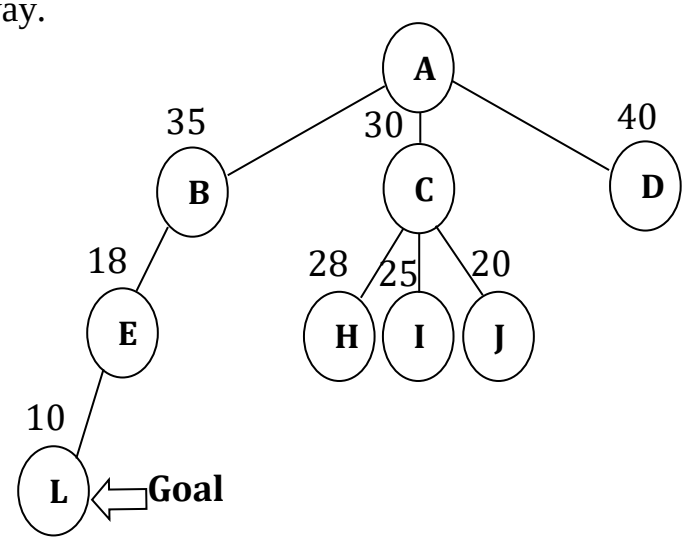


Search Space: A --- B --- C --- D --- Z --- E --- Z --- Z
Solution Path: A --- D --- E --- Z = 2+1+5=8 ⇨ (The actual path cost)
No. of nodes: 8, No. of links: 7, g: 3.

Problems with Hill Climbing Search Method:

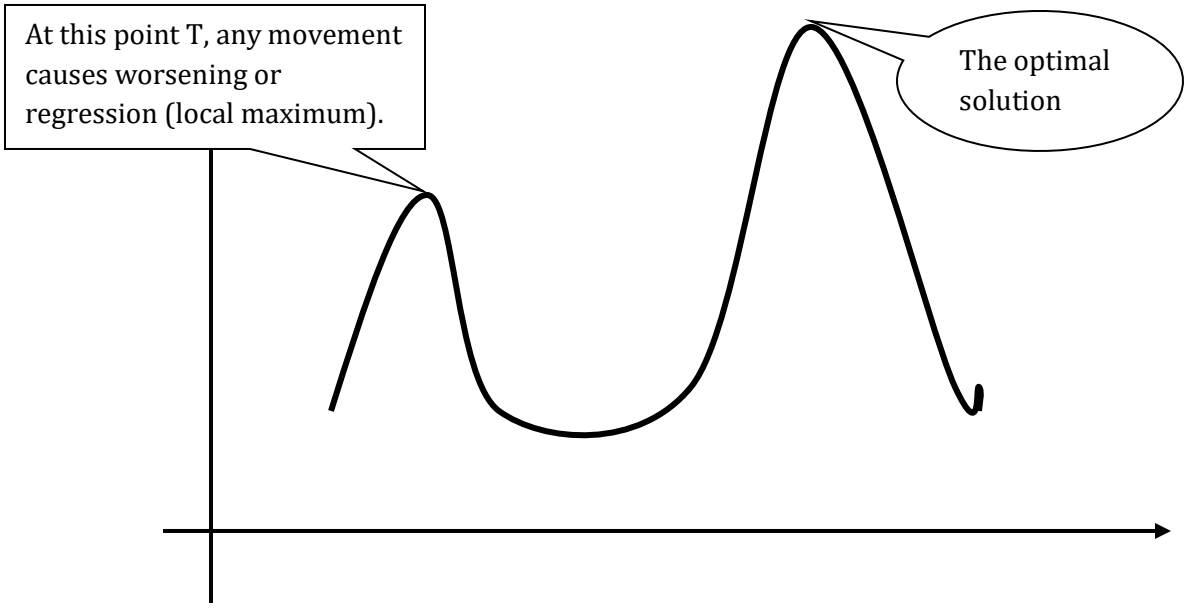
1-Local Maximum: is a state that is better than all its neighbors but it's not better than some other states far away.

Example:

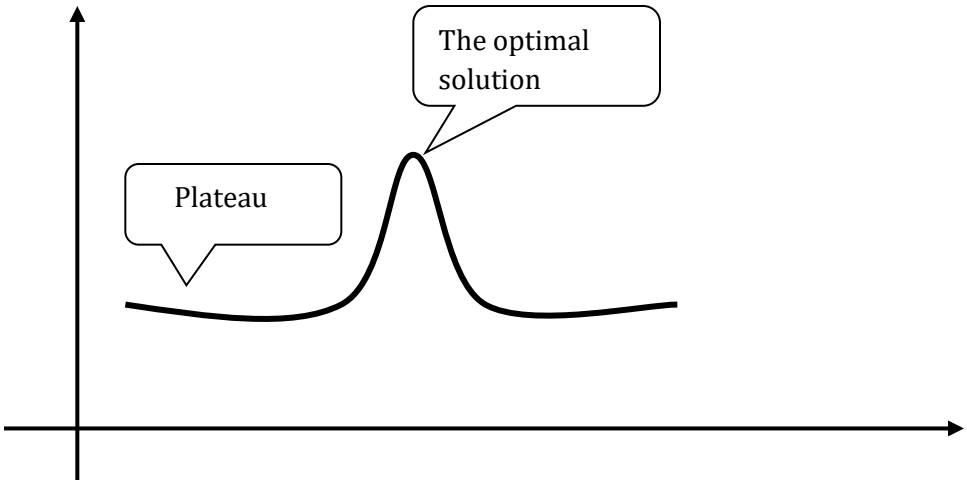


In this tree, we will proceed with the solution towards (C) and then (J), which is a path that does not lead to the goal (L), while path (E) is the one that leads to the goal.

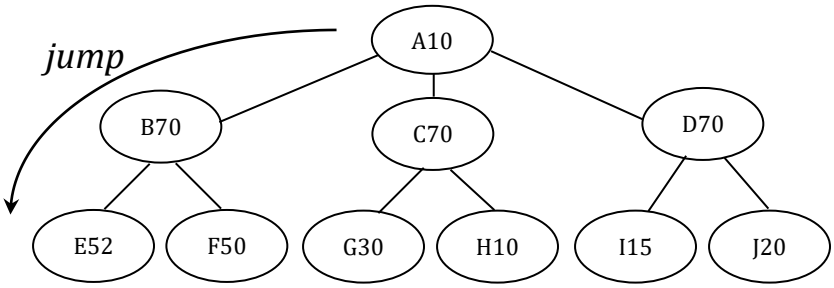
A local maximum moves appear to make things worse. It is mainly frustrating because they often occur almost within sight of the solution. In this case they are called Foothills. In the figure below, when we reach point (T), the method will fail because we will consider it the solution, even though it is not the best solution. Therefore, if we were to continue, we would have to compromise on quality slightly [accept a temporarily worse state] in order to reach the optimal solution.



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

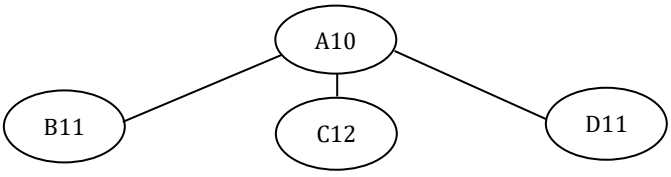


Example:



We notice in this tree that nodes (B, C, D) have similar values, therefore a jump must be made to move to the next level.

3-Ridge: is an area of the search space that is higher than surrounding areas, but that cannot be traversed by single move in any one direction. It is considered a difficult problem because the children's values are greater than the parents', meaning we have a repetition in the execution steps (descent area) which leads to the collapse of the heuristic function, so the algorithm must be repeated, i.e., finding the Heuristic Value.
Example:



We notice in this tree that the children are greater than the parents.

HCS Problems Solutions:

1. To solve the Local Maximum problem, we backtrack to a node prior to the problem and take another path, even though the first path seemed optimal.
2. To overcome the Plateau problem, we make a big jump in some direction to try to get to new section of the search space.
3. To solve the Ridge problem, we apply two or more rules before doing the test.

Note: In this algorithm, it is not necessary that we obtain the goal, meaning that if it fails in the path, it cannot return to choose another path. By adding Backtracking to this method, we will get the Best-First Search method.

2nd: Best-First Search (Best-FS)

- It combines the advantages of DFS & BFS into a single method.
- At each step we select the most promising nodes (meaning the most effective one), this is done by applying an appropriate heuristic function to each of them.
- In this method, we must reach the goal. The main benefit of this algorithm is that it provides room for correction in case of taking a wrong path.

Best-FS Algorithm:

1. We define the State.

2. We calculate the heuristic function for the node. f(n) = h(n)

3. We put this node in List L. If the node is (n), it will be as follows: L=[n].

4. We calculate (n) and its children. If the node (n) has three children, it will be as follows: [f(x1), f(x2), f(x3)].

$L=[f(x_1), f(x_2), f(x_3)]$

5. From these children placed in List L, we take the children of the node that has the lowest value (Minimum). But when there is more than one node with the same value, we choose any of them.

eg. $L=[f(x_1), f(x_2), f(x_3)]$

⇩

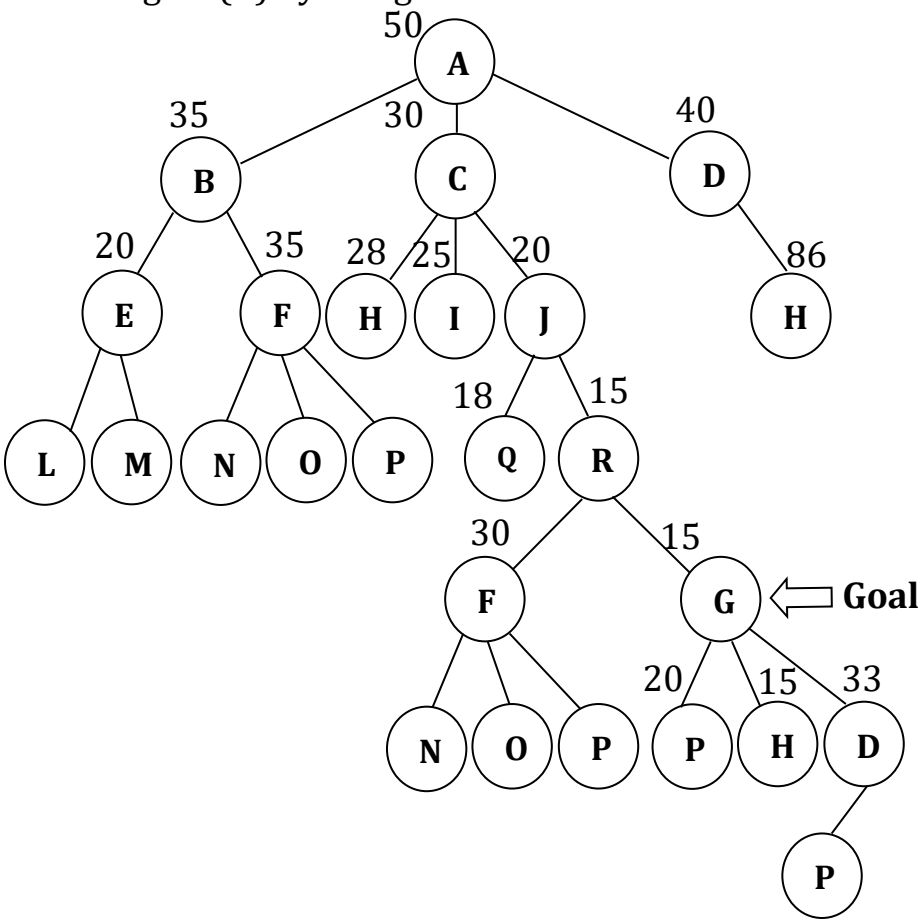
minimum(n)

take its children

$L=[f(x_1), \underline{f(y_1), f(y_2)}, f(x_3)]$

Children of n

Example: Find the goal (G) by using the Best-FS method to the following tree:



Solution:

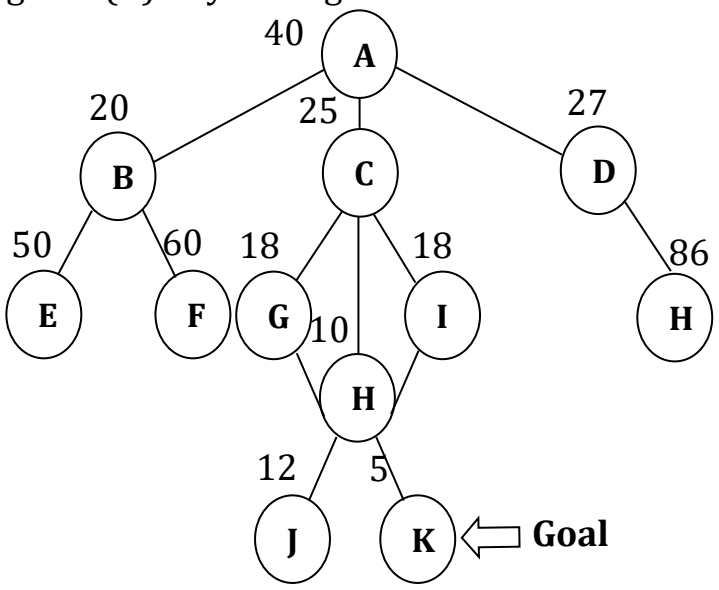
#	Best CS	Open	Cost	Path
0	A	[A]	A: 50	A
1	C	[B, C, D]	C: 30, B: 35, D: 40	A-C
2	J	[B, D, H, I, J]	J: 20, I: 25, H: 28, B: 35, D: 40	A-C-J
3	R	[B, D, H, I, Q, R]	R: 15, Q: 18, I: 25, H: 28, B: 35, D: 40	A-C-J-R
4	G	[B, D, H, I, Q, F, G]	G: 15, Q: 18, I: 25, H: 28, F: 30, B: 35, D: 40	A-C-J-R-G

Note: In the Cost field, we arrange from lowest to highest cost.

Note: When the Goal value is small, the lowest value will be searched for, but when the Goal value is large, the largest value will be searched for.

Note: In the HCS method, the 'Open' list is emptied before children are generated, whereas in this method (Best-FS), the 'Open' list is retained while children are generated. Additionally, nodes can be added from either the right or the left.

***H.W.*:** Find the goal (K) by using the Best-FS method to the following graph:



Best-FS Algorithm to Solve the 8-Puzzle Problem:

Computer representation relies heavily on matrices. When solving a specific problem, we move certain numbers. However, for simplicity and to understand the solution principle of this method, we will deal with it based on the empty space. If the empty space is in the middle, we will have four states. If the empty space is in one of the corners, there will only be two states. But if the empty space is on an edge (in the middle of a side), there will be three states. This is illustrated in the figures below:

1	↓ ← →	3
4	8	5
6	7	2

The edges (3*3)

1	2	3
4	8	5
6	7	↑ ←

*The corners (3*3)*

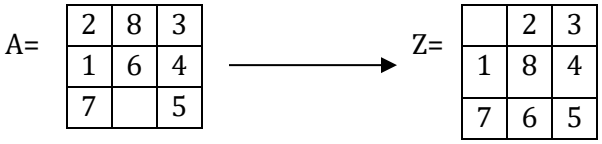
1	2	3
4	↑ ↓ ← →	5
6	7	8

The middle (3*3)*

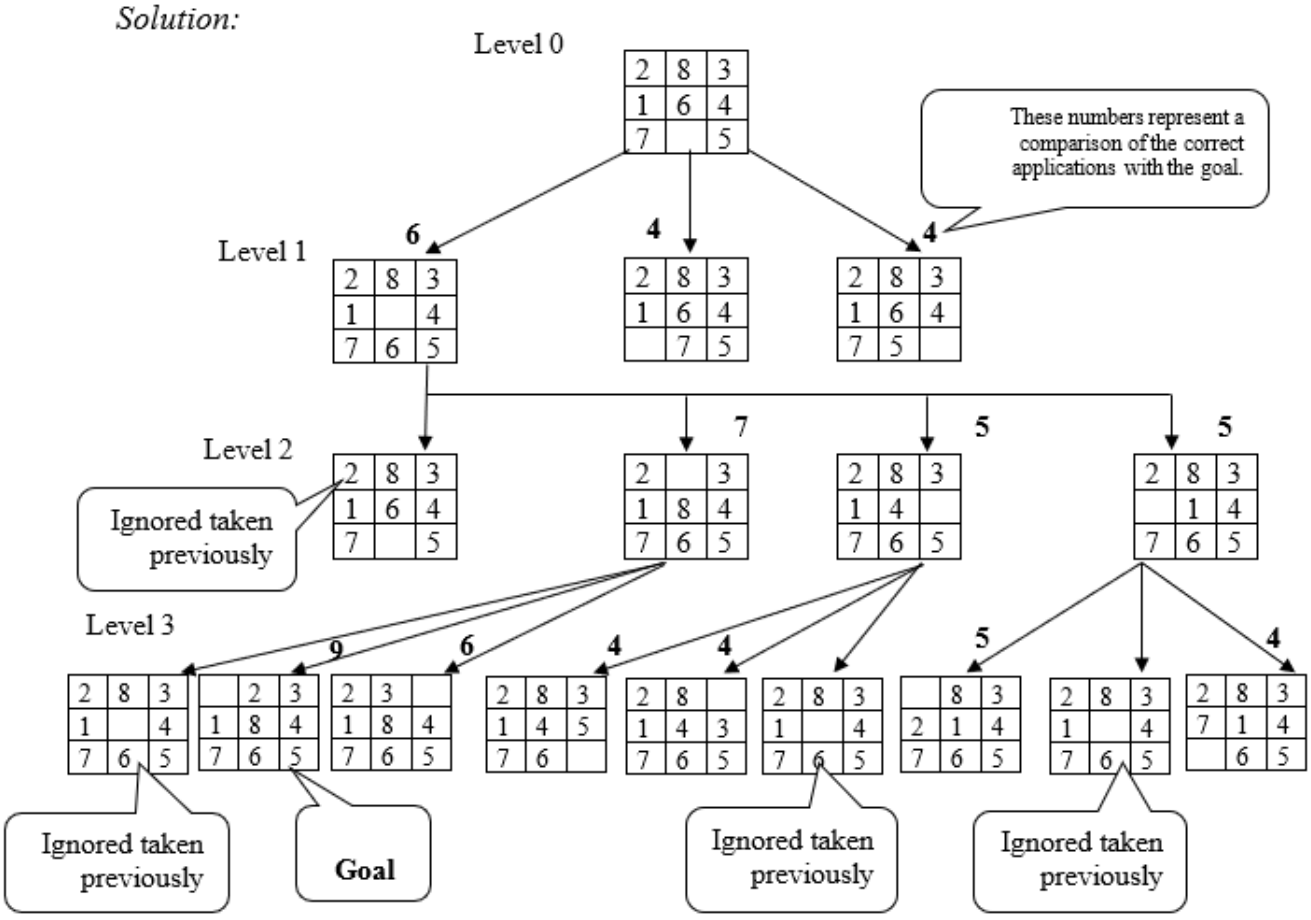
Note: When the movement is specified, we must proceed based on that specified movement. However, when the movement is not specified, we must proceed according to priority based on the following movements:

1. ←
2. ↑
3. →
4. ↓

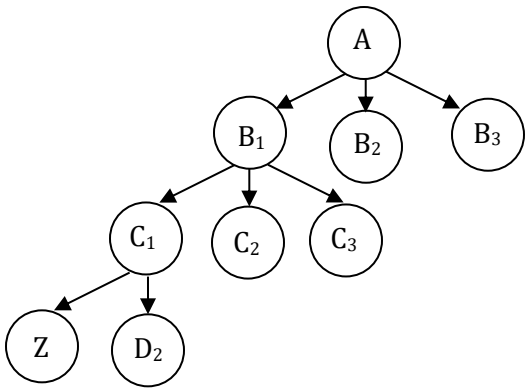
Example: Find the goal (Z) by using the Best-FS method, where (A) is the initial state.



Solution:



ثم يتم تحويل الحل الناتج إلى(Tree):



***H.W.*:** Complete the solution by using (DFS and BFS) methods.

***H.W.*:** Find the goal (Z) by using the Best-FS method, where (A) is the initial state.

A=

1	2	3	4
5	6	7	8
9	10	11	12
13		14	15

→ Z=

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

3rd: A Star Search (A* Search)

-Minimizing the total estimated solution cost. (Pronounced: "A star search").
-This algorithm focuses on searching for the best solutions and the lowest cost. It relies on the concepts of (Best-FS, DFS, and BFS) in its search, and this method is considered one of the approaches that relies most heavily on logical answers to reach the goal.

It evaluates nodes by the following relation: f(n) = g(n) + h(n)

Where:

- g(n): the cost to reach the node.
- h(n): the cost to get from the node to the goal.
- f(n): estimated cost of the cheapest solution through n.

Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n).

We can conclude that this strategy is more than just reasonable: provided that the heuristic function h(n) satisfies certain conditions.

A* search is both complete and optimal, the optimality of(A*) is straightforward to analyze if it is used with Tree-Search.

In this case, A* is optimal if h(n) is an admissible heuristic that is, provided that h(n) never overestimates the cost to reach the goal. Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is.

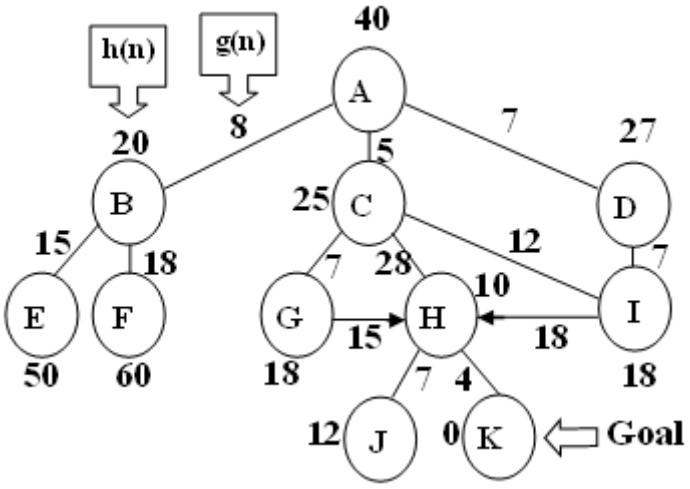
Since g(n) is the exact cost to reach (n) we have as immediate consequence that f(n) never overestimates the true cost of a solution (C*) through (n):

$f(n) = g(n) + h(n) < C^*$.

Note: The path may have a value or there may be values on the paths. It was mentioned in the question that the path value is of no importance, in which case g(n) = 0. If it is mentioned in the question that all paths have the same value, then we substitute g(n) = 1 even if there are different values on the paths.

Note: The A* function is considered Admissible because it finds the shortest path to the goal.

Example 1: Consider the graph bellow, starting with the node(A) to find the goal node(K) with the minimum coast by using the A* search method.



Solution of 1:

$$f(n) = h(n) + g(n)$$ نحسب قيمة الدالة عند كل عقدة:

$f(A) = 40 + 0 = 40$
$f(B) = 20 + 8 = 28$
$f(C) = 25 + 5 = 30$
$f(D) = 27 + 7 = 34$
$f(E) = 50 + (8 + 15) = 73$
$f(F) = 60 + (8 + 18) = 86$
$f(G) = 18 + (5 + 7) = 30$
$f(H) = 10 + (5 + 28) = 43$
$f(I) = 18 + (7 + 7) = 32$
$f(J) = 12 + (5 + 7 + 15 + 7) = 46$
$f(K) = 0 + (5 + 7 + 15 + 4) = 31$

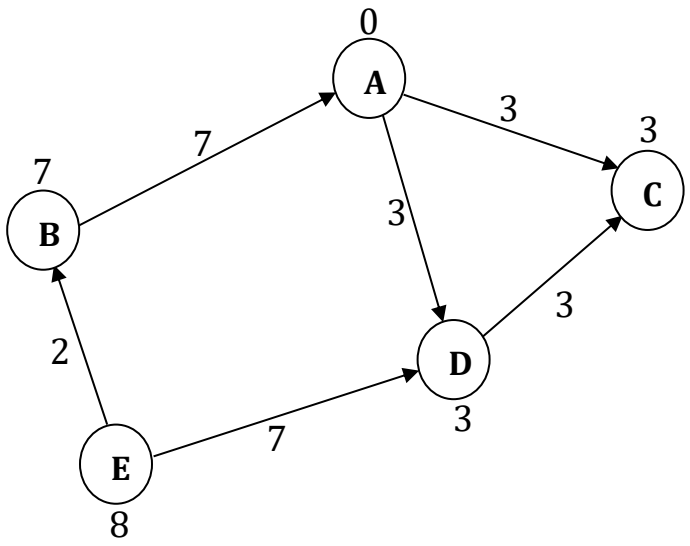
#	State	Open	Closed	Node
0	-	[A40]	[]	-
1	A40	[B28, C30, D34]	[A]	(A,0)
2	B28	[C30, D34, E73, F86]	[A,B]	(B,A)
3	C30	[G30, H43, I35, D34, E73, F86]	[A,B,C]	(C,A)
4	G30	[H37, I35, D34, E73, F86]	[A,B,C,G]	(G,C)
				(H,C,G)
5	D34	[I32, H37, E73, F86]	[A,B,C,G,D]	(D,A)
				(I,C,D)
6	I32	[H37, E73, F86]	[A,B,C,G,D,I]	(I,D)
7	H37	[J46, K31, E73, F86]	[A,B,C,G,D,I,H]	(H,G)
8	K31	[J46, E73, F86] ***Goal is found***	[A,B,C,G,D,I,H,K]	(K,H)

Solution Path: A --- B --- C --- G --- H --- K
Search Space: A --- B --- C --- G --- D --- I --- H --- K
State Space: All the nodes of the graph.

Note: In step (3) above, the old node (H43) was deleted because its new value is (H37), which is smaller than the old one, so the old node (H) was replaced with the new one.

Note: In step (5) above, the old node (I35) was deleted because its new value is (I32), which is smaller than the old one, so the old node (I) was replaced with the new one.
Note: In step (6) above, the old node (H37) was not deleted because its new value is (H42), which is larger than the old one, so the old node (H) was not replaced with the new one.

Example 2: Consider the graph bellow, use the A* search method to find the goal node(A) and its path with minimum cost, where the initial state is(E).



Solution of 2:

$$f(n) = h(n) + g(n)$$ نحسب قيمة الدالة عند كل عقدة:

$f(E) = 8 + 0 = 8$

$f(B) = 7 + 2 = 9$

$f(D) = 3 + 7 = 10$

$f(A) = 0 + (2 + 7) = 9$

#	State	Open	Closed	Node
0	-	[E8]	[]	-
1	E8	[B9, D10]	[E]	(E,0)
2	B9	[A9, D10]	[E,B]	(B,E)
3	A9	[D10] ***Goal is found***	[E,B,A]	(A,B)

Solution Path: E --- B --- A
Search Space: E --- B --- A
State Space: All the nodes of the graph.
***H.W.*:** Find the goal node(C) to the same example(2) above.