Lecture 4

4.1. State space search

problem space: A problem space is represented by directed graph, where nodes represent search state and paths represent the operators applied to change the state.

Problem solving : is a process of generating solutions from observed data.

- a problem is characterized by a set of goals,
- a set of objects, and
- a set of operations.

What is Search?

Search is an important aspect of AI. **Search can be defined as a problem-solving technique that enumerates a problem space from an initial position in search to a goal position (or solution). The manner in which the problem space is searched is defined by the search algorithm or strategy**. As search strategies offer different ways to enumerate the search space. Ideally, the search algorithm selected is one whose characteristics match that of the problem at hand.

4.2. Some common terms in the searching issues:

State: is a representation that an agent can find itself in.

State Space: is a graph whose nodes are the set of all states, and whose links are actions that take the agent from one state into another.

Search Tree: is a tree in which the root node is the start state and has a reachable set of children.

Search Node: is a node in the search tree.

Goal: is a state that the agent is trying to reach.

Action: is something that the agent can choose to do.

Branching Factor: The branching factor in a search tree is the number of actions available to the agent.

4.3. Search algorithms

The most search algorithms is illustrated below. It begins with two types of search - Uninformed and Informed.

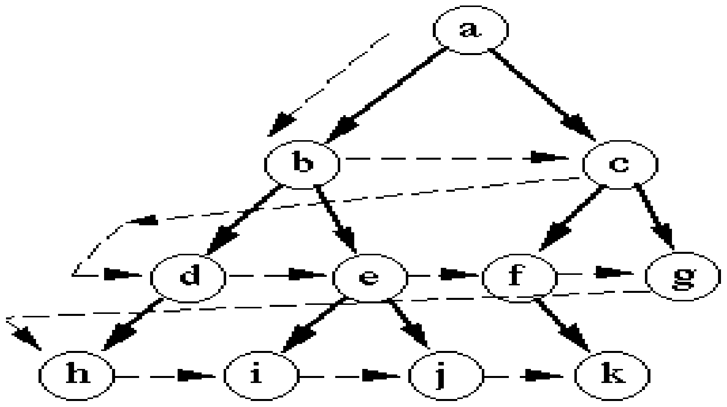
Uninformed Search	Informed Search
Also called blind, exhaustive or brute-force search	Also called heuristic or intelligent search
not efficient, uses no information about the problem to guide the search	Efficient, uses information about the problem to guide the search, usually guesses the distance to a goal state
Comparatively high in cost	Cost is low
May be time consuming	Quick solution to problem
Examples:- - Depth first search (DFS) - Breadth first search (BFS)	Examples:- - Hill climbing algorithm - Best first search - A^*search - Greedy search

4.3.1. Uninformed Search (Blind Search, Brout force)

This type of search takes all nodes of tree in specific order until it reaches to goal

A) Breadth First Search (BFS)

In breadth first search, when a state is examined, all of its siblings are examined before any of its children. The space is searched level-by-level, proceeding all the way across one level before doing down to the next level.



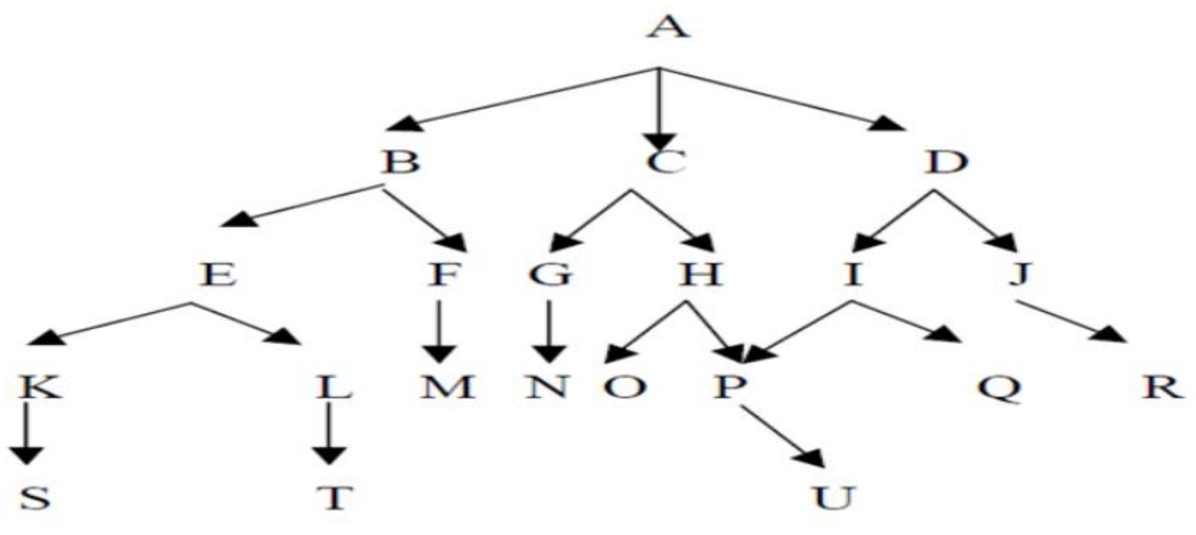
Breadth-first search

Breadth – first – search Algorithm

```
Begin
Open: = [start];
Closed: = [ ];
While open ≠ [ ] do
Begin
Remove left most state from open, call it x;
If x is a goal the return (success)
Else
Begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed;
Put remaining children on right end of open
End
End
Return (failure)
End.
```

<u>Example:</u> trace the following figure to find the node (U).

So the Goal is: U



Breadth – first – search

1 – Open = [A]; closed = [ ].

2 – Open = [B, C, D]; closed = [A].

3 – Open = [C, D, E, F]; closed = [B, A].

4 – Open = [D, E, F, G, H]; closed = [C, B, A].

5 – Open = [E, F, G, H, I, J]; closed = [D, C, B, A].

6 – Open = [F, G, H, I, J, K, L]; closed = [E, D, C, B, A].

7 – Open = [G, H, I, J, K, L, M]; closed = [F, E, D, C, B, A].

8 – Open = [H, I, J, K, L, M, N,]; closed = [G, F, E, D, C, B, A].

9 – and so on until either U is found or open = [ ].

Advantages:-

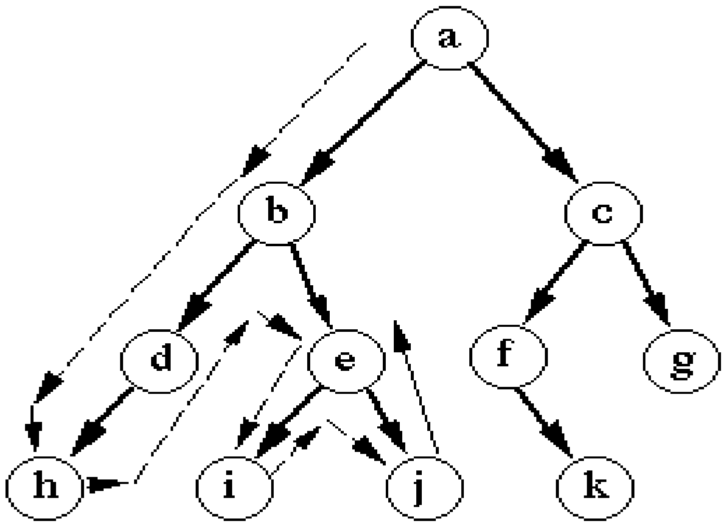
- BFS guarantee to find a solution if exist.
- BFS is efficient where the answer is near the top of the tree and work well.
- If multiple solutions exist, BFS will provide minimal solution which requires the least number of steps.

Disadvantages:-

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

B) Depth – first – search

In depth – first – search, when a state is examined, all of its children and their descendants are examined before any of its siblings. Depth – first search goes deeper in to the search space whenever this is possible.

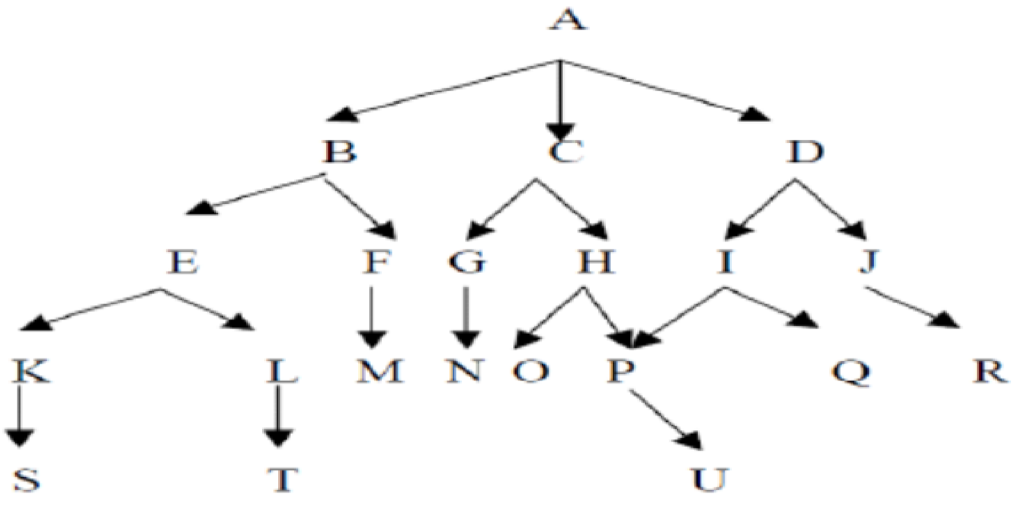


Depth-first search

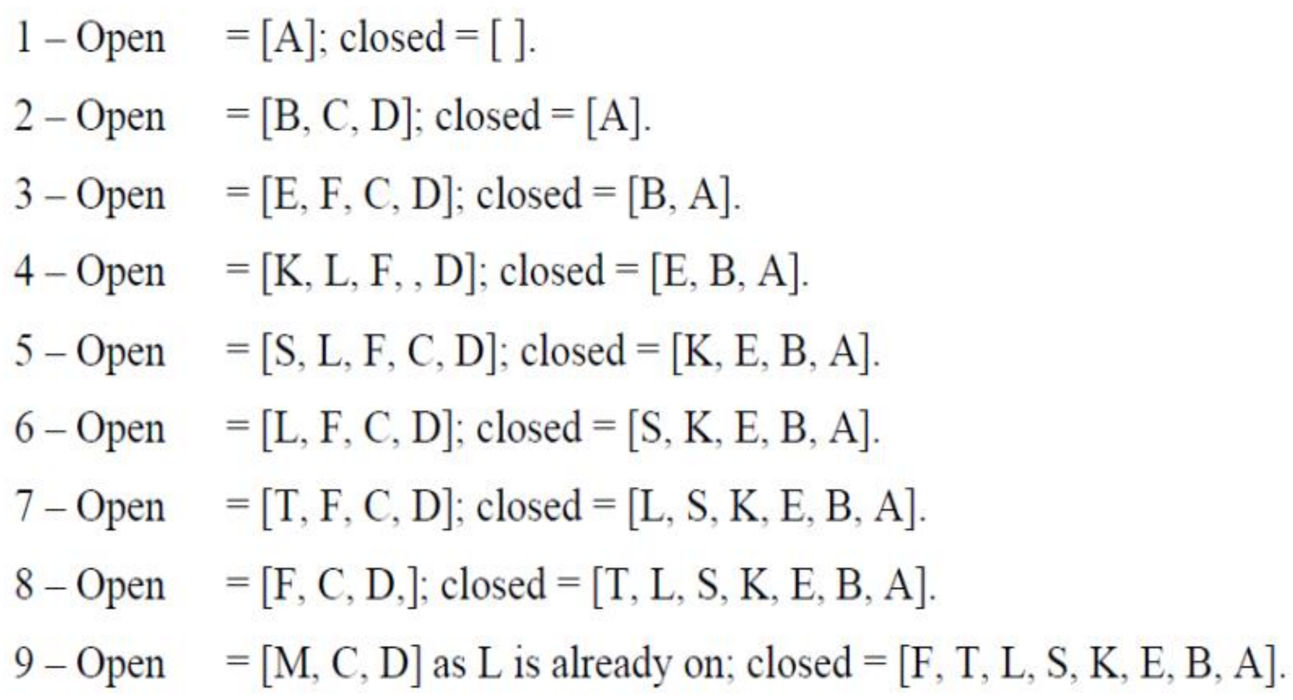
Depth – first – search Algorithm

```
Begin
Open: = [start];
Closed: = [ ];
While open ≠ [ ] do
Remove leftmost state from open, call it x;
If x is a goal then return (success)
Else begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed; put remaining children on
left end of open end
End;
Return (failure)
End.
```

Example: trace the following figure to find the node (U). So the Goal is: U



Depth first search



1 – Open = [A]; closed = [ ].

2 – Open = [B, C, D]; closed = [A].

3 – Open = [E, F, C, D]; closed = [B, A].

4 – Open = [K, L, F, , D]; closed = [E, B, A].

5 – Open = [S, L, F, C, D]; closed = [K, E, B, A].

6 – Open = [L, F, C, D]; closed = [S, K, E, B, A].

7 – Open = [T, F, C, D]; closed = [L, S, K, E, B, A].

8 – Open = [F, C, D,]; closed = [T, L, S, K, E, B, A].

9 – Open = [M, C, D] as L is already on; closed = [F, T, L, S, K, E, B, A].

Advantages:-

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).
- DFS may find a solution without examining much of the search space at all. DFS can stop when one of them is found.

Disadvantages:-

- There is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.
- If multiple solutions exist, minimal solution is not found very easily.

Tikrit University
College of computer science and mathematics
Department of computer science

Artificial Intelligence

Lecture three

Topics that must be covered in this lecture:

- Heuristic search (Hill-climbing).
- Heuristic function.
- Methods of Heuristic search (Best-First-Search).

Informed Search (Heuristic Search):

A heuristic is a method that might not always find the best solution but is guaranteed to find a good solution in reasonable time. By sacrificing completeness it increases efficiency. Heuristic search is useful in solving problems which:-

• Could not be solved any other way.

• Solution takes an infinite time or very long time to compute.

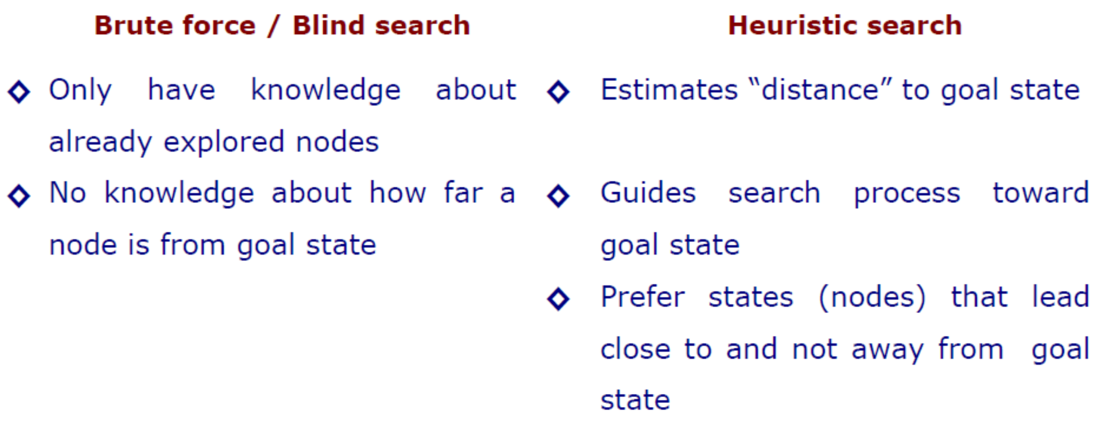
• Heuristic search methods generate and test algorithms, from these methods are:-

1- Hill Climbing.

2- Best-first search (generic best-first, Dijkstra’s algorithm, A*, admissibility of A*).

Heuristic Search compared with other search:

The Heuristic search is compared with Brute force or Blind search techniques



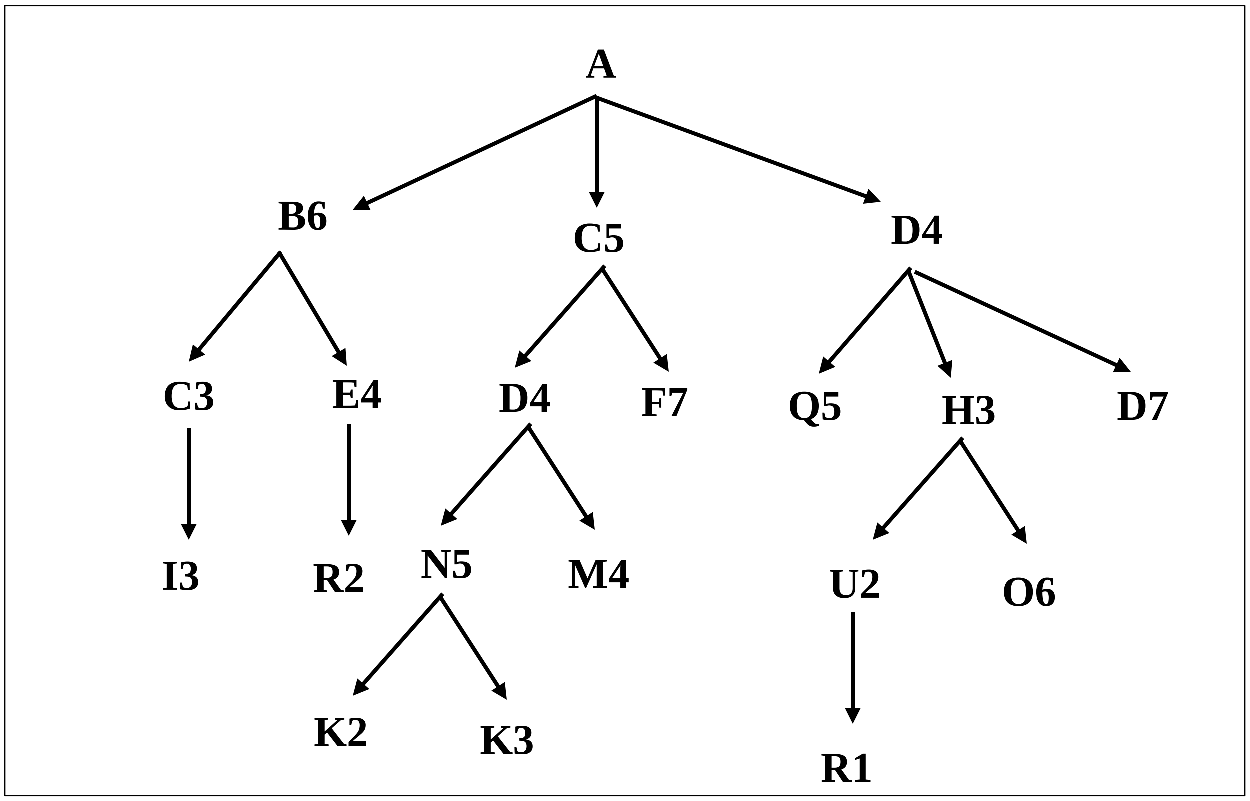
Brute force / Blind search	Heuristic search
◇ Only have knowledge about already explored nodes	◇ Estimates "distance" to goal state
◇ No knowledge about how far a node is from goal state	◇ Guides search process toward goal state
	◇ Prefer states (nodes) that lead close to and not away from goal state

1) Hill Climbing

The idea here is that, you don't keep the big list of states around. You just keep track of the one state you are considering, and the path that got you there from the initial state. At every state you choose the state leads you closer to the goal (according to the heuristic estimate), and continue from there.

The name "Hill Climbing" comes from the idea that you are trying to find the top of a hill, and you go in the direction that is up from wherever you are. This technique often works, but since it only uses local information.

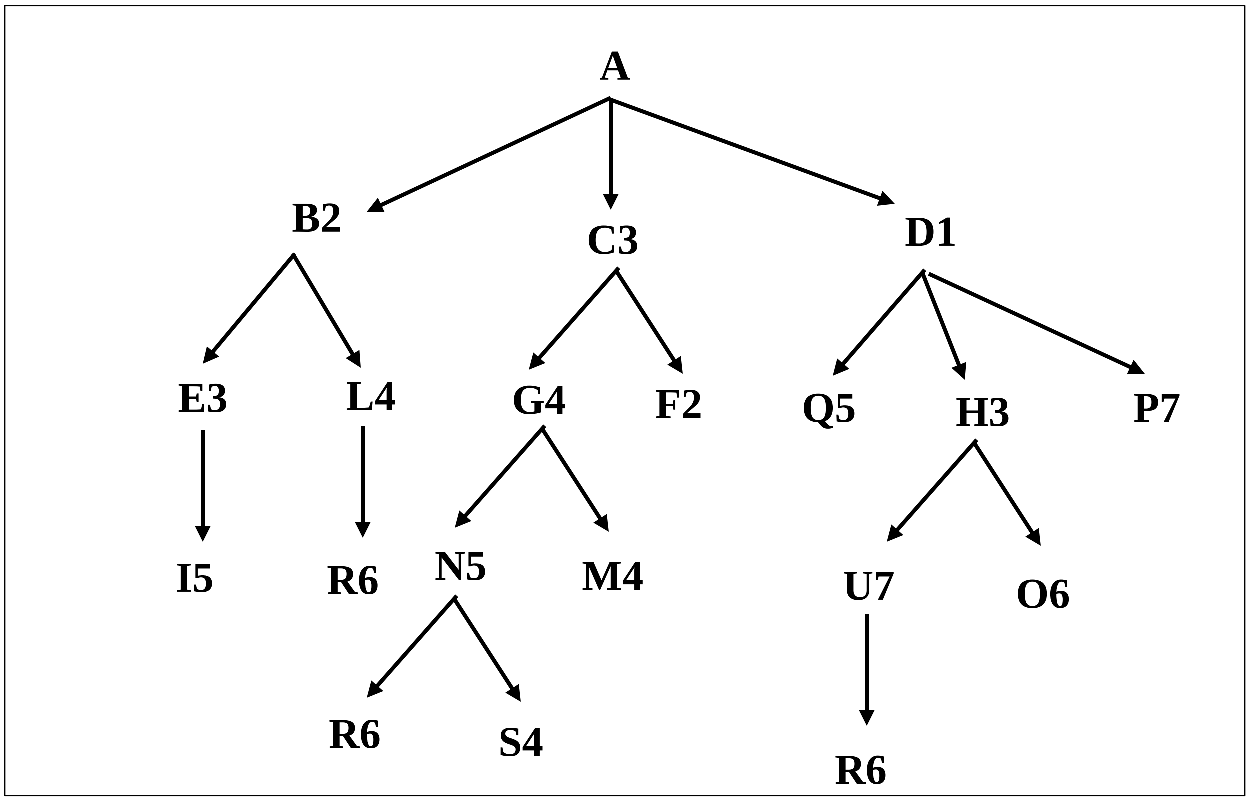
<u>Example:</u> **search for (R) with local minima from A.**



[OPEN]	[CLOSED]	X
[A]	[]	A
[D4,C5,B6]	[A]	D4
[H3,Q5,P7]	[D4,A]	H3
[U2,Q6]	[H3,D4,A]	U2
[R1]	[U2,H3,D4,A]	R1
[]	[R1,U2,H3,D4,A]	

Final path: A→ D4→ H3→ U2→ R1

<u>Example:</u> search for (R) with local minima from A.



نقطة الصعود الى اعلى التلة

[OPEN]	[CLOSED]	X
[A]	[]	A
[C3,B2,D1]	[A]	C3
[G4,F2]	[C3,A]	G4
[N5,M4]	[G4,C3,A]	N5
[R6,S4]	[N5,G4,C3,A]	R6
[S4]	[R6, N5,G4,C3,A]	

Final path: A→ C3→ G4→ N5→ R6

ملاحظات:-

ملاحظة 1:- في خوارزمية ال (heuristic) أي (node) داخل ال (search tree) لا تحتوي على (heuristic) عندها يكون ال (heuristic) لتلك (node) داخل ال (search tree) يساوي صفر.

ملاحظة 2:- في خوارزمية ال (Hill climbing) تحديداً ال (heuristic) الخاص ب (start state) غير مهم لكون ال (start state) تعتبر النقطة التي يتم الوقوف عليها للصعود الى قمة التل.

ملاحظة 3:- لا تحتوي خوارزمية (Hill climbing) على ذاكرة والسبب عدم قدرة خوارزمية ال (Hill climbing) على الاحتفاظ ب (nodes) ال (levels) السابقة.

ملاحظة 4:- لا تحتوي خوارزمية (Hill climbing) على ال (back-tracking) والسبب عدم قدرة خوارزمية ال (Hill climbing) على الاحتفاظ ب (nodes) السابقة لغرض الرجوع اليهم ومعالجتهم ولنفس السبب تعتبر هذه الخوارزمية سريعة الوصول الى الهدف.

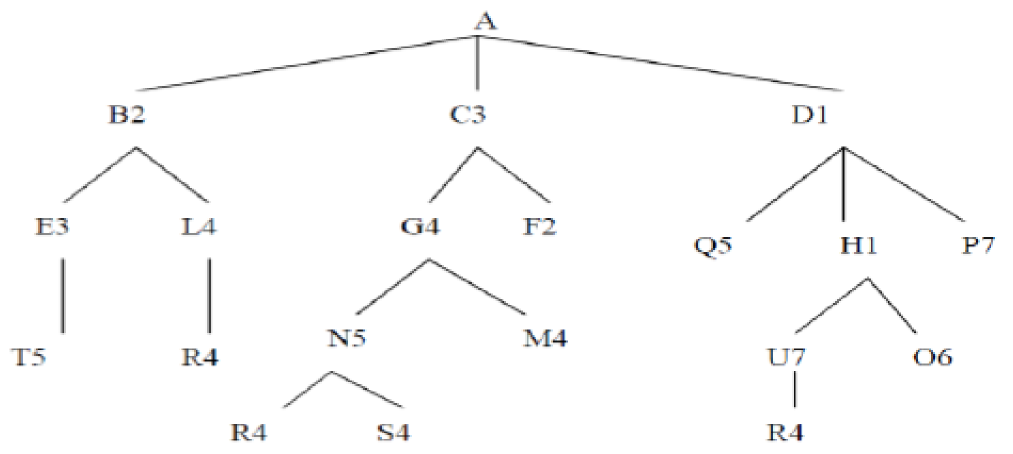
ملاحظة 5:- نفترض ان افضل قيمة هي الأصغر، وفي مشاكل أخرى قد نفترض ان افضل قيمة هي الأكبر.

Hill Climbing Algorithm

```
Begin
Cs=start state;
Open=[start];
Stop=false;
Path=[start];
While (not stop) do
{
    if (cs=goal) then
        return (path);
    generate all children of cs and put it into open
    if (open=[]) then
        stop=true
    else
    {
        x:= cs;
        for each state in open do
        {
            [illegible]
            compute the heuristic value of y (h(y));
            if y is better than x then
                x=y
        }
        if x is better than cs then
            cs=x
        else
            stop =true;
    }
}
return failure;
}
```

For Example:

Searches for R4



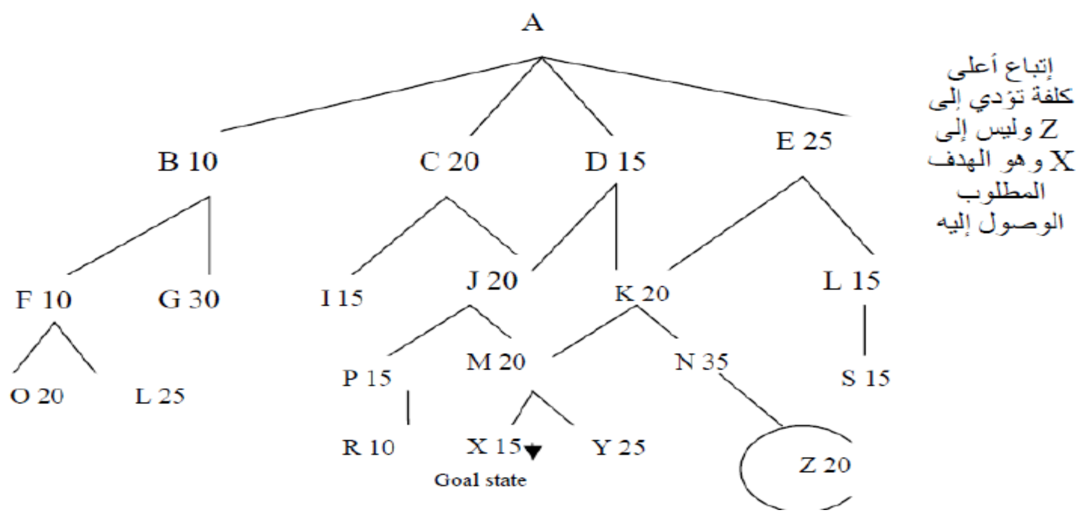
Open	Close	X
A	_	A
C3 B2 D1	A	C3
G4 F2	C3 A	G4
N5 M4	G4 C3 A	N5
R4 S4	N5 G4 C3 A	R4

Optimal path (A $\rightarrow$ C3 $\rightarrow$ G4 $\rightarrow$ N5 $\rightarrow$ R4)

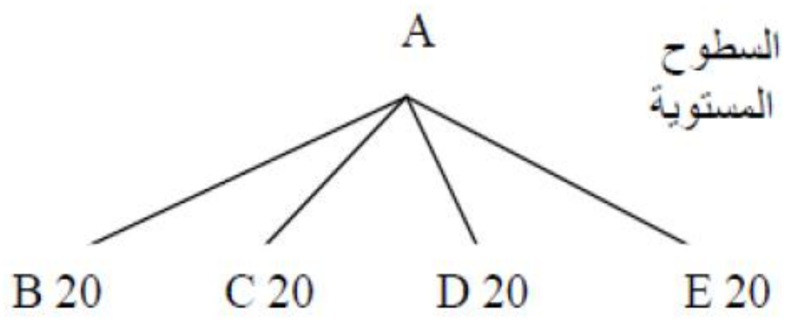
Hill climbing Problems:

Hill climbing may fail due to one or more of the following reasons:-

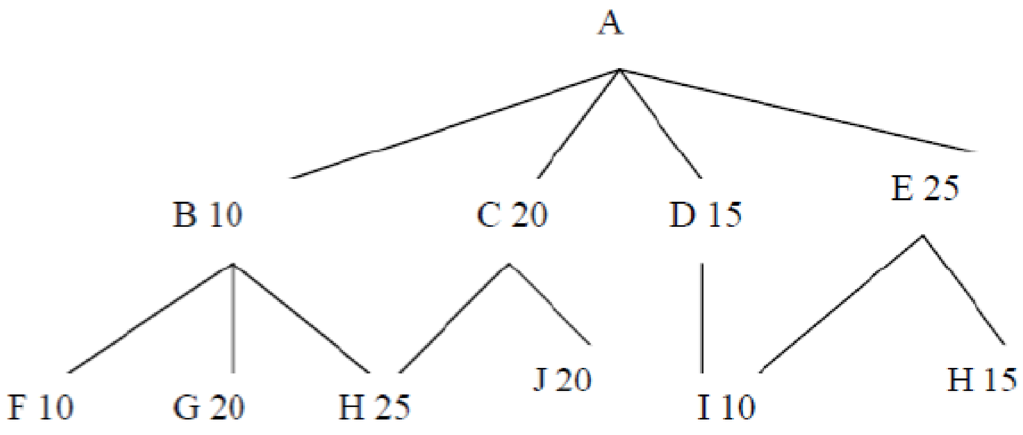
1- **A local maxima**: Is a state that is better than all of its neighbors but is not better than some other states.



2- **A Plateau:** Is a flat area of the search space in which a number of states have the same best value, on plateau it's not possible to determine the best direction in which to move.



3- **A ridge:** Is an area of the search space that is higher than surrounding areas, but that cannot be traversed by a single move in any one direction.



سلسلة الانخفاضات و الارتفاعات

Heuristic function:

A heuristic function is a function f(n) that gives an estimation on the "cost" of getting from node n to the goal state – so that the node with the least cost among all possible choices can be selected for expansion first.

Three approaches to defining f:

f measures the value of the current state (its "goodness")

f measures the estimated cost of getting to the goal from the current state:

f(n) = h(n) where h(n) = an estimate of the cost to get from n to a goal

f measures the estimated cost of getting to the goal state from the current state and the cost of the existing path to it. Often, in this case, we decompose f:

f(n) = g(n) + h(n) where *g(n)* = the cost to get to *n* (from initial state).

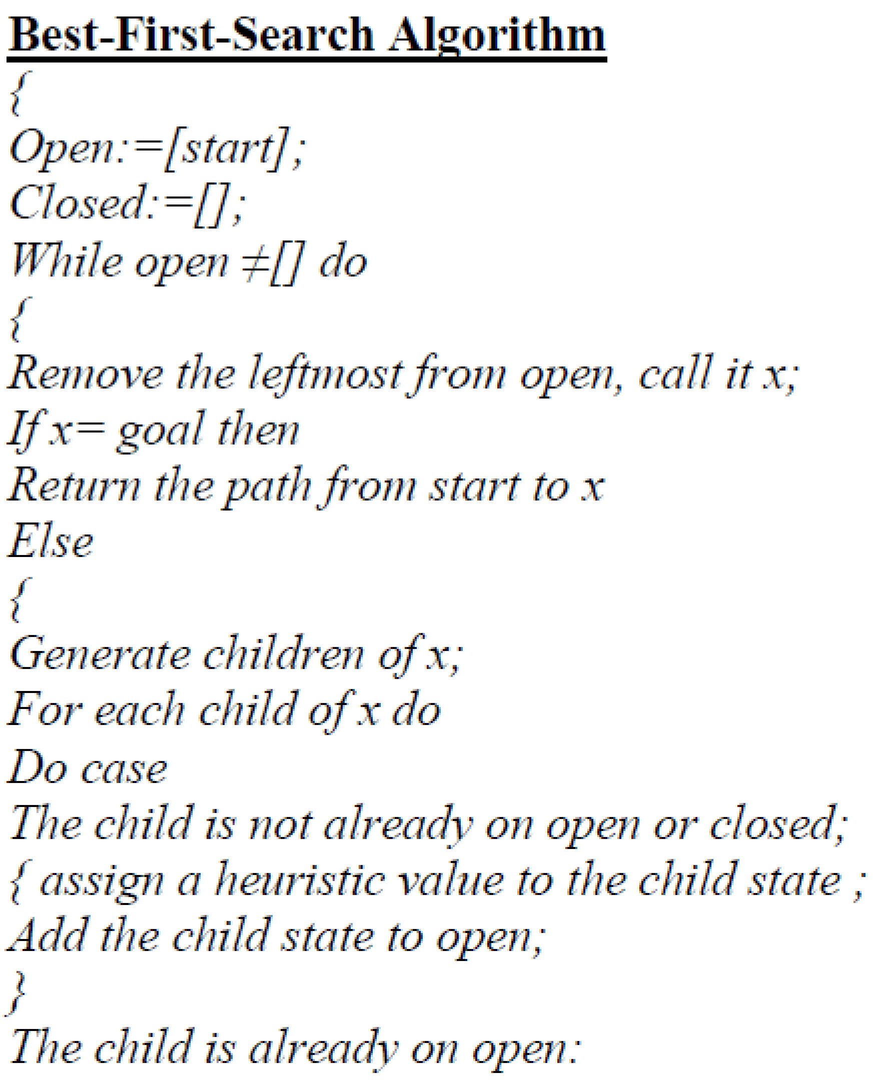
Methods of Heuristic search:

2-Best-First-Search

Best-First-search is a way of combining the advantages of both depth-first and breadth-first search into a single method.

The actual operation of the algorithm is very simple. It proceeds in steps, expanding one node at each step, until it generates a node that corresponds to a goal state. At each step, it picks the most promising of the nodes that have so far been generated but not expanded. It generates the successors of the chosen node, applies the heuristic function to them, and adds them to the list of open nodes, after checking to see if any of them have been generated before. By doing this check, we can guarantee that each node only appears once in the graph, although many nodes may point to it as a successors. Then the next step begins.

In Best-First search, the search space is evaluated according to a heuristic function. Nodes yet to be evaluated are kept on an OPEN list and those that have already been evaluated are stored on a CLOSED list. The OPEN list is represented as a priority queue, such that unvisited nodes can be queued in order of their evaluation function. The evaluation function f(n) is made from only the heuristic function (h(n)) as: f (n) = h(n) .



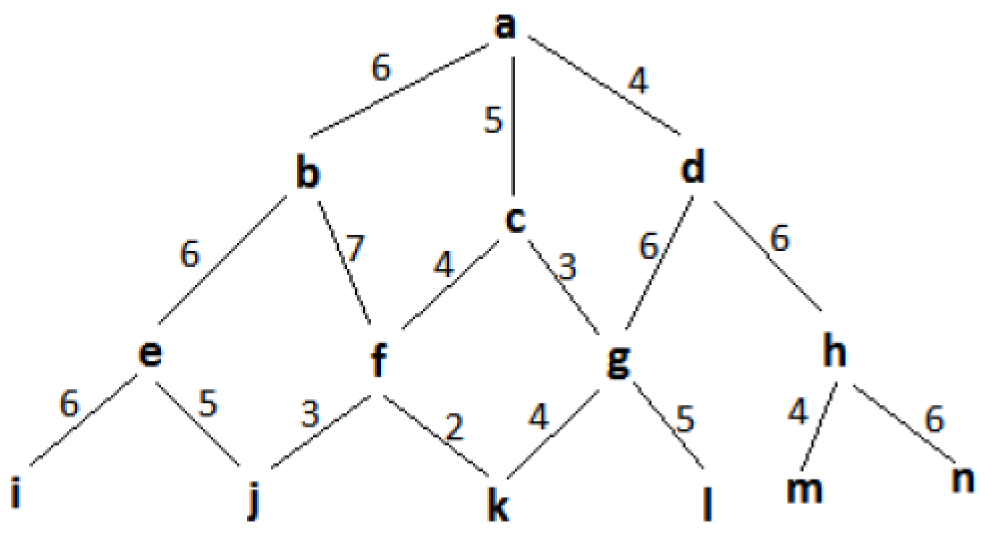
Best-First-Search Algorithm

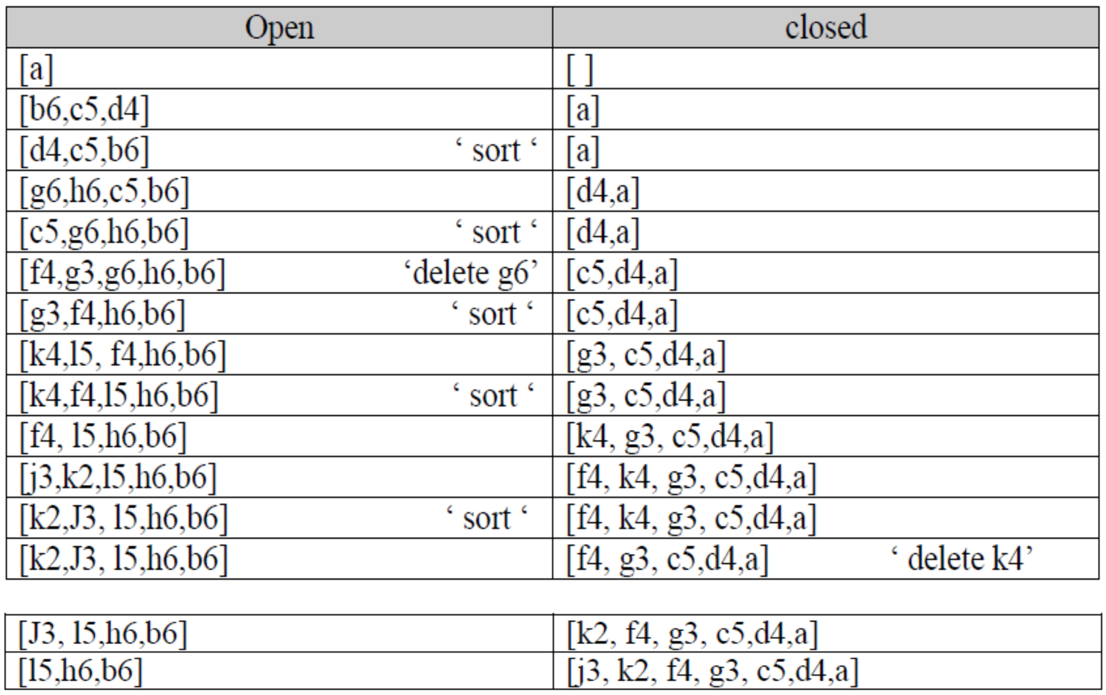
```
{
Open:=[start];
Closed:=[];
While open ≠[] do
{
Remove the leftmost from open, call it x;
If x= goal then
Return the path from start to x
Else
{
Generate children of x;
For each child of x do
Do case
The child is not already on open or closed;
{ assign a heuristic value to the child state ;
Add the child state to open;
}
The child is already on open:
```

```
If the child was reached along a shorter path than the state currently on open then
give the state on open this shorter path value.
The child is already on closed:
If the child was reached along a shorter path than the state currently on open then
{
Give the state on closed this shorter path value
Move this state from closed to open
}
}
Put x on closed;
Re-order state on open according to heuristic (best value first)
}
Return (failure);
}
```

For Example

Searches for l





Open	closed
[a]	[ ]
[b6,c5,d4]	[a]
[d4,c5,b6] ‘ sort ‘	[a]
[g6,h6,c5,b6]	[d4,a]
[c5,g6,h6,b6] ‘ sort ‘	[d4,a]
[f4,g3,g6,h6,b6] ‘delete g6’	[c5,d4,a]
[g3,f4,h6,b6] ‘ sort ‘	[c5,d4,a]
[k4,l5, f4,h6,b6]	[g3, c5,d4,a]
[k4,f4,l5,h6,b6] ‘ sort ‘	[g3, c5,d4,a]
[f4, l5,h6,b6]	[k4, g3, c5,d4,a]
[j3,k2,l5,h6,b6]	[f4, k4, g3, c5,d4,a]
[k2,J3, l5,h6,b6] ‘ sort ‘	[f4, k4, g3, c5,d4,a]
[k2,J3, l5,h6,b6]	[f4, g3, c5,d4,a] ‘ delete k4’
[J3, l5,h6,b6]	[k2, f4, g3, c5,d4,a]
[l5,h6,b6]	[j3, k2, f4, g3, c5,d4,a]

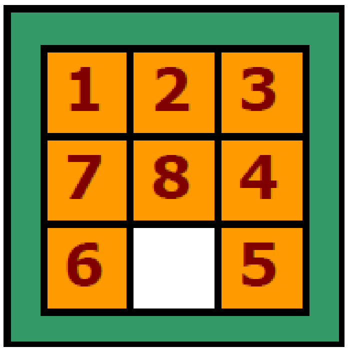
Optimal path (a → d4 → c5 → g3 → f4 → k2 → j3 → l5)

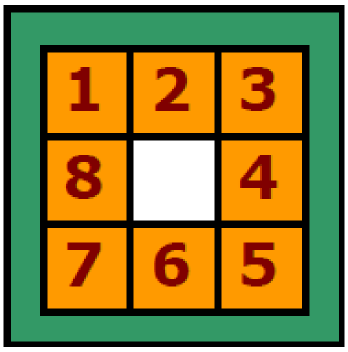
Example : 8 – Puzzle:

◊ State space: Configuration of 8- tiles on the board

◊ state **Initial**: any configuration

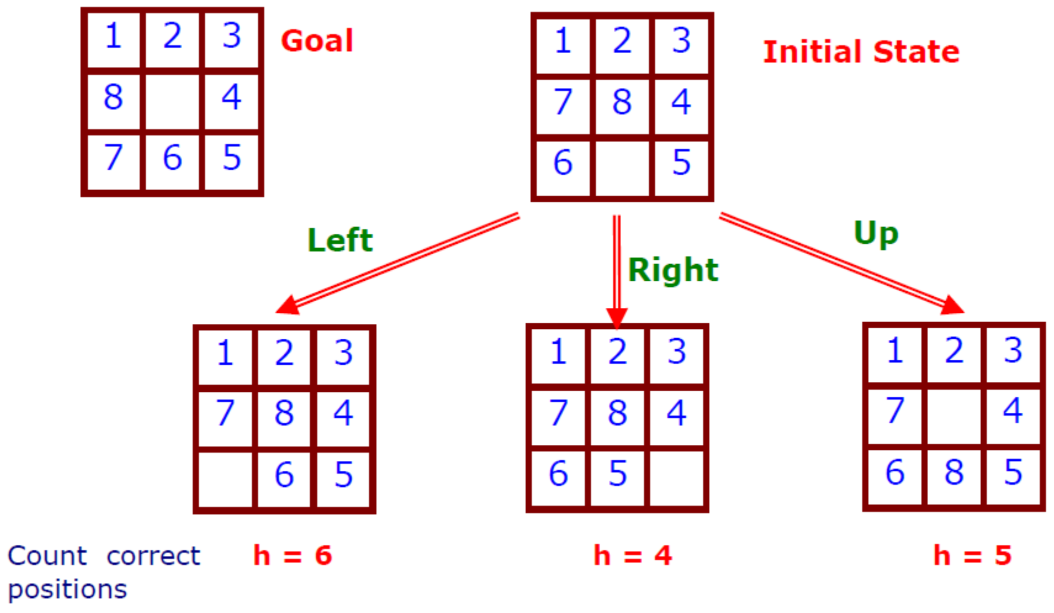
◊ **Goal**: tiles in a specific order





◊ **Actions**

Figure below shows: three possible moves - left , up, right

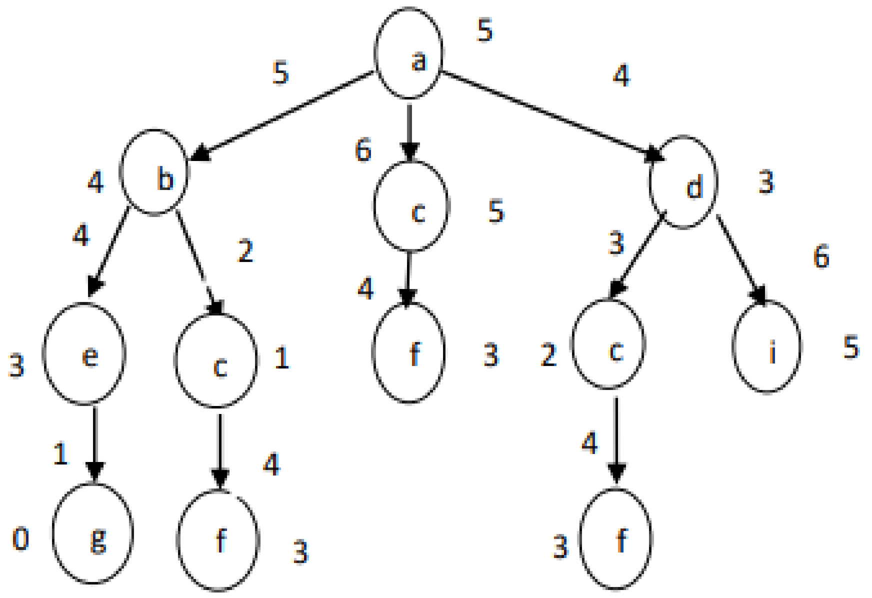


Apply the Heuristic : Three different approaches

- Count correct position of each tile, compare to goal state
- Count incorrect position of each tile, compare to goal state
- Count how far away each tile is from it is correct position.

Approaches	Left	Right	Up
1. Count correct position	6	4	5
2. Count incorrect position	2	4	3
3. Count how far away	2	4	4

Example:

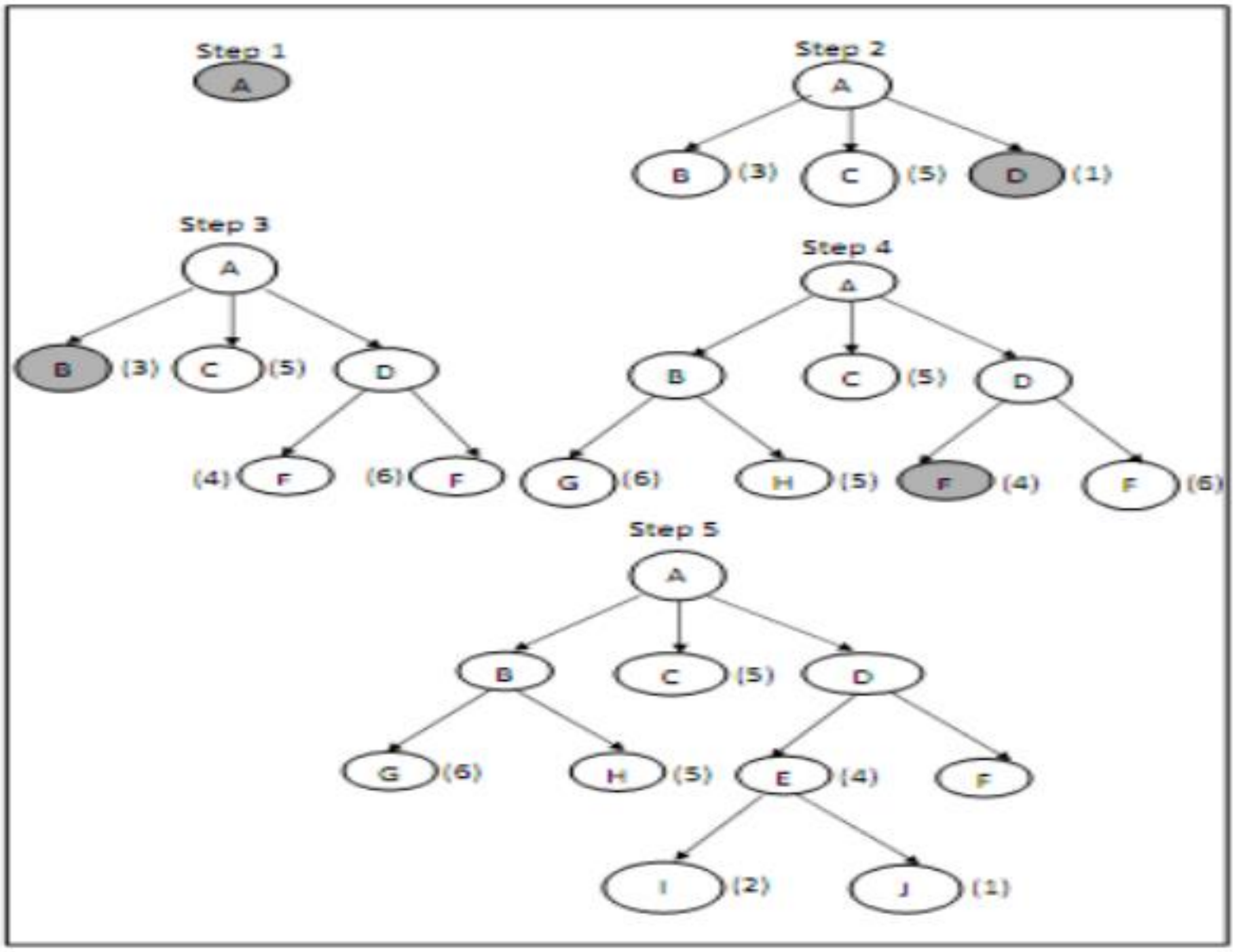


Open	Closed
[A5]	[]
[D3,B4,C5]	[A5]
[C2,B4,I5]	[A5,D3]
[F3,B4,I5]	[A5,D3,C2]
[B4,I5]	[A5,D3,C2,F3]
[C1,E3,I5]	[A5,D3,C2,F3,B4]
[E3,I5]	[A5,D3,F3,B4,C1]
[G0,I5]	[A5,D3,F3,B4,C1,E3]
	[A5,D3,F3,B4,C1,E3,G0]

The goal is found &the resulted path is:

A0 → D4 → F7 → B 16 → C 2 → E 6 → G1 =36

The figure bellow shows the steps of the best first search algorithm on a given tree as an assumption search space.



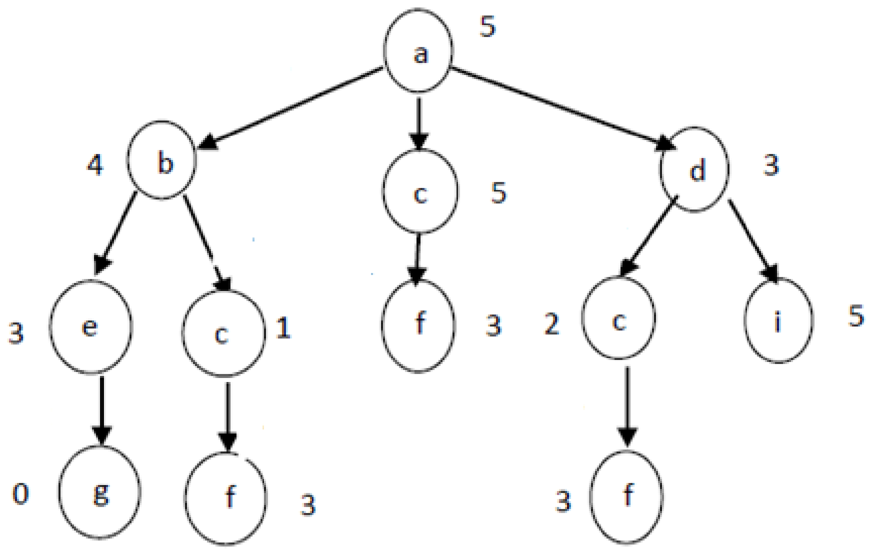
Lecture four

Topics that must be covered in this lecture:

- Another example about best first algorithm.
- Methods of Heuristic search (A- algorithm).
- Complex Search Space and problem solving Approach (in 8-puzzle problem).

Another example about best first algorithm:

Consider the directed graph shown below where the numbers at the states are heuristic estimates (h(n)). Note that the arcs are directed. Use **best first algorithm** to find goal where: A be the start state and G be the goal state.



Solution:

open	closed
[A5]	[]
[D3,B4,C5]	[A5]
[C2,B4,I5]	[D3,A5]
[F3,B4,I5]	[C2,D3,A5]
[B4,I5]	[F3,C2, D3,A5]
[C1,E3,I5]	[B4, F3,C2, D3,A5]
[E3,I5]	[C1, B4, F3, D3,A5]
[G0,I5]	[E3, C1, B4, F3, D3,A5]
	[G0, E3, C1, B4, F3, D3,A5]

3- A - Search algorithm

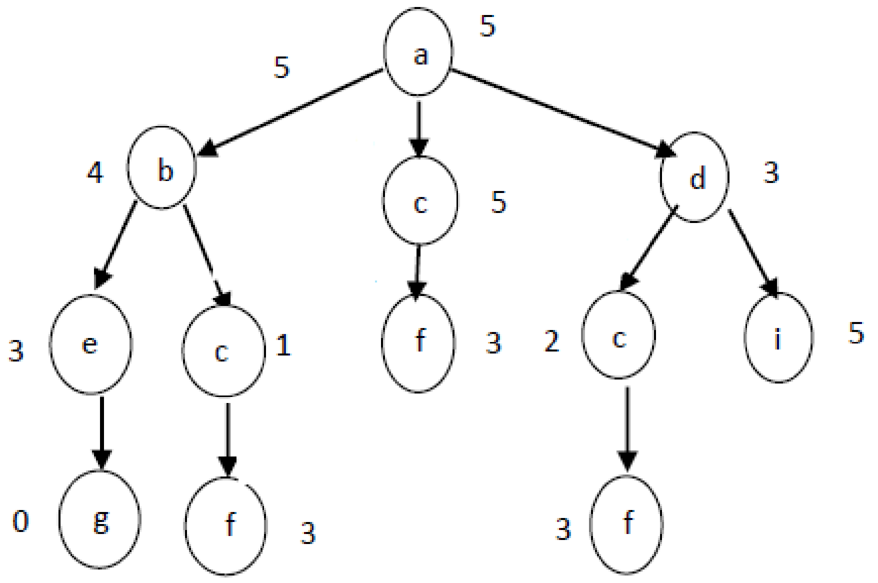
A algorithm is simply define as a best first search plus specific function. This specific function represent the actual distance (levels) between the initial state and the current state and is denoted by g(n). A notice will be mentioned here that the same steps that are used in the best first search are used in an A algorithm but in addition to the g (n) as follow;

$$F(n) = h(n) + g(n) \text{ , where:}$$

h(n):- is a heuristic estimate of the distance from state n to the goal.
g(n):- Measures the actual length of path from any state (n) to the start state.

example about A-algorithm:

Consider the directed graph shown below where the numbers on the links are link costs and the numbers at the states are heuristic estimates (h(n)). Note that the arcs are directed. Use **A-algorithm** to find goal where: A be the start state and G be the goal state.

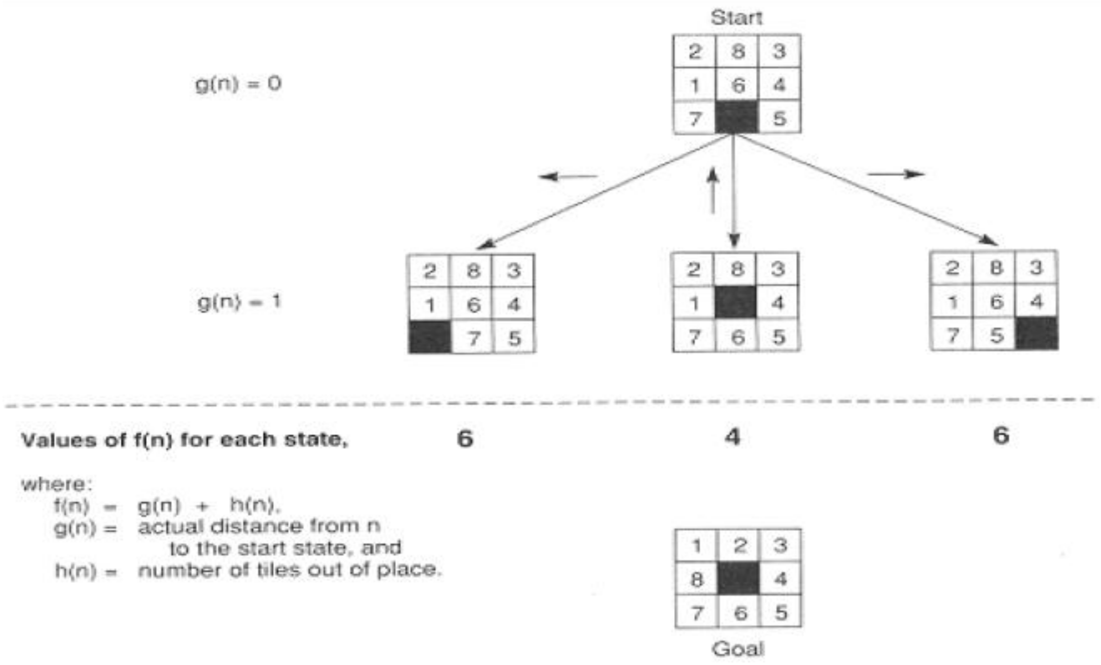


Solution:

open	closed
[A5]	[]
[D4,B5,C6]	[A5]
[C4,B5,I7]	[D4,A5]
[B5,F6,I7]	[C4,D4,A5]
[C3,E5,F6,I7]	[B5,D4,A5]
[E5,F6,I7]	[C3,B5,D4,A5]
[G3,F6,I7]	[E5,C3,B5,D4,A5]
	[G3,E5,C3,B5,D4,A5]

Complex Search Space and problem solving Approach (in 8-puzzle problem):

Example 1:


Start
g(n) = 0
g(n) = 1
Values of f(n) for each state,
6
4
6
where:
f(n) = g(n) + h(n),
g(n) = actual distance from n to the start state, and
h(n) = number of tiles out of place.
Goal

To summarize:

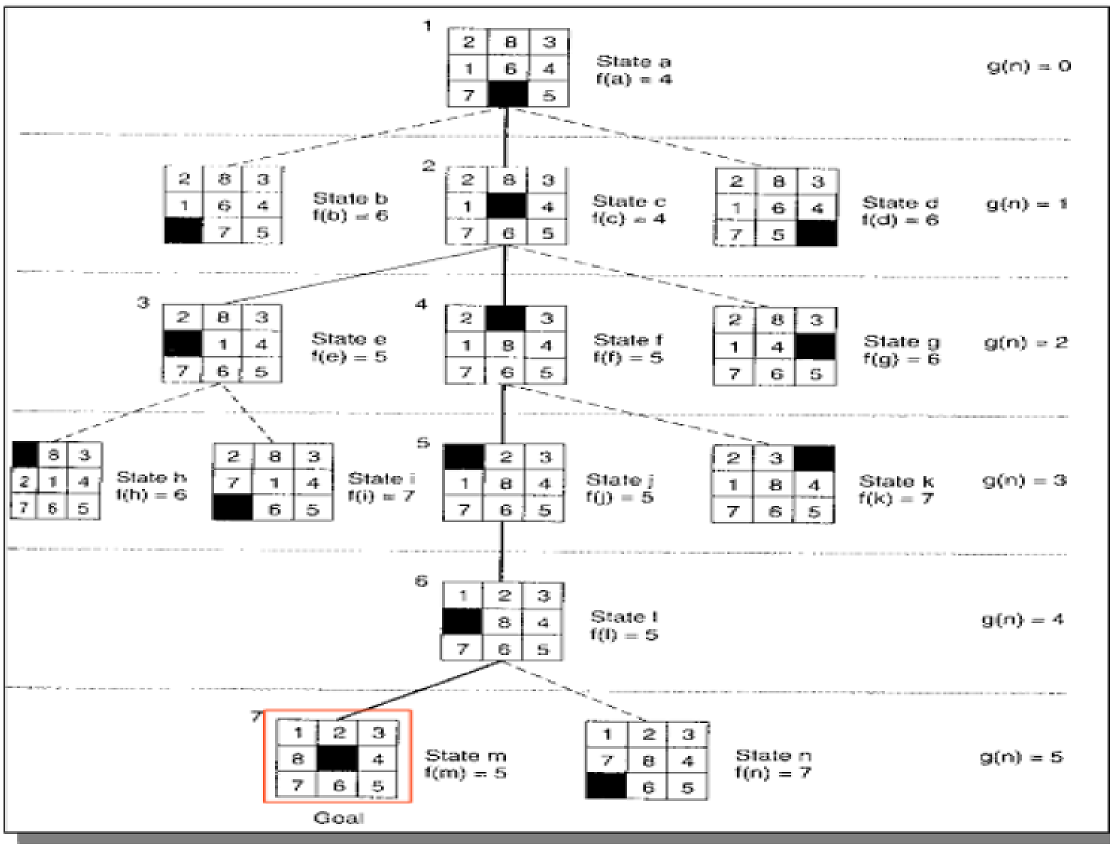
1. Operations on states generate children of the state currently under examination.

2. Each new state is checked to see whether it has occurred before (is on either open or closed), thereby preventing loops.

3. Each state n is given an I value equal to the sum of its depth in the search space g(n) and a heuristic estimate of its distance to a goal h(n). The h value guides search toward heuristically promising states while the **g** value prevents search from persisting indefinitely on a fruitless path.

4. States on open are sorted by their f values. By keeping all states on open until they are examined or a goal is found, the algorithm recovers from dead ends.

5. As an implementation point, the algorithm's efficiency can be improved through maintenance of the open and closed lists, perhaps as heaps or leftist trees.



After implementation of A algorithm, the Open and Closed is shown as follows:

1. Open=[a4], Closed=[]
2. Open=[c4,b6,d6],Closed=[a4]
3. Open=[e5,f5,b6,d6,g6],Closed=[c4,a4]
4. Open=[f5,b6,d6,g6,h6,i7],Closed=[e5, c4,a4]
5. Open=[j5,b6,d6,g6,h6,j7,k7], Closed=[,f5, e5,c4,a4]
6. Open=[l5, b6,d6,g6,h6,j7,k7],Closed=[j5,f5,e5,c4,a4]
7. Open=[m5, b6,d6,g6,h6,j7,k7,n7],Closed=[l5, j5,f5,e5, c4,a4]
8. Success, m=goal

Example2: Consider 8-puzzle problem with start state is shown as follows:

2	8	3
1	6	4
7		5

And the goal state is:

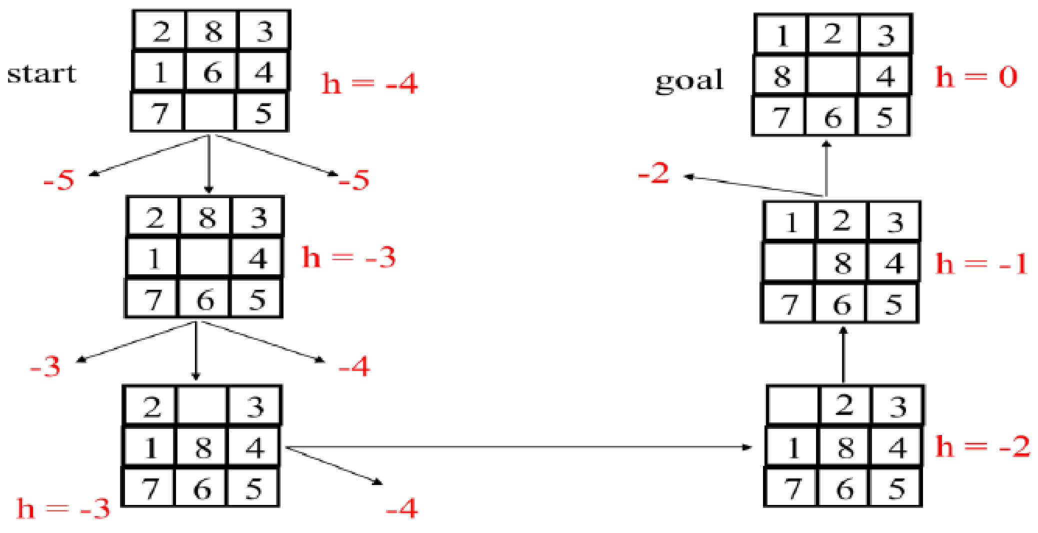
1	2	3
8		4
7	6	5

Assume the heuristic is calculated as following:

h (n) = -(number of tiles out of place)

Draw the path to get the goal using **Hill Climbing** search algorithm?

Answer:



Lecture five

Topics that must be covered in this lecture:

- Methods of Heuristic search (A*- algorithm).
- A* Algorithm Properties

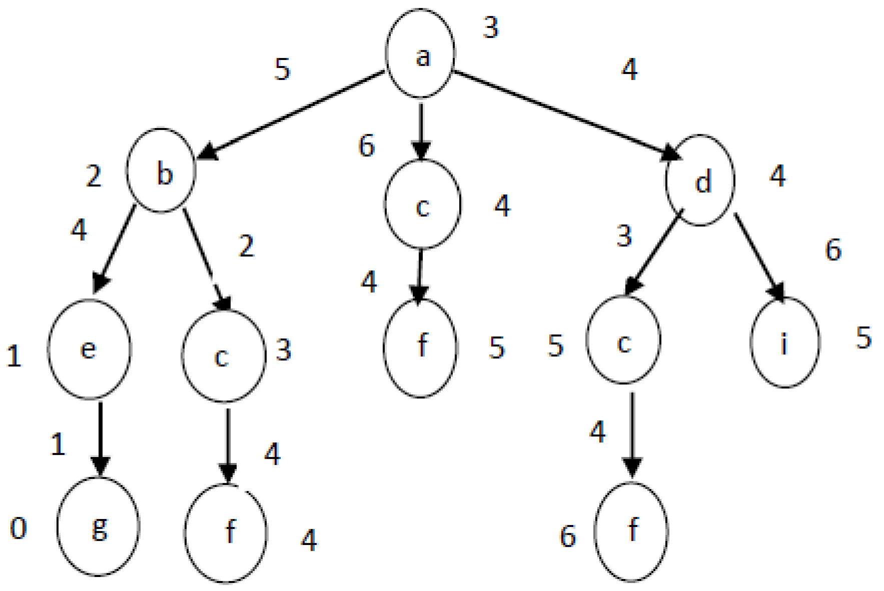
4- A-Star search algorithm

A* algorithm is simply define as a best first search plus specific function. It 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:

$$F^*(n) = g^*(n) + h^*(n).$$

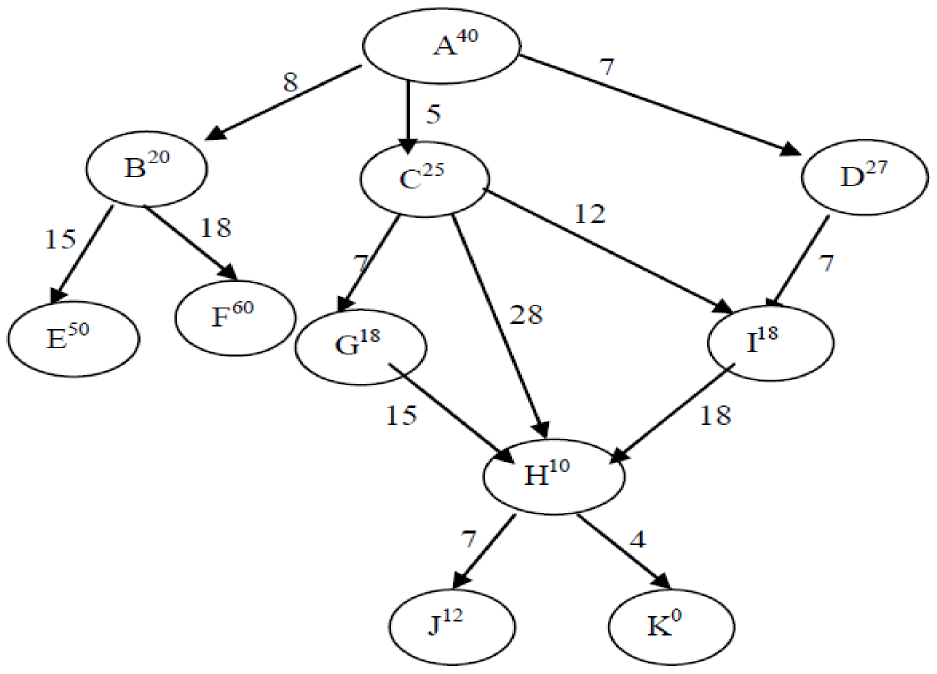
Since g* (n) gives the path cost from the start node to node n, and h* (n) is the estimated cost of the cheapest path from n to the goal, we have f* (n) = estimated cost of the cheapest solution through n .

Example:



open	closed
[A3]	[]
[B7,D8,C10]	[A3]
[D8,E10,C10]	[B7,A3]
[E10,C10,l15]	[D8,B7,A3]
[G10,C10,l15]	[E10,D8,B7,A3]
	[G10,E10,D8,B7,A3]

Example: Use A* algorithm to find the path between A and K for the following search space. Start=[A], goal=K.



open	closed
[A40]	[ ]
[B28,C30,D34]	[A40]
[C30,D34,E73,F86]	[B28, A40]
[G30,D34,I35,H43,E73,F86]	[C30,B28, A40]
[D34,I35,H37,E73,F86]	[G30,C30,B28, A40]
[I32,H37,E73,F86]	[D34,G30,C30,B28, A40]
[H37,E73,F86]	[I32,D34,G30,C30,B28, A40]
[K31,J46]	[H37,I32,D34,G30,C30,B28, A40]

Since K is a goal, stop .

A* Algorithm Properties:-

1) Admissibility

Admissibility means that h(n) is less than or equal to the cost of the minimal path from n to the goal

2) Consistency

Consistency means that the difference between the heuristic of a state and the heuristic of its descendent is less than or equal the cost between them, and the heuristic of the goal equal zero. In other words,

1) $h(ni)-h(nj) \leq cost(ni,nj)$.

2) $h(goal)=0$.

Example: Use A* algorithm to find the path between A and G for the following search space.