- Heuristic search techniques

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. Heuristic search is useful in solving though problems which could not be solved any other way and where solutions take an infinite time or very long time to compute.

Heuristic is the information about the likelihood that a specific node is a better choice to try next rater than another one. Most often heuristic search method are based on maximizing or minimizing some aspect of the problem.

The heuristic function (evaluation function) is a function that evaluates individual problem states and determine how desirable they are. The desirability is usually represented as numbers.

Heuristic search methods

- Generate and test:

This method is basically a depth first search. This is so because complete solution must be created before testing is done. A heuristic is needed so that the search is improved. It can operate by generating solution randomly, but then there is no guarantee that solution will ever be found.

The algorithm for generate and test is as follows:

1. Generate a possible solution which can either be a point in the problem space or a path from the initial state.
2. Test to see if this possible solution is a real solution by comparing the state reached with the set of goal state.
3. if it is a real solution, return. Otherwise repeat from 1.

- Hill climbing search

It is used to help the generator decide which direction to move in the search space. Hill climbing actually a combination of generate and test + direction to move.

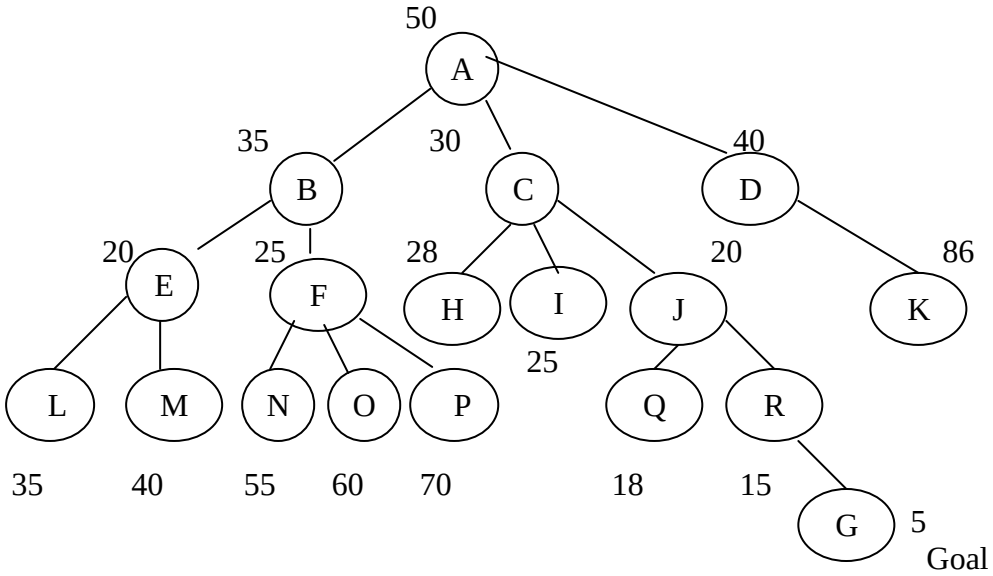
The algorithm for hill climbing is as follows:

1. evaluate the initial state, if it is goal state quit otherwise make current state as initial state.
2. select a new operator that could be applied to this state and generate a new state.
3. evaluate the new state
   if this new state is closer to the goal state, than current state make the new state as the current state
   if it is not better ignore this state and proceed with the current state
   if the current state is goal state or no new operators are available, quit.
   Otherwise repeat from 2.

- Hill climbing search algorithm

```
Begin
  Cs= start, open=[start], stop= false.
   Path=[start]
   While (not stop) do
           Begin
            If(Cs= goal) then return (path)
            Empty open;
            Generate all children of Cs and put them in open;
             If (open[]) 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 is better than X) then X=Y
                       End
                  If (X is better than Cs) then CS=X, add CS to path
                  Else stop
                 End
             Return (fail)
   End
End.
```

Ex: find the goal (G) with a minimum cost by using the Hill climbing search method to the following tree:



#	Cs	Open	Path	
0	A	[A]	[A]	
1	C	[B,C,D]	[A,C]	X=A 50 X=B 35 X=C 30 اقل كلفة
2	J	[H,I,J]	[A,C,J]	X=C X=H 28 X=I 25 X= J 20
3	R	[Q,R]	[A,C,J,R]	X=J X=Q 18 X=R 15
4	G	[G]	[A,C,J,R,G]	X=R X=G is the goal

State space : all node of tree
Search space: A, B, C, D, H, I, J, Q, R, G
Solution path : A ⟶ C ⟶ J ⟶ R ⟶ G

- Problems with hill climbing:

1. Local maxima: is a state that is better than all its neighbors but it's not better than some other states far away.
2. Plateau: is a flat area of the search space, in which all neighboring states have the same value.
3. Ridges: 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

- To solving problems for hill climbing search

These problems could be solved using methods:

1. backtracking to some earlier node and try going in a different direction.

لحل مشكلة ال local maxima يمكن الرجوع الى الوراء أي الى عقدة سابقة ومحاولة الذهاب في اتجاهات مختلفة

2. making big jumps in some direction to try get new section of the search

لتجاوز مشكلة plateau نعمل قفزة (أي نتجاهل بعض الخطوات التي نمر بها) أي عمل قفزة طويلة في بعض الاتجاهات للحصول على مجال جديد في البحث.

3. Apply two or more rules before doing the test.

لحل مشكلة الridge نحاول تطبيق اكثر من قانون قبل اجراء عملية البحث عن الحل

- Best-first search method:

By adding backtracking to hill climbing the result that method called Best first search. Given a path in a rooted graph a node is said to be an ancestor(predecessor) of all nodes positioned after it is the sequence and descendent(successors) of all nodes positioned before it is sequence. An immediate ancestor to be a node is a parent. An immediate successors of a node are referred to as children, sibling or offspring.

Best first search is a way of combining the advantages of both depth first and breadth first search into a single method. At each step of the best first search process we select the most promising of the nodes we have generated so for. This is done by applying an appropriate heuristic function to each of them.

- Best-first search method:

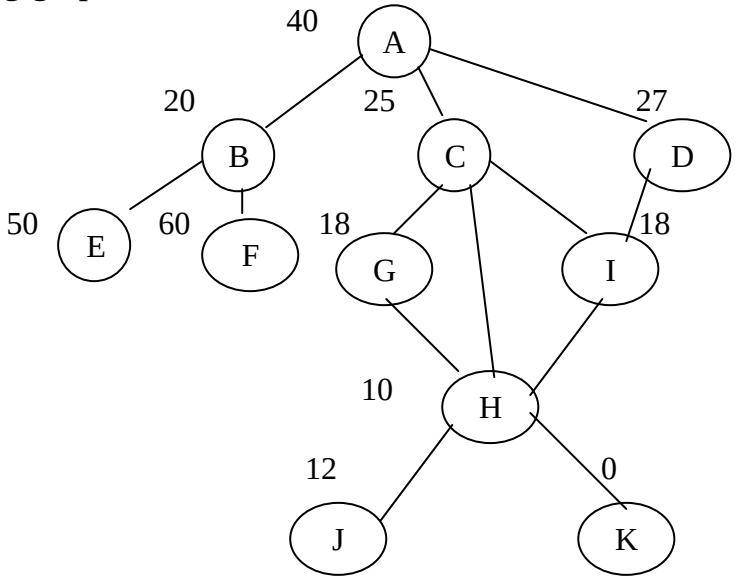
1. Use two ordered lists open and close
2. start with the initial node n0 and put it on the ordered list open
3. create a list close. This is initially an empty list.
4. if open is empty exit with failure.
5. select first node on open. Remove it from open and put it on close. Call this node n.
6. if n is the goal node exit. The solution is obtained by tracing a path backward a long the arcs in the tree from n0 to n.
7. expand node n. this will generate successors. Let the set of successors generated be m. create arcs from n to each member of m.
8. reorder the list open according to the heuristic and go back to step 4.

- Best-first search algorithm

```
Begin
   Input the start node S and the set G of goal node
   Open[s], closed=[];
```

```
    Pred[s]=0; found=false;
    Cs= S;
While (open<>[]) & (found = false) do
 Begin
      L= the set of nodes on open for which F has the best value;
      If (L is singleton) then X= that element
      Else
       If there are any goal nodes in L
       Then X= one of them
        Else
        X= the first element of L
        Remove X from open and put it on closed
         If (X is a goal node) then found=true
         Else  Begin
               Generate the set successors of children of X for each child
               Y do
               If(Y is not already in open or closed) then
                Begin
                    Compute h[Y], pre[Y]=X
                     Insert Y in open;
                 End
             End
         If found= false then output "frailer"
         Else trace the path using pred pointers
    End
End
```

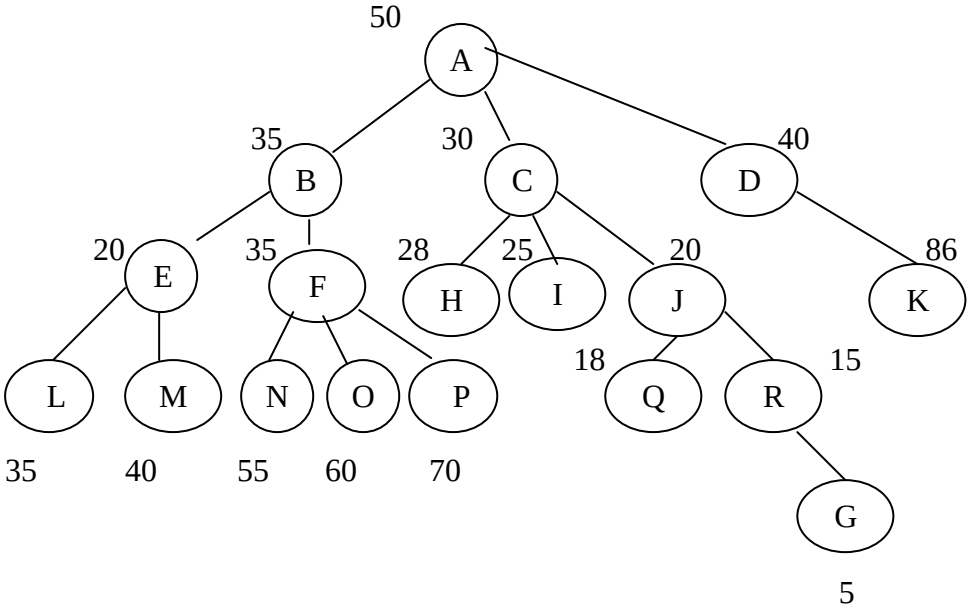
Ex: find the goal (K) by using the Best-first search method to the following graph



#	Cs	Open	Close
0	A	[A]	[ ]
1	B	[B,C,D]	[A]
2	C	[E,F,C,D]	[B,A]
3	H	[G,H,I,E,F,D]	[C,B,A]
4	K	[J,K,G,I,E,F,D]	[H,C,B,A]
5	K	The goal	stop

Solution path: A → C → H → K

Ex: find the goal (G) by using the Best first search method to the following tree:



#	Cs	Open	close	Path
0	A	[A]	[]	A
1	C	[B,C,D]	[A]	A- C
2	J	[B,D,H,I,J]	[C,A]	A- C – J
3	R	[B,D,H,I,Q,R]	[J,C,A]	A – C – J – R
4	G	[B,D,H,I,Q,G]	[R,J,C,A]	A – C – J – R - G

Solution path: A → C → J → R → G

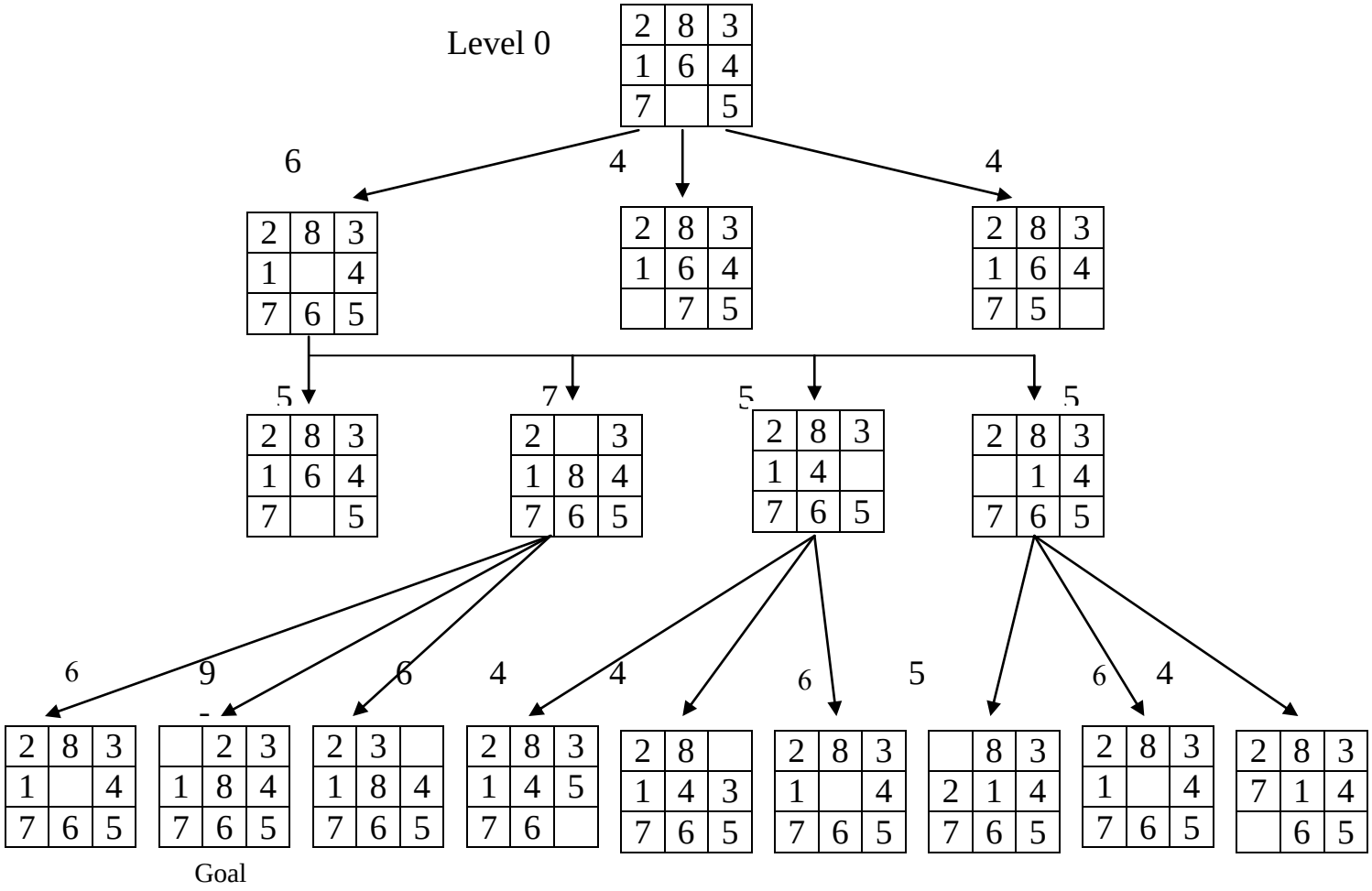
Ex: find the goal (Z) by using the best first search method, where (A) the initial state.

A=

2	8	3
1	6	4
7		5

Z=

	2	3
1	8	4
7	6	5



- A star search (A* search)

Minimizing the total estimated solution cost. (A star search)

تقليل كلفة الحل التخمينية الكلية باستخدام طريقة (A star search) ترتكز هذه الخوارزمية في البحث على افضل الحلول واقلها كلفة.

- by this method the informed(heuristic) search strategy which uses problem- specific knowledge can find solutions more efficiently.
- 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.

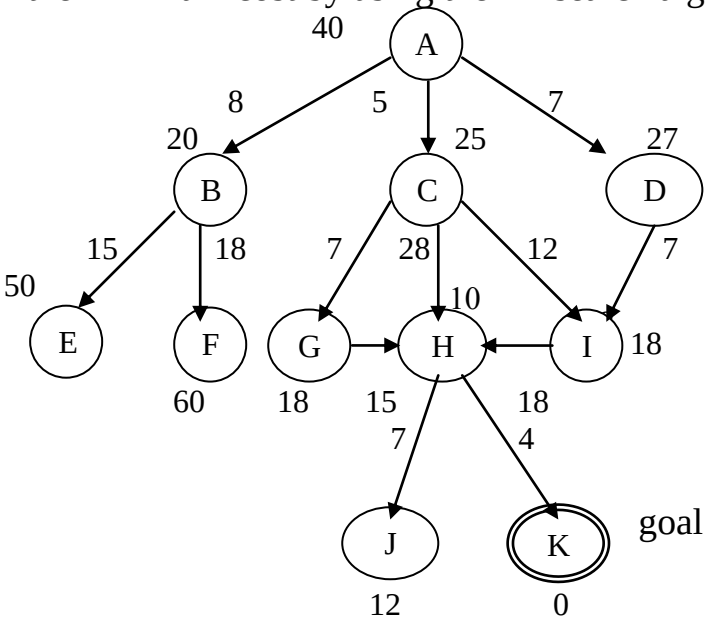
f(n) : estimated cost of the cheapest solution through n.

- A star search (A* search) algorithm:

```
Begin
   Input the start node S and the set G of goal nodes
   Open[s]; closed = [];
    G[s]=0; pred[s]=NULL; found = false;
     F[s]= h(s);
    While (open<> []) & (found = false) do
     {
       L= the set of nodes on open for which F has the best value;
       If (L is singleton) then X= that element
       Else
        If there are any goal nodes in L
        Then X= one of them
         Else
         X= the first element of L
         Remove X from open and put it on closed
          If (X is a goal node) then found=true
          Else  Begin
           Generate the set successors of  X for each Y in successors do
            If(Y is not already on open or closed) then
            Begin
            G[Y]=G[X]+cost(X,Y)
              F[Y]= G[Y]+h(Y)
              Pred[Y]=X
              Insert Y I open;
              End
           Else
              Begin      Z= pred[Y]
                         Temp= F[Y]-G[Z]-cost(Z,Y)+G[X]+cost(X,Y)
                         If(temp < F[Y]) then
                    Begin
                       G[Y]= G[X]+cost(X<Y)
                        F[Y}= temp;
                        Pred[Y]= X ;
                        If(Y in closed) then insert Y on open and remove
                        Y from closed
                        End
                       End
                   End
                 End;
```

}
If found= false then output "goal not found"
Else trace the pointers in pred fields from X
Back to S to get the path from S to the Goal state
End.

Ex: consider the graph bellow, starting with the node (A) to find the goal node (K) with the minimum cost by using the A* search algorithm.



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

State space : all the nodes of the graph
Search space: A, B, C, G, D, I, H, K
Solution path: A → C → G → H → K

ملاحظة:

- في الخطوة (4) تم حذف العقدة (H43) القديمة لان القيمة الجديدة لها هي (H37) وهي اصغر من القديمة لذلك يتم اخذها
- في الخطوة (5) تم حذف العقدة (I35) القديمة لان القيمة الجديدة لها هي (I32) وهي اصغر من القديمة
- في الخطوة (6) لم يتم حذف العقدة (H37) القديمة وذلك لان القيمة الجديدة H42

- Properties of heuristic function

1. Admissibility: A heuristic function is admissible if it finds the shortest path to a goal state.

Note: all A* algorithms are admissible.

2. monotoncity : A heuristic function is monotonic if the first visit to any intermediate node gives the shortest path to that node.

Note: each monotoncity function is admissible but not the complement because it choose the shortest path.

3. informedness : A heuristic function is h1 is said to be more informed than a heuristic function h2 if h1(n)>= h2(n) for all nodes n. if h1 is more informed than h2 then the subsequence searched by h1 will be less than that searched by h2.

1- الدالة التي تجد اقصر طريق للهدف تعتبر admissible
2- الدالة التخمينية التي تجد اقصر طريق للهدف باول زيارة لل node أي تحصل على اقصر طريق بزيارة واحدة لل node .
3- h1 تكون اكثر حكمة من h2 اذا كانت h1 تزور اقل node من h2 لذلك هي اكثر حكمة.