Informed Search

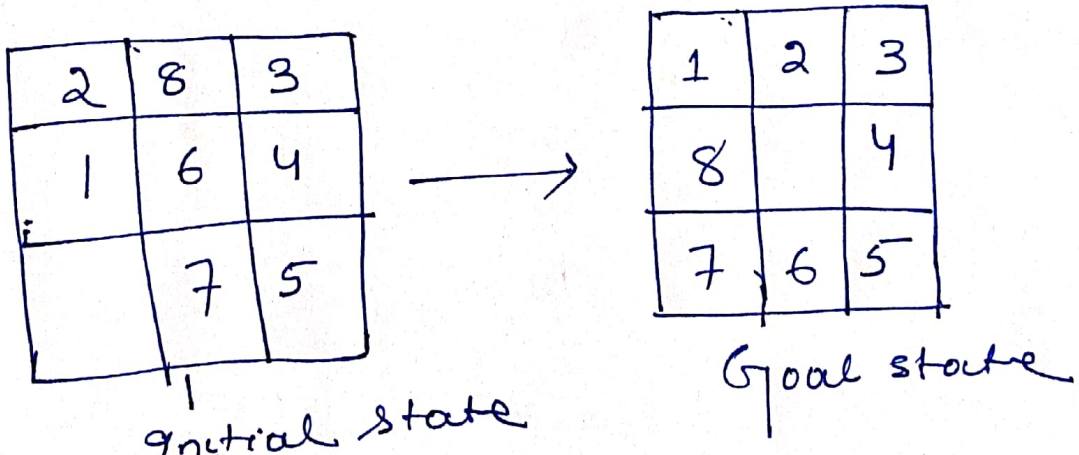
Informed search is also known as heuristic or intelligent search, uses information about the problem to guide the search. usually guesses the distance to a goal state.

Therefore if it is efficient, but the search may not be always possible.

Uninformed Search

Also called blind or exhaustive or brute-force search, Use no information about the problem to guide the search and therefore may not be very efficient.

Ex 8-puzzle problem



Here heuristic function : No. of tiles misplaced

& the heuristic value

$$h(n) = 5$$

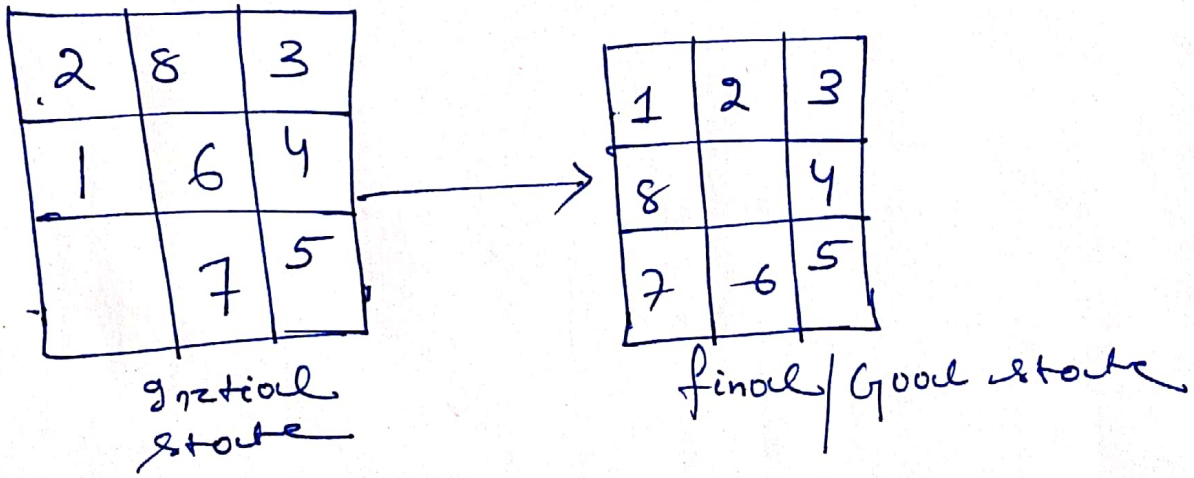
So, More than 5-steps are required to arrange the puzzle.

Note Heuristic function/search may yield an optimal solution or may not. But maximum time it yields optimal solution.

⇒ The principle of heuristic is to decide which among several alternatives are the best or to be most effective to reach the goal state.

→ A heuristic function at a node 'n' is used to estimate the cost from current node to the goal node. It is represented as 'h(n)'

Ex Define the heuristic function for 8-puzzle tiles.



Here $h(n) = 5$ as the tile no. 2, 8, 1, 6 and 7 are misplaced.

Manhattan Distance:

Another heuristic function of the 8-puzzle tile is manhattan distance. i.e the sum of the distance of all the tiles that are out of place

$h(n) = 1+1+0+0+0+1+1+2 = 6$

The no of tiles displaced to get the soln is the heuristic value.

AI depends on heuristic function

A* Algorithm

→ A* is a search algorithm that finds the shortest path between some nodes S and T in a graph

⇒ Suppose we want to get to the node T, and we are currently at node V. informally a heuristic function $h(V)$ is the function that 'estimate' how V is away from T.

Ex Suppose I am driving from Srinagar to Jammu. A heuristic function would tell me approximately how much longer I have to drive.

→ A heuristic function is admissible if it never overestimate the distance to the goal.

→ Example: $h(V) = 0$ is an admissible heustic

⇒ If nodes are points on the plane then the straight line distance $h(V) = \sqrt{(V_x - T_x)^2 + (V_y - T_y)^2}$ is an admissible heuristic.

→ Suppose two nodes u and v are connected by an edge. A heuristic function h is consistent or monotone. If it satisfies the following:

$$h(u) \leq e(u,v) + h(v)$$

where $e(u,v)$ is the edge distance from u to v

A* Algorithm

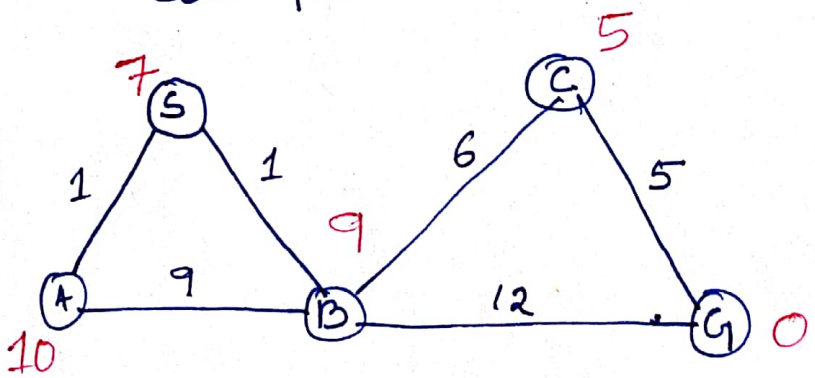
Input: Priority Queue

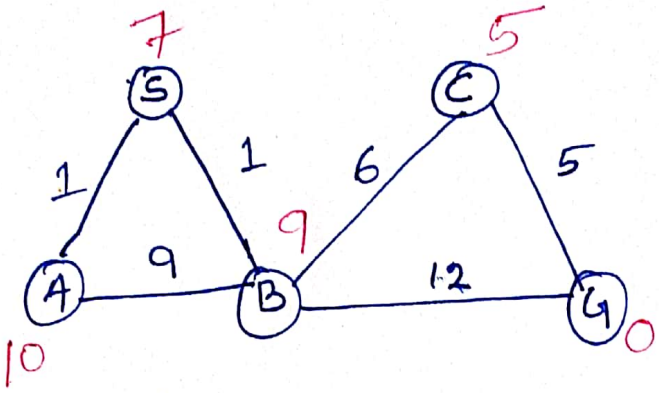
Queue: Path only containg root.

Algo

- while (Q is not empty and first path not reach goal) do
  - Remove first path from Q.
  - Creat paths to all children
  - Reject paths with loops
  - Add paths and Q (by $f(n) = c(n) + h(n)$)
  - If Q contains path: P, R and P ends in node N_j and R contains node N_j and cost_P ≥ cost_R

    Then Remove P
  - If goal reached then success else failure.



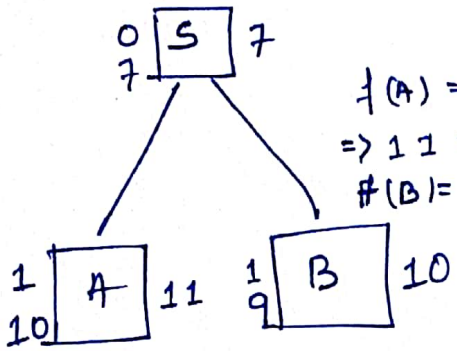


for (G)

(1) 0 S 7 $f(S) = g(S) + h(S)$
7 $7 = 0 + 7$

Q : <S>

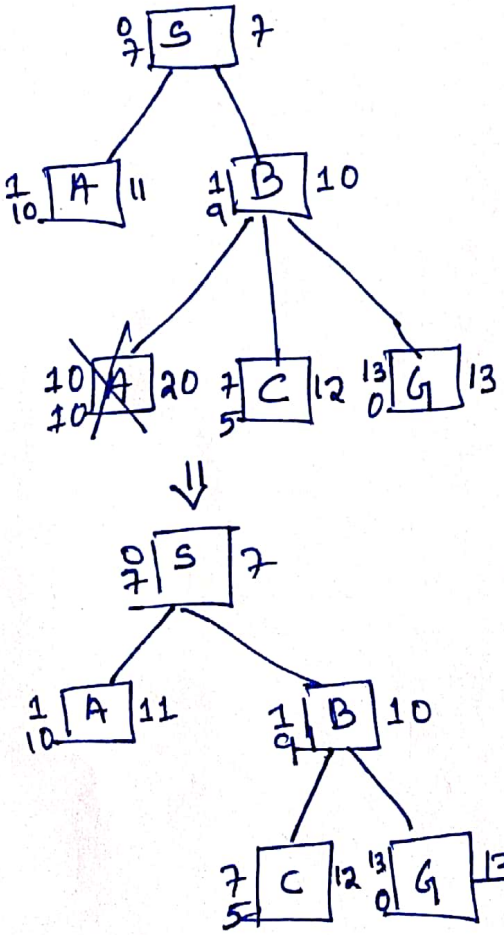
(2)



$f(A) = g(A) + h(A)$
$\Rightarrow 11 = 1 + 10$
$f(B) = 10 = 1 + 9$

Q : <SB, SA>
10 11

(3)



Remove pop SB from Q and insert SBA, SBC and SBG

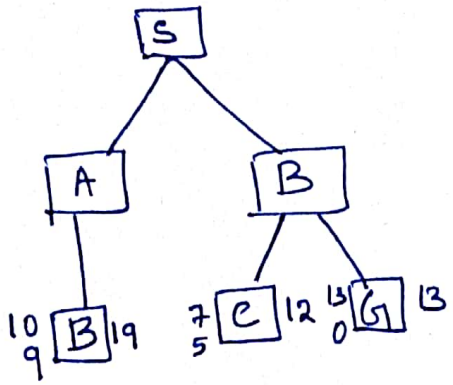
Q : <SA, SBC, SBG, SBA>
11 12 13 20

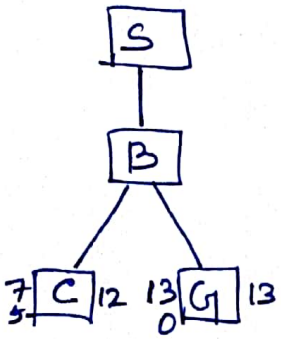
Here SA and SBA
Both path ends with vertex A
so remove the higher cost path
i.e SBA

Now

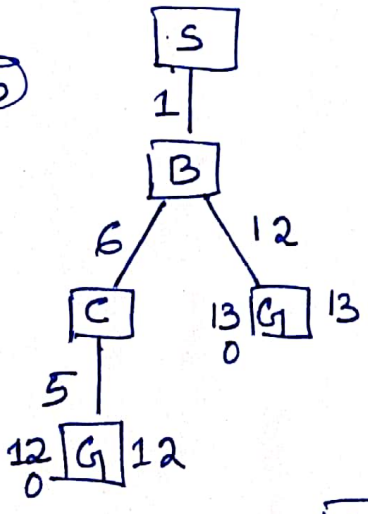
Q : <SA, SBC, SBG>
11 12 13

(4)

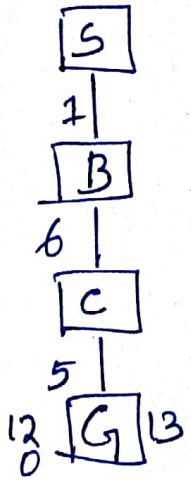




(5)



(6)



(5)

Now Remove/delete SA from Q & insert the respective adj edges

Q: <SAB, SBC, SBG>
19 12 13

After sorting

Q: <SBC SBG SAB>
12 13 19

Now SAB end with B and SBC and SBG containing B with less cost, so delete SAB from Q.

Q: <SBC, SBG>
12 13

So remove SBC and add adj of SBC i.e SBCG

Q: <SBCG, SBG>
12 13

Now Both the paths in Q ~~containing~~ end with vertex G so remove the higher cost path i.e SBG