1. **Suppose you decide not to keep the explored set (i.e., the place where the explored nodes are saved) in the A* algorithm.**

   - **If the heuristic used is admissible, will the new algorithm still be guaranteed to find the optimal solution? Explain why (or why not). [2 marks]**

A heuristic is said to be admissible if it doesn't overestimate the cost to the goal. If we decide decide not to keep a record of the explored set in A* algorithm, then the new algorithm will not find the optimal solution. Incase we don't have the knowledge about the already visited nodes, we may visit the same nodes multiple times and it could even cause an infinite loop to occur, which leads to an ineffective and incomplete algorithm.

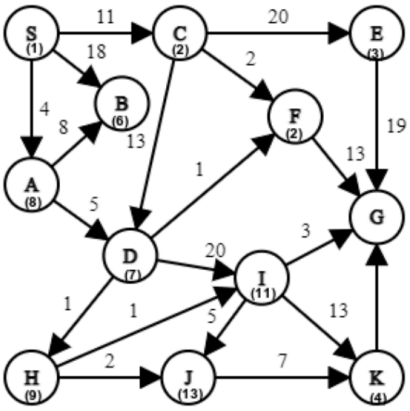
   - **Will the new algorithm be complete? Explain. [1 marks]**

No, the algorithm will be incomplete because the possibility of an infinite loop looms around. In case of an infinite loop occurring, even if there is an optimal solution, our modified algorithm would not be able to get to the optimal solution and that's why the algorithm would be incomplete.

   - **Will the new algorithm be faster or not? Explain. [1 marks]**

There might be some memory improvements, because we no longer need to write the visited nodes onto the memory and overheads are reduced. So it will be a little more faster. But, because of that , it leads to a non optimal solution and may be caught in an infinite loop. It greatly affects the overall efficiency of the algorithm, but yes, algorithm will be bit more faster.

2. **Given the graph, find the cost-effective path from S to G. S is the source node, and G is the goal node using A* [2 marks], BFS [2 marks], and Dijkstra algorithm. [2 marks]Please note: cost written on edges represents the distance between nodes. The cost written on nodes represents the heuristic value. Heuristic for Goal Node G is 0. Distance from K to G is 6**



Initially , we have starting vertex as S. From S to A(4),B(18),C(11)

A* Search

Using A* search, we first consider the Source S. Then compute the heuristic f(n)=g(n)+h(n) for each of the nodes which may be reached from S. We have
A(4+8=12), B(6+18=24), C(11+2=13).
Now out of these three nodes in the frontier, we will go with A(12) as it has the lowest cost. Now, we expand the node A. We can go to the node D(4+5+7=16) and node B(8+6=14). Now in the frontier we have D(16),B(24), B(14) via A and C(13). We take C(13) as it has the lowest cost now. We expand C -
D(11+13+7=31), F(11+2+2=15) and E(11+20+3=34).
The B(14) path could have taken but it is a dead end

F(15) has the lowest cost path so we take that.

On expanding F, we see we can reach Goal G with a cost of 11+2+13=26.
Optimal path : S->C->F->G

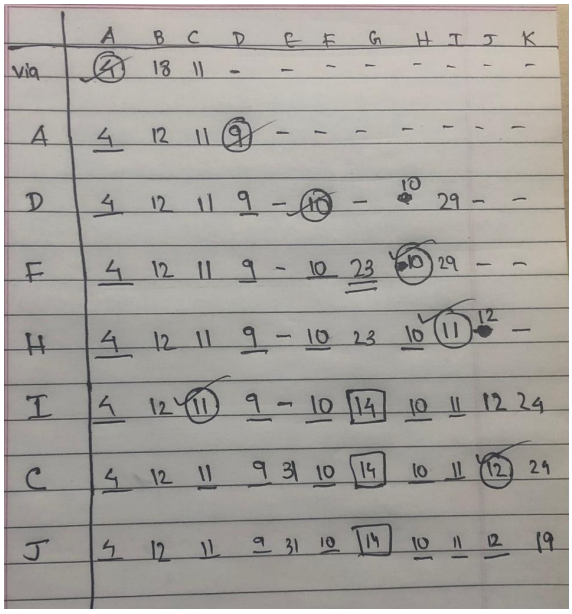
BFS

From S, We have
A(4+8=12), B(6+18=24), C(11+2=13).
Now out of these three nodes in the frontier, we will go with A(12) as it has the lowest cost. Now, we expand the node A. We can go to the node D(4+5+7=16) and node B(8+6=14). Now in the frontier we have D(16),B(24), B(14) via A and C(13). We take C(13) as it has the lowest cost now. We expand C -
D(11+13+7=31), F(11+2+2=15) and E(11+20+3=34).
The B(14) path could have taken but it is a dead end

F(15) has the lowest cost path so we take that.

On expanding F, we see we can reach Goal G with a cost of 11+2+13=26.
Optimal path : S->C->F->G

Dijkstra's algorithm



Dijkstra's algorithm has found the path **S-> A -> D -> H -> I -> G** with a cost of 14. I did not expand the remaining two nodes because they wouldn't be giving any better routes anyway.

3.

(c) Show Uniform Cost Search and A* search on this data.

In uniform cost search we would be finding the cost to reach each of the neighbours and select the path to the neighbour in which we can reach the distance in the smallest cost possible. For example , from S, if we have cost from S to A as 5 and from S to B as 10, we would choose the path S to A. We make use of a queue to store the paths and the corresponding cost and we sort the queue based on the total path cost in ascending and pop out the queue every time we want to select the path. It is an efficient algorithm that is uninformed or it just doesn't have a heuristic value.

In A* search, we would be using a heuristic function to determine which path to choose. We use a function f(n)=g(n)+h(n) where h(n) is the heuristic function and g(n) is the cost from one node to the next neighbour node. It is computationally more expensive but it is more efficient.

(f) Design two different, non-trivial (e.g., a constant h(n) = c is a trivial heuristic) heuristics for A*. (1) The first one should be admissible. Empirically verify that the admissible heuristic yields the optimal solution. (2) Come up with an inadmissible heuristic function that will expand fewer nodes than your admissible heuristic in (1). Show this property for at least two origin-destination pairs in your code.

The two non trivial heuristics I have taken are :

1. f(n)=g(n)+h(n), where g(n) is the distance between once node to its neighbouring node and h(n) is the straight line distance from one node to the goal node. I have used the opencagegeocode api to get those data regarding latitude and longitude values. From that we could calculate the straight line distance from one location to the goal location. This heuristic is admissible because it will never overestimate the cost to goal. On observation itself we can see that the straight line distance from one point to the other point is the minimum possible distance between any two points. Hence , our proposed heuristic is indeed admissible.
2. The second heuristic is a weighted heuristic , where we will be multiplying the heuristic function h(n) by a weight value. So the total formula would look like :
   f(n)=g(n)+weight*h(n)

This is inadmissible because multiplying the straight line distance value (h(n)) by a weight would indeed be overestimating the cost to reach the goal and hence it is an inadmissible heuristic.

For example:

- **Agartala -> Hubli**

A* Admissible heuristic :
['Agartala', 'Calcutta', 'Panjim', 'Hubli']

A* Inadmissible heuristic:
['Agartala', 'Panjim', 'Hubli']

Cost for A* admissible is : 3027.0
Cost for A* Inadmissible is : 3697.0

- **Cochin -> Agartala**

A* Admissible:

['Cochin', 'Pondicherry', 'Allahabad', 'Patna', 'Agartala']

A* Inadmissible :
['Cochin', 'Agartala']

Cost for A* admissible is : 3690.0
Cost for A* Inadmissible is : 4304.0