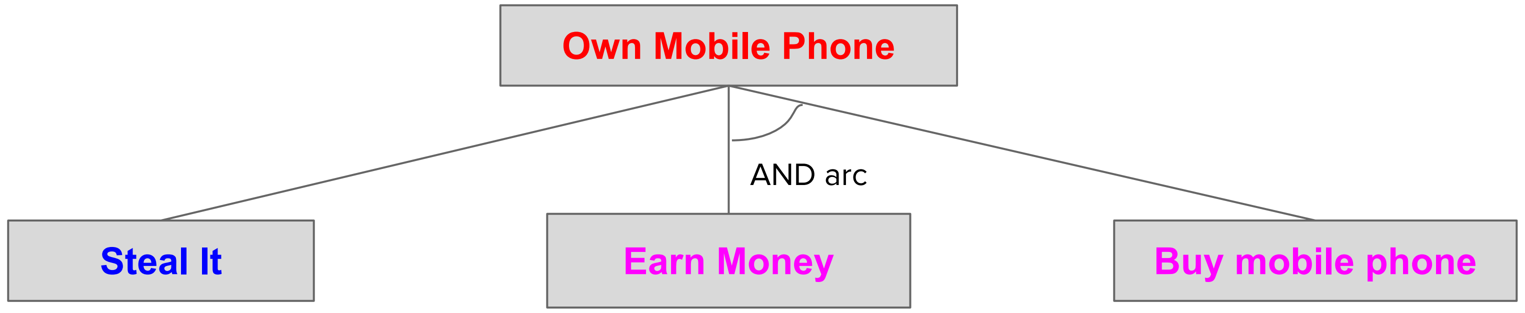
AO* Algorithm

Introduction

- Its an informed search and works as **best first search**.
- AO* Algorithm is based on problem decomposition (Breakdown problem into small pieces).
- Its an efficient method to explore a solution path.
- AO* is often used for the common pathfinding problem in applications such as video games, but was originally designed as a general graph traversal algorithm.
- It finds applications in diverse problems, including the problem of parsing using stochastic grammars in NLP.
- Other cases include an Informational search with online learning.
- It is useful for searching game trees, problem solving etc.

AND-OR Graph

- AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems, all of which must then be solved.



AND-OR Graph

- Node in the graph will point both down to its successors and up to its parent nodes.
- Each Node in the graph will also have a heuristic value associated with it.

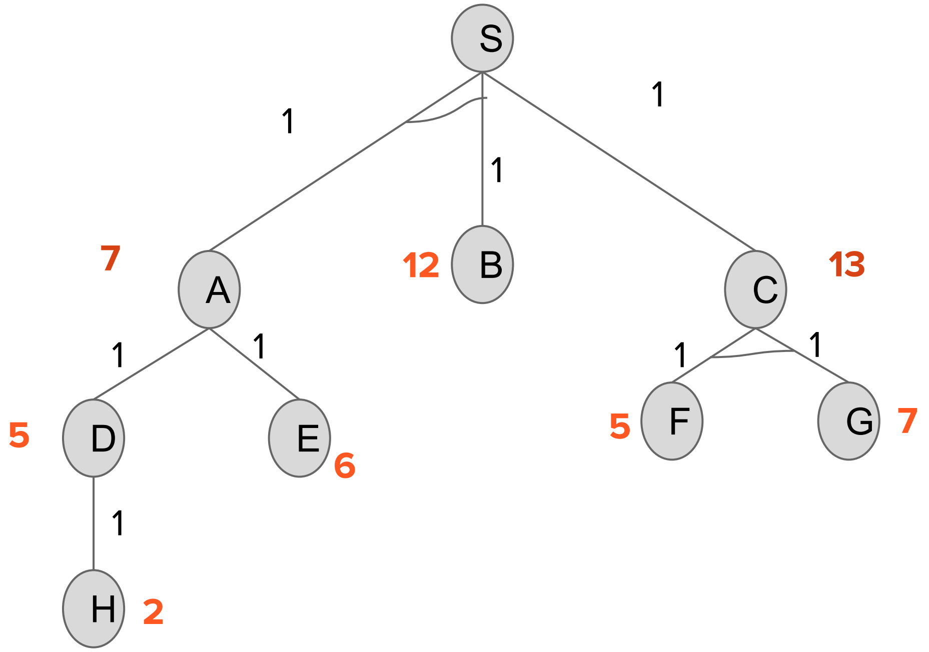
$$f(n)=g(n)+h(n)$$

f(n): Cost function.

g(n): Actual cost or Edge value

h(n): Heuristic/ Estimated value of the nodes

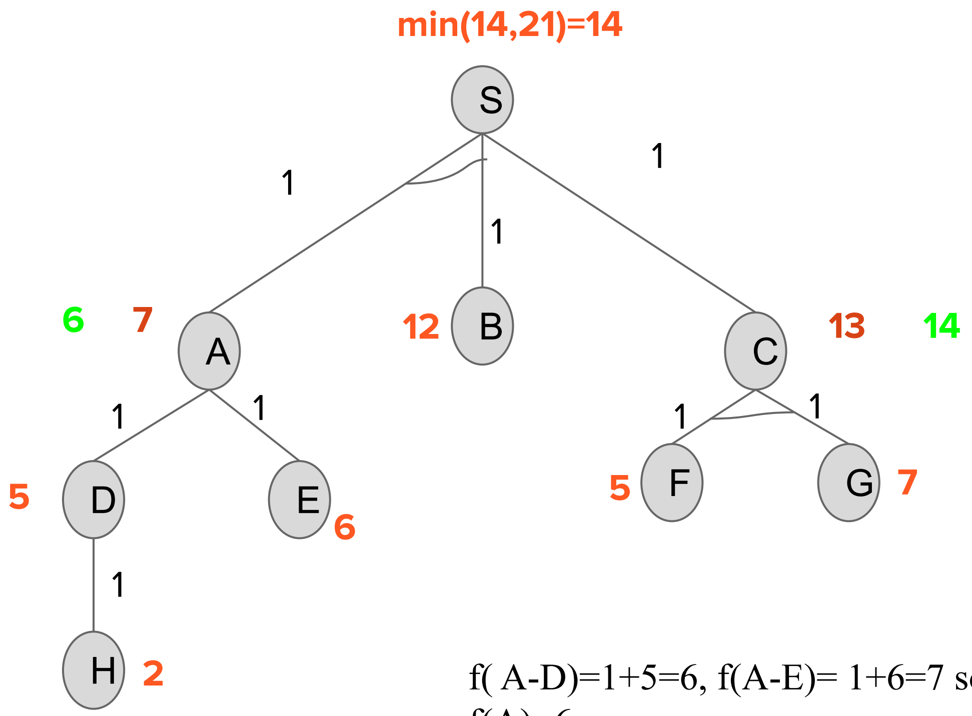
AO* Example 1



AO* Example 1

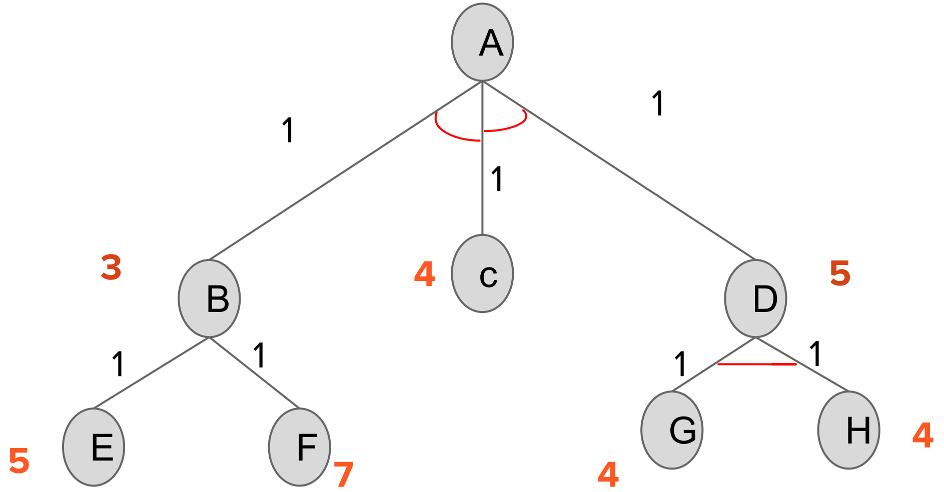
f(n)=g(n)+h(n)
Path-1: f(S-C)= 1+13=14
Path-2: f(S-A-B)=1+1+7+12=21
f(C-F-G)=1+1+5+7=14
f(S-C)=1+14=15 **(revised)**

Revised cost: 15

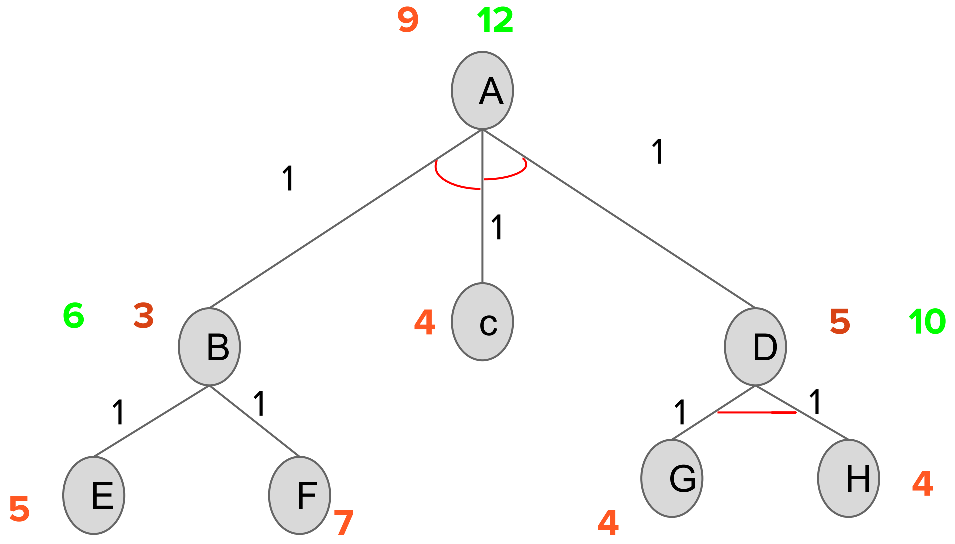


f(A-D)=1+5=6, f(A-E)= 1+6=7 so revised
f(A)=6
Revised (S-A-B)=1+6+1+12=20 X

AO* Example 2



AO* Example 2



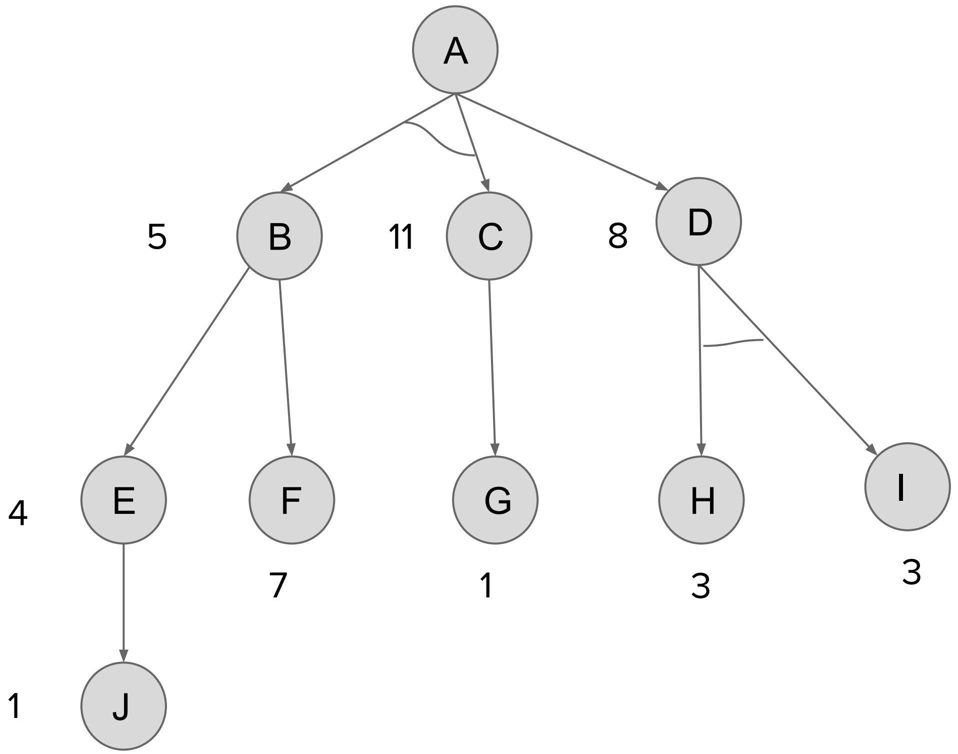
Path -1: f(A-B-C)= 1+1+3+4= 9
f(B-E)=1+5=6
f(B-F)=1+7=8
f(A-B-C)= 1+1+6+4= 12

Path-2: f(A-C-D)=1+1+4+5=11
f(D-G-H)= 1+1+4+4 =10
f(A-C-D)= 1+1+4+10= 16

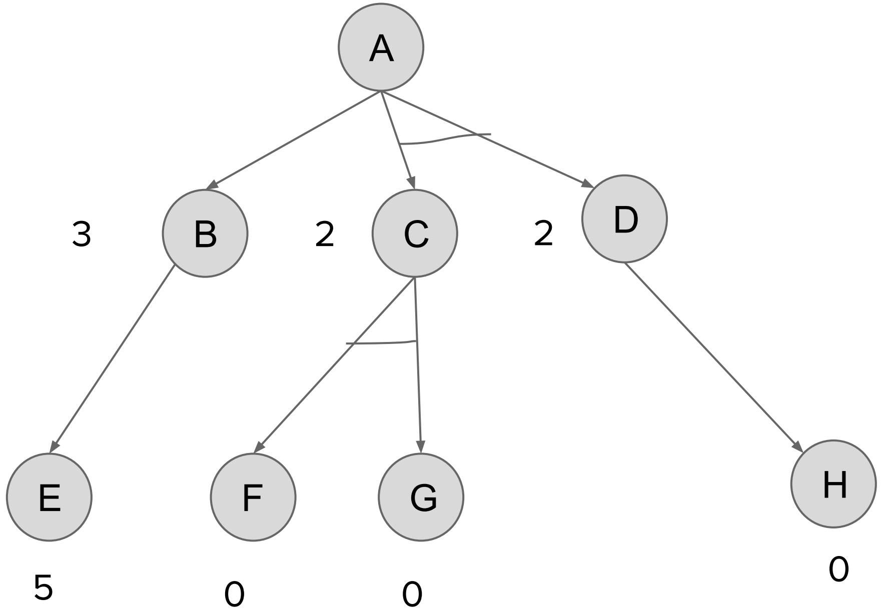
AO* Algorithm:

The algorithm does not explore all the solution path once it find a solution

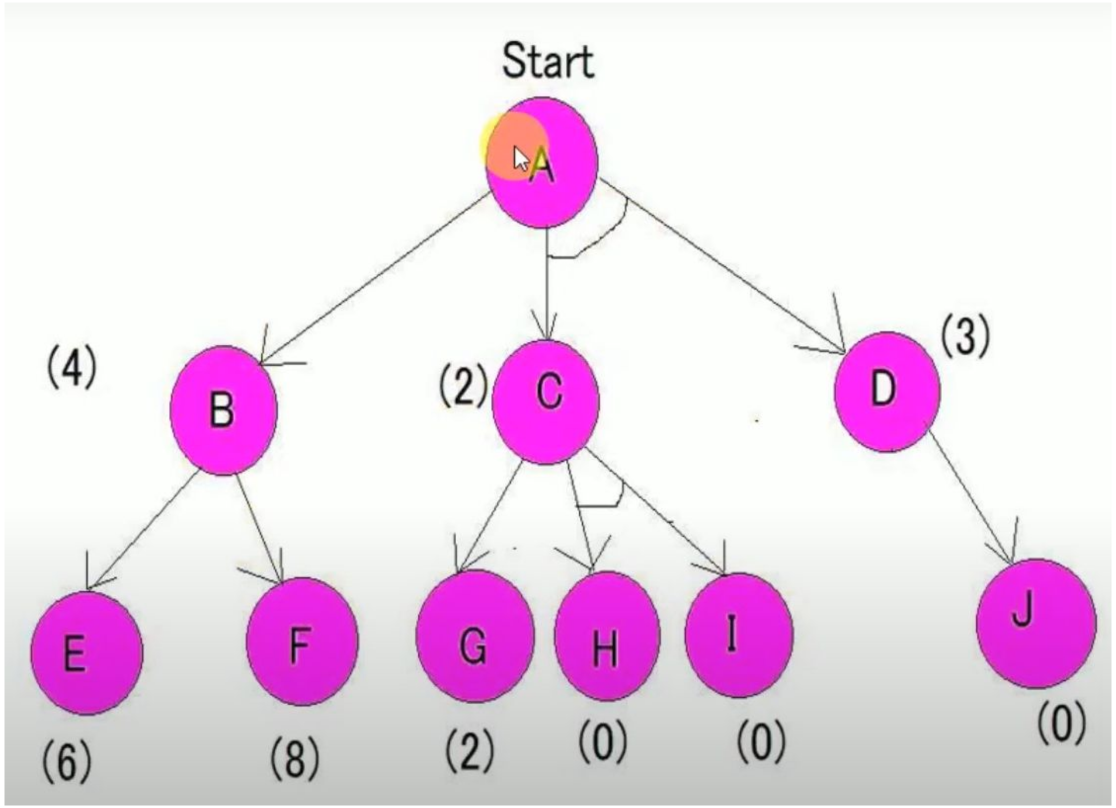
Example 3



Example 4



Example 5:



AO* Algorithm

1. Initialise the graph to start node
2. Traverse the graph following the current path accumulating nodes that have not yet been expanded or solved
3. Pick any of these nodes and expand it and if it has no successors call this value FUTILITY otherwise calculate only f` for each of the successors.
4. If f` is 0 then mark the node as SOLVED
5. Change the value of f` for the newly created node to reflect its successors by back propagation.
6. Wherever possible use the most promising routes and if a node is marked as SOLVED then mark the parent node as SOLVED.
7. If starting node is SOLVED or value greater than FUTILITY, stop, else repeat from 2.

A*	AO*
It represents an OR graph algorithm that is used to find a single solution (either this or that). It is a computer algorithm which is used in path-finding and graph traversal. It is used in the process of plotting an efficiently directed path between a number of points called nodes. In this algorithm you traverse the tree in depth and keep moving and adding up the total cost of reaching the cost from the current state to the goal state and add it to the cost of reaching the current state.	It represents an AND-OR graph algorithm that is used to find more than one solution by ANDing more than one branch. In this algorithm you follow a similar procedure but there are constraints traversing specific paths. When you traverse those paths, cost of all the paths which originate from the preceding node are added till that level, where you find the goal state regardless of the fact whether they take you to the goal state or not.

Thank You