BRANCH & BOUND

Terminology

Live node is a node that has been generated but whose children have not yet been generated.

E -node (Expanded Node) is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.

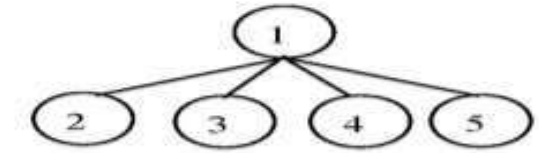
Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.

Branch-and-bound refers to all state space search methods in which all children of an E-node are generated before any other live node can become the E-node.

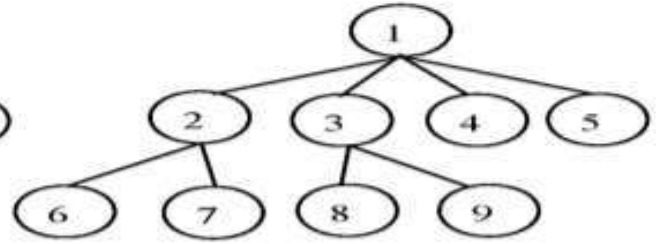
Branch and Bound

Definitions:

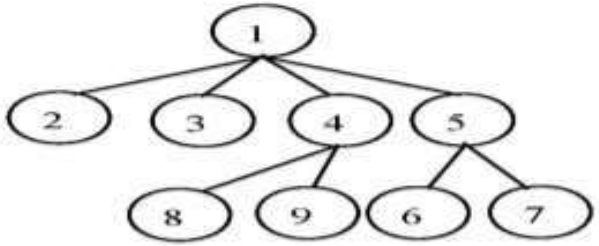
- Branch and Bound is a state space search method in which all the children of a node are generated before expanding any of its children.
- **Live-node**: A node that has not been expanded.
- It is similar to backtracking technique but uses BFS-like search.



Live Node: 2, 3, 4, and 5



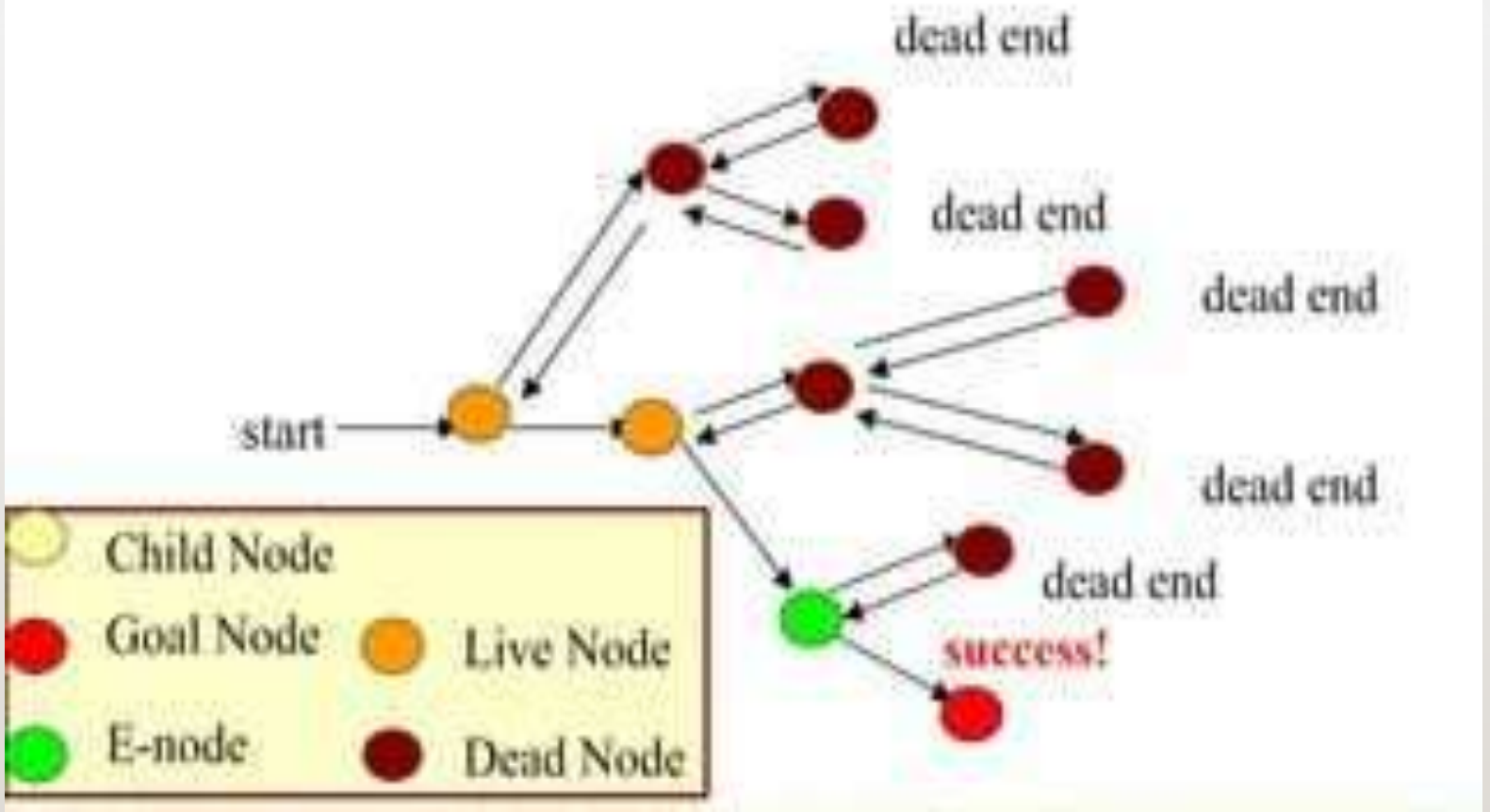
FIFO Branch & Bound (BFS)
Children of E-node are inserted in a queue.



LIFO Branch & Bound (D-Search)
Children of E-node are inserted in a stack.

- Dead-node: A node that has been expanded
- Solution-node

Types Of Nodes - Demonstration



BRANCH AND BOUND -- THE METHOD

The term branch-and-bound refers to all state space search methods in which all children of the E-node are generated before any other live node can become the E-node.

In branch-and-bound terminology breadth first search(***BFS***)- like state space search will be called FIFO (First In First Output) search as the list of live nodes is a first-in-first-out list (or queue).

A ***D-search*** (depth search) state space search will be called LIFO (Last In First Out) search, as the list of live nodes is a last-in-first-out list (or stack).
Bounding functions are used to help avoid the generation of sub trees that do not contain an answer node.

•A search strategy that uses a cost function c(x)= f(h(x))+ g(x) to select the next E-node, would always choose for its next E-node a live node with least cost. Hence, such a search strategy is called an LC-search (Least Cost search).

• BFS and D-search are the special cases of LC-search.

• If g(x)=0 and f(h(x)) = level of node x, then a LC-search becomes BFS.

•If f (h(x))=0 and g(x) > g(y) whenever y is a child of x, then LC search is essentially a D-search.

•An LC-search coupled with bounding functions is called an LC branch-and-bound search.

•Generating state space tree using FIFO search with bounding functions is called FIFOBB

•Generating state space tree using LIFO search with bounding functions is called LIFOBB

One of the problem-solving approaches is the branch and bound. It is similar to backtracking in that it also employs the state space tree. It is employed in the solution of optimization and minimization problems. If we are given a maximization problem, we can use the Branch and bound technique to transform it by simply changing the problem to a minimization problem.

Example: Jobs = {j1, j2, j3, j4}

P = {10, 5, 8, 3}

d = {1, 2, 1, 2}

The jobs, problems, and problems given above are listed above. The solutions can be written in two ways, as shown below:

If we want to do jobs j1 and j2, the solution can be represented in two ways:

The subsets of jobs are the first way to represent the solutions. S1 = {j1, j4}

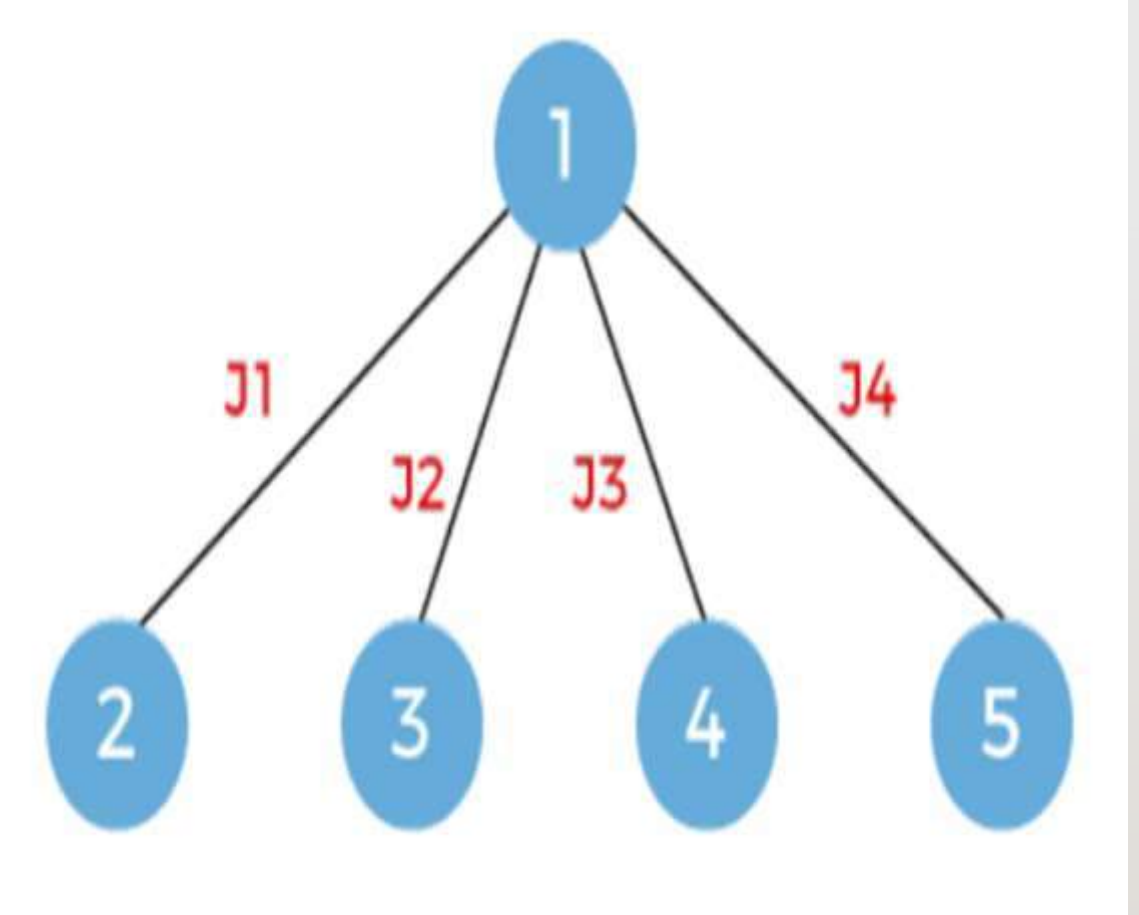
The second way to represent the solution is that the first job is completed, the second and third jobs are not completed, and the fourth job is completed. S2 = {1, 0, 0, 1}

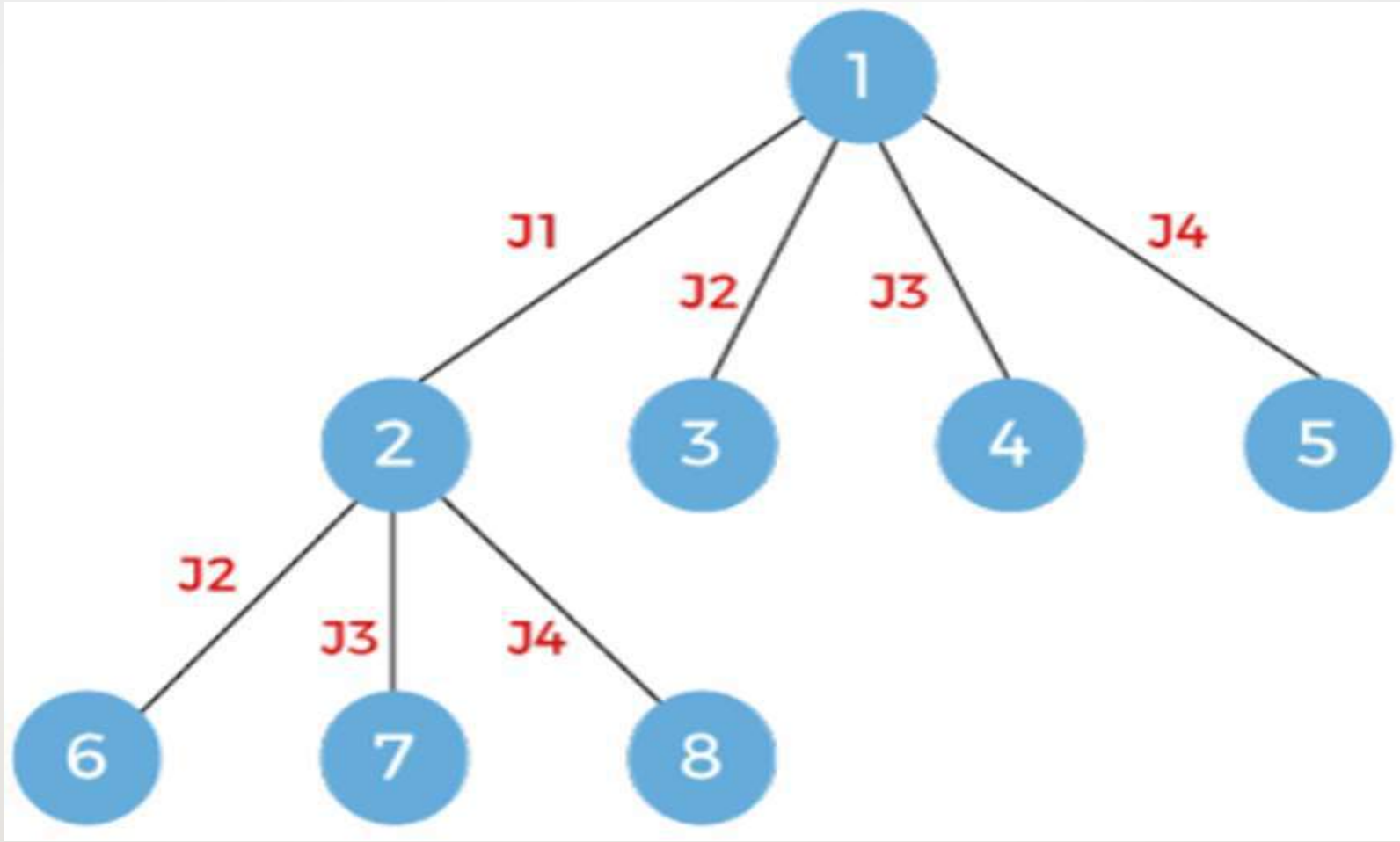
The variable-size solution s1 is the variable-size solution, while the fixed-size solution s2 is the fixed-size solution.

The variable-size solution is s1, and the fixed-size solution is s2.

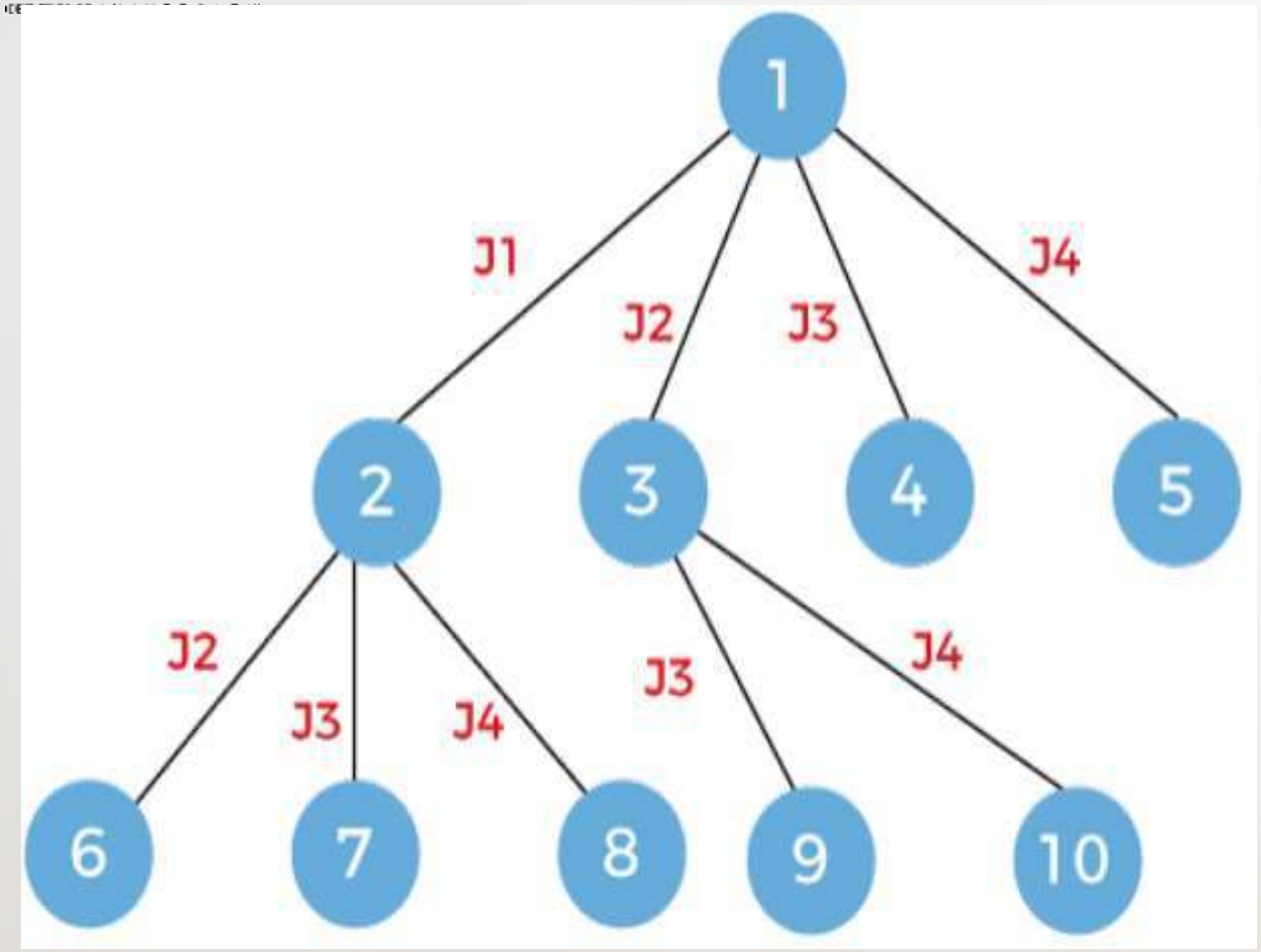
Method 1- FIFO BRANCH & BOUND

- In this case, we consider the first job, then the second job, then the third job, and finally the last job.
- **As shown in the above figure, the breadth first search is used but not the depth first search. Here, we move in a horizontal direction to investigate the solutions.**
- **BACKTRACKING IS DONE IN DEPTH, WHEREAS BRANCH AND BOUND IS DONE IN BREADTH.**

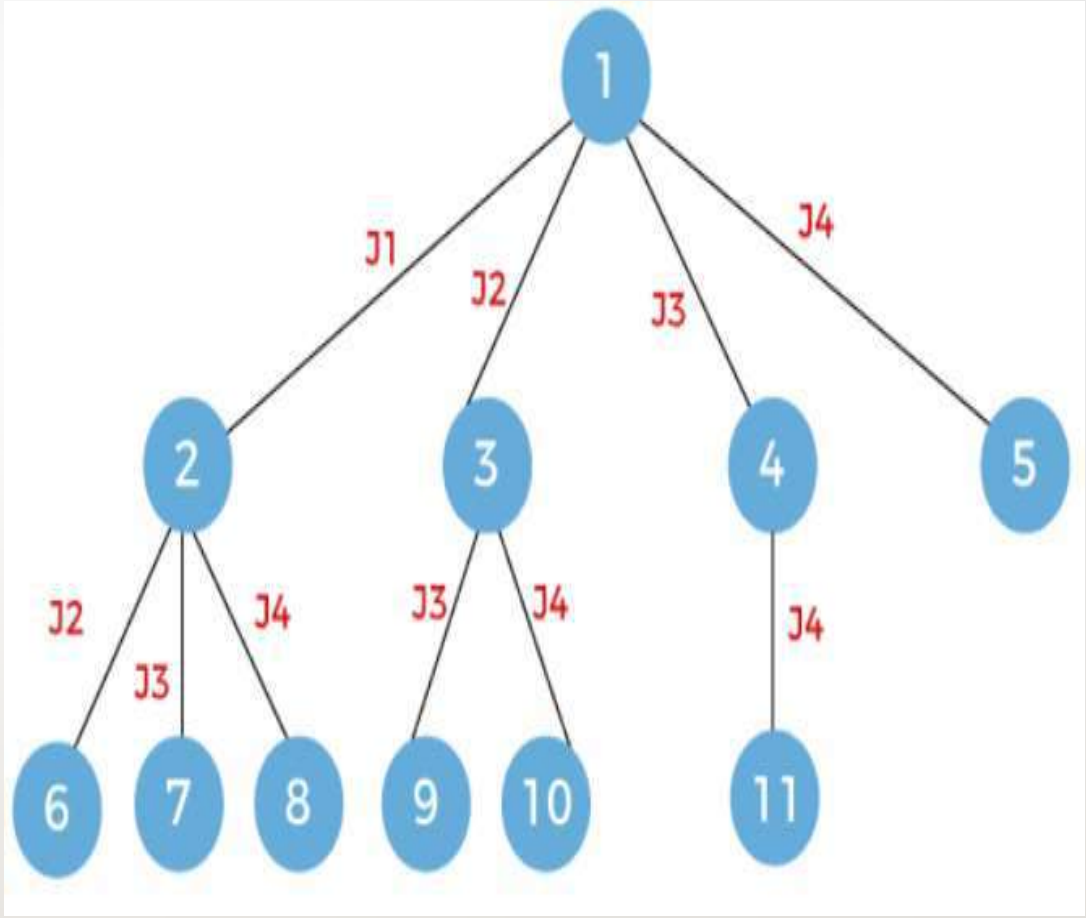




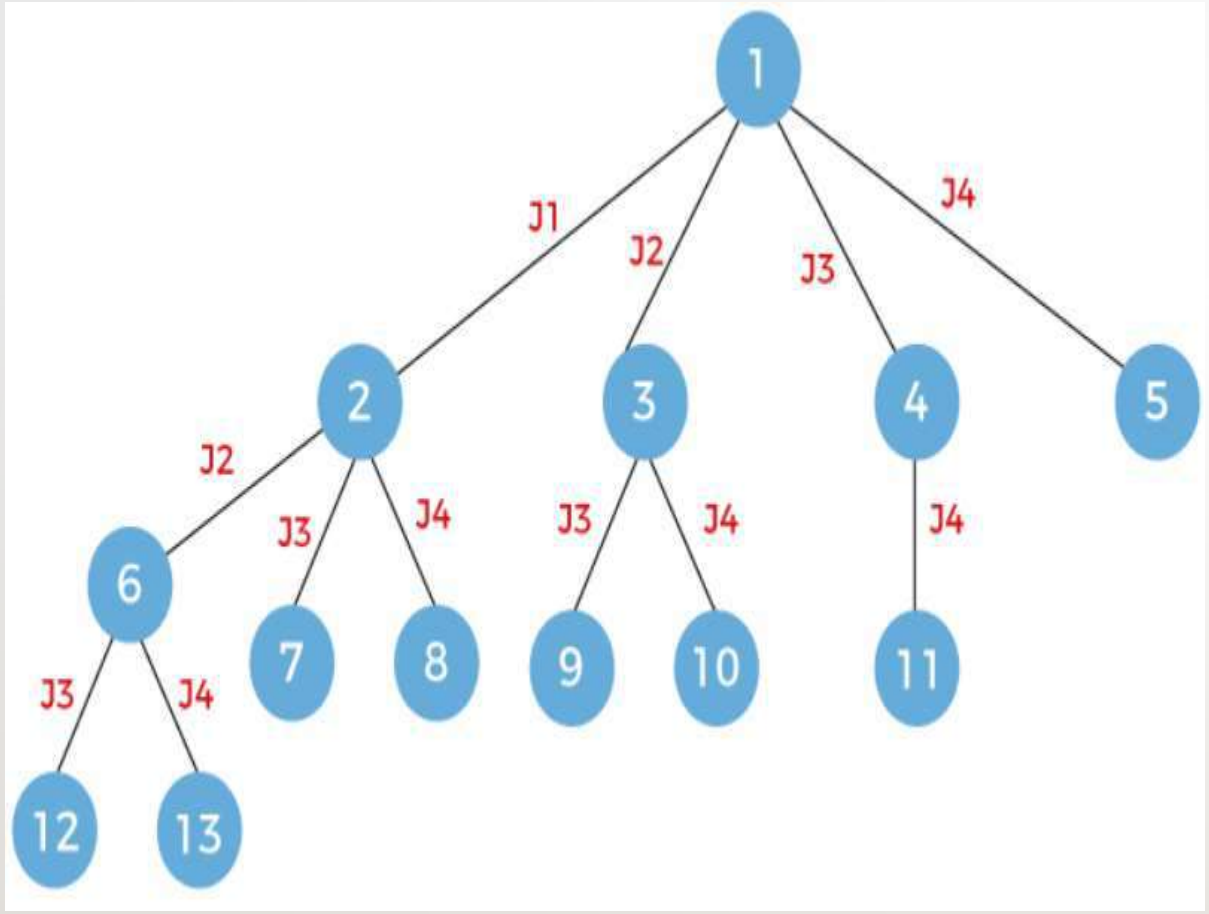
One level has now been completed. After I accept the first job, we can consider j2, j3, or j4. If we follow the path, it says we are doing jobs j1 and j4, so we will disregard jobs j2 and j3.



We will now look at node 3. We are doing job j2 in this case, so we can recognize either job j3 or j4. In this case, we have removed the job j1.

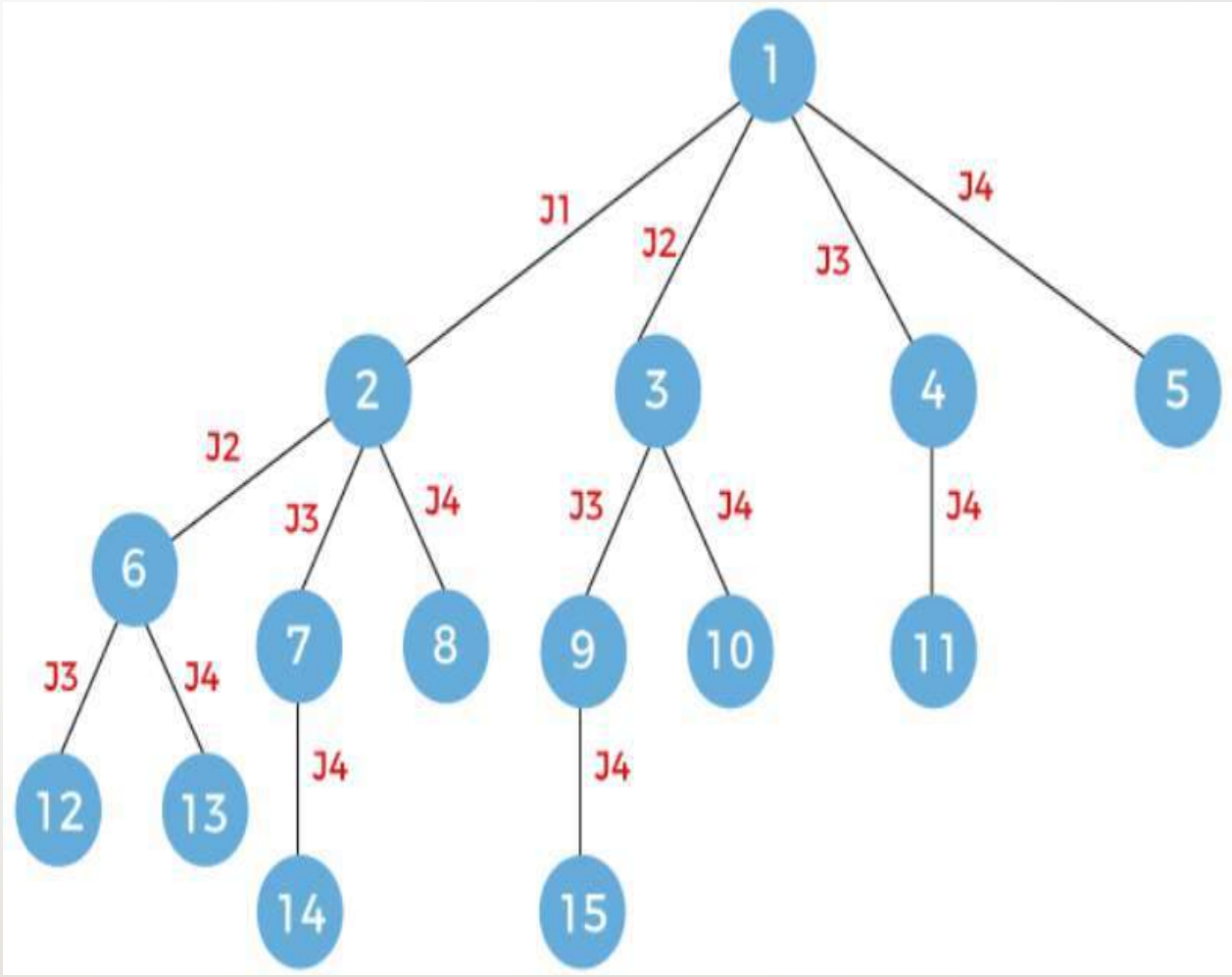


We will now expand node 4. Because we are currently working on job j3, we will only consider job j4.



Node 6 will now be expanded, and the jobs j3 and j4 will be considered.

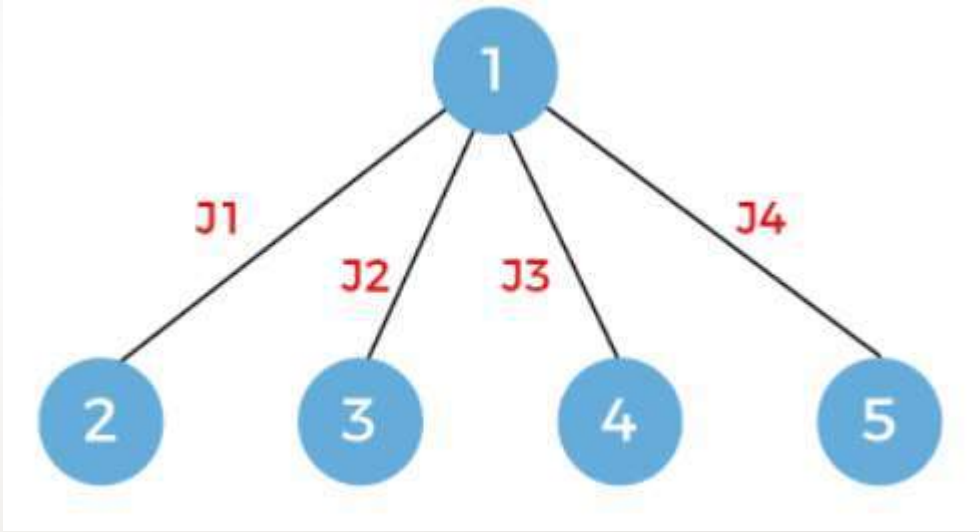
Now we will expand node 7, and we will look at job j4.



Using the queue and the FIFO concept, we explored all of the nodes in the preceding manner.

Method-2 - LIFO BRANCH & BOUND

We'll look at another technique to tackle the problem to get to solution s1.

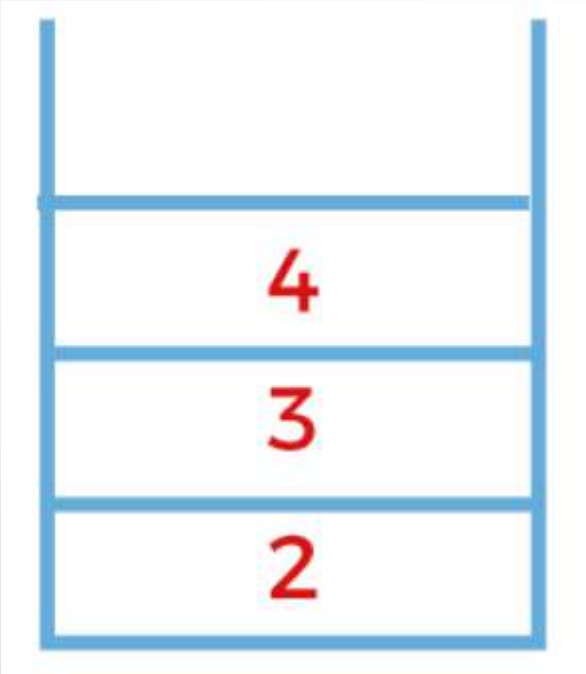


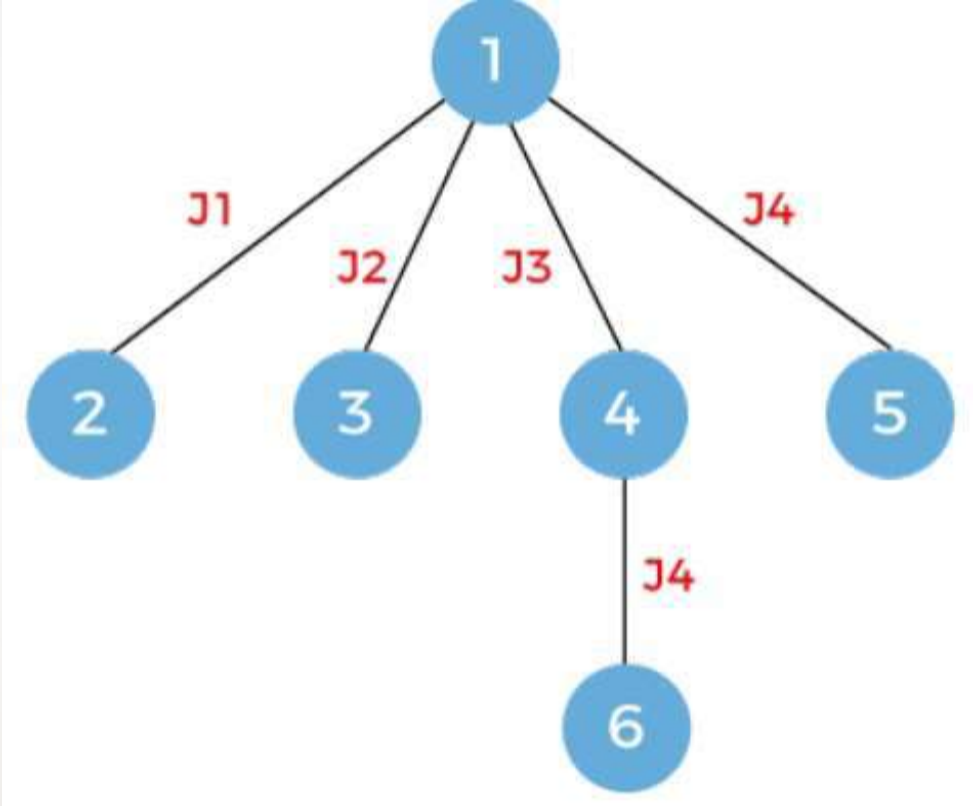
First, consider node 1 as indicated below:

We shall now expand node 1. After expansion, the state space tree might look like this:

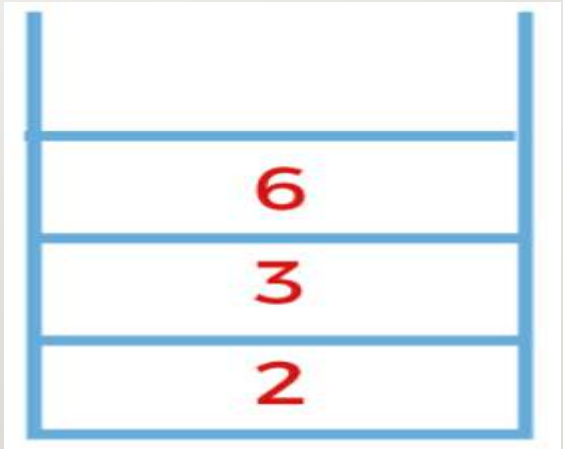
The node will be pushed into the stack indicated below on each expansion:

The expansion would now be based on the node at the top of the stack. Because node 5 appears at the top of the stack, we shall enlarge it. We shall remove node 5 from the stack. There is no additional possibility for enlargement because node 5 is in the last job, i.e., j4.

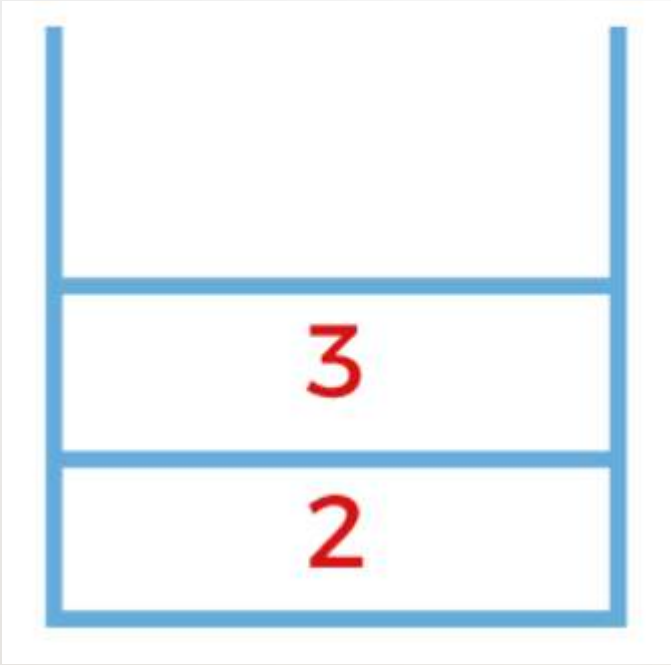


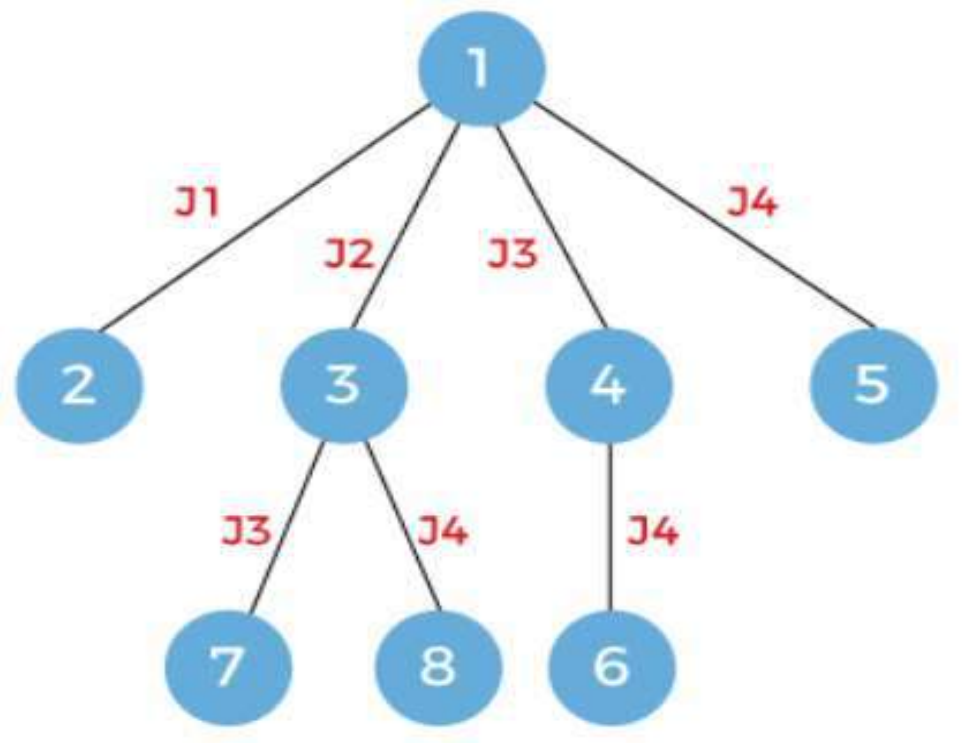


Node 4 is the next node to appear at the top of the stack. Remove node 4 and enlarge it. Job j4 will be examined for expansion, and node 6 will be added to the stack illustrated below:

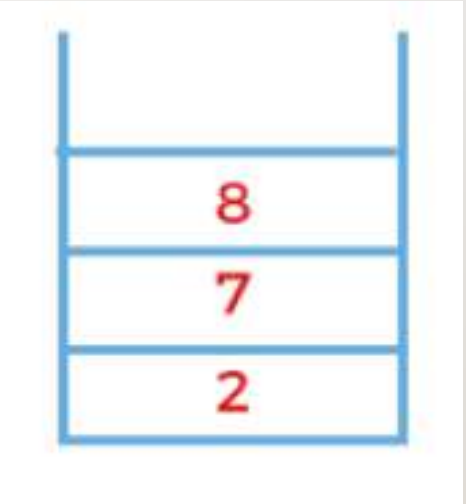


The next node to be expanded is node 6. Remove node 6 and enlarge it. There is no additional possibility for extension because node 6 is in the last job, i.e., j4.

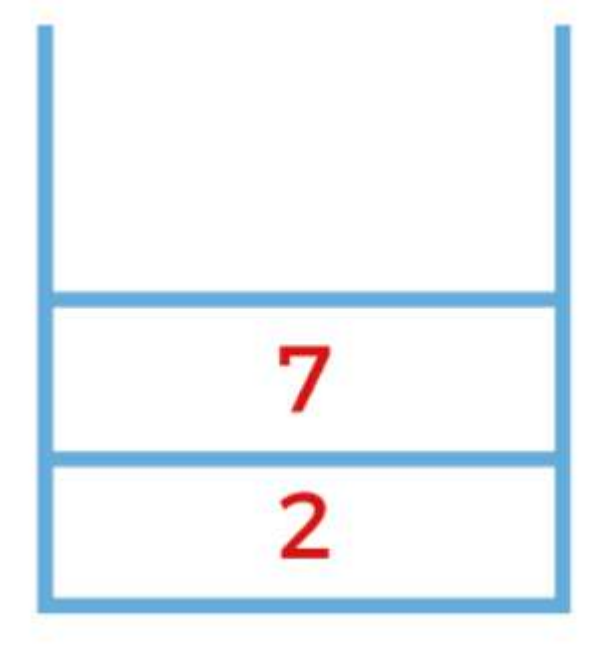


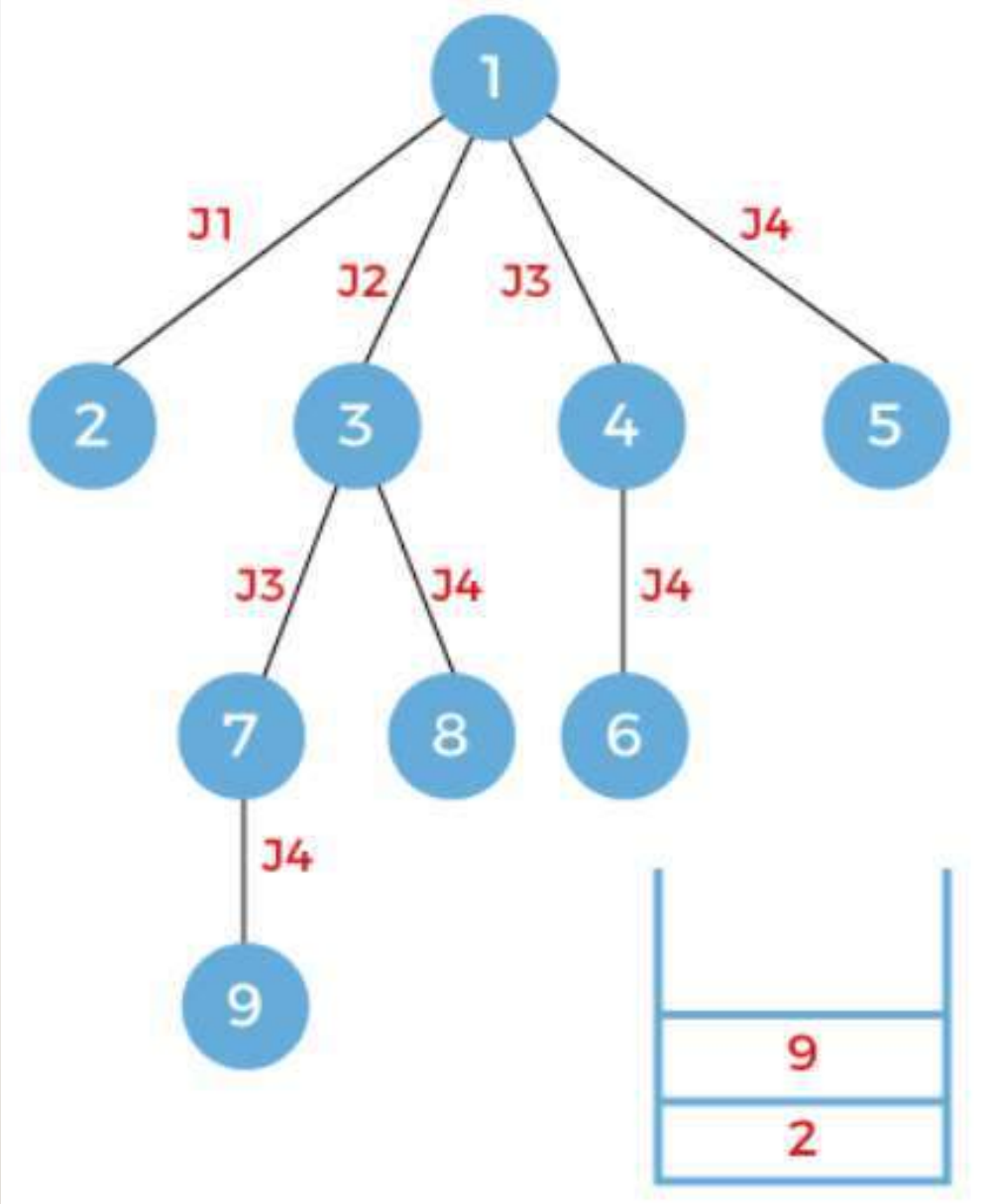


Node 3 is the next to be expanded. Because node 3 works on job j2, it will be enlarged to two nodes, namely 7 and 8, which will work on jobs 3 and 4, respectively. Nodes 7 and 8 will be added to the stack illustrated below:



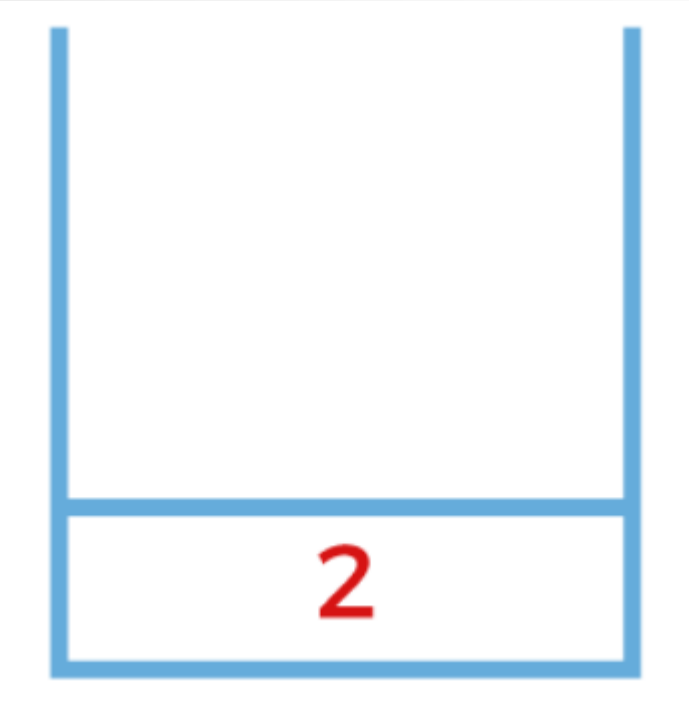
Node 8 is the next node to appear at the top of the stack. Remove node 8 and enlarge it. There is no additional possibility for enlargement because node 8 works on job j4.

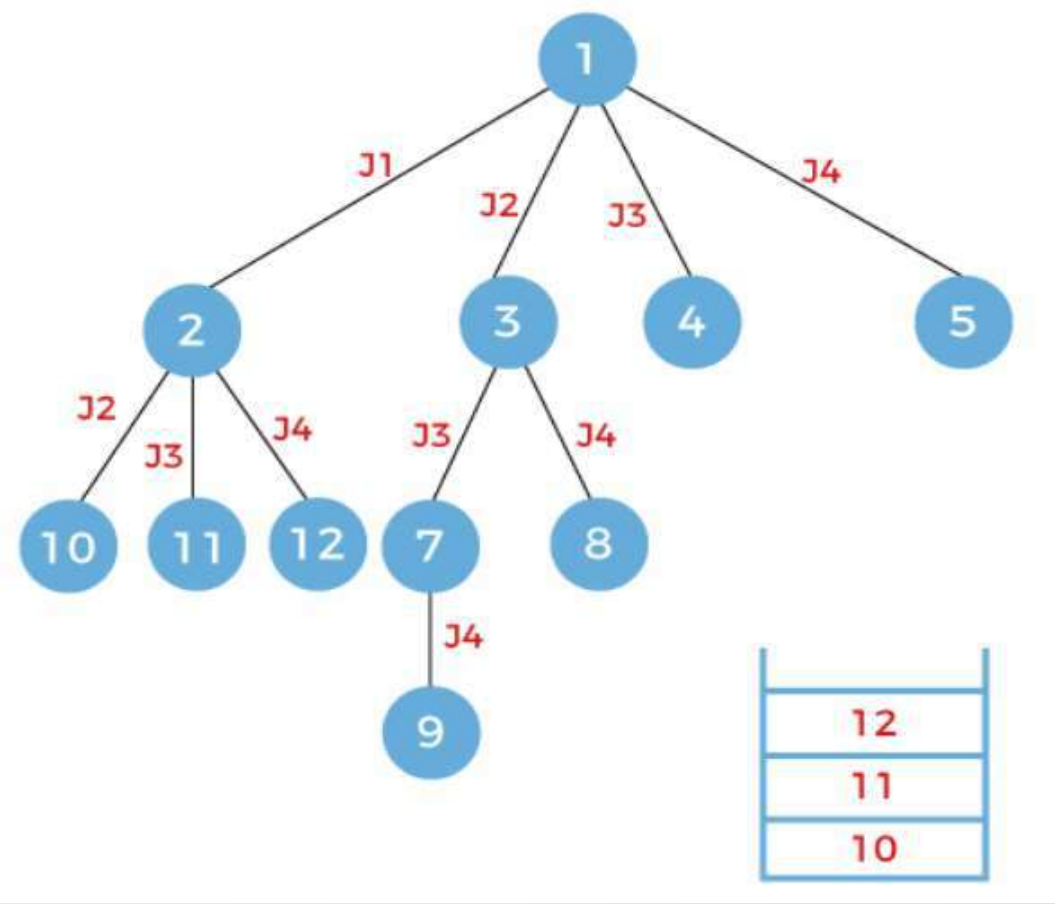




Node 7 is the next node to appear at the top of the stack. Remove node 7 and enlarge it. Because node 7 works on task j3, node 7 will be enlarged to node 9, which works on job j4, as illustrated below, and node 9 will be pushed into the stacks.

Node 9 is the next node to appear at the top of the stack. There is no more room for expansion because node 9 is working on job 4.





Node 2 is the next node to appear at the top of the stack. Because node 2 is working on job j1, it means that node 2 can be expanded further. It can be expanded to three nodes, 10, 11, and 12, each working on job j2, j3, and j4. New nodes will be added to the stack as illustrated below:

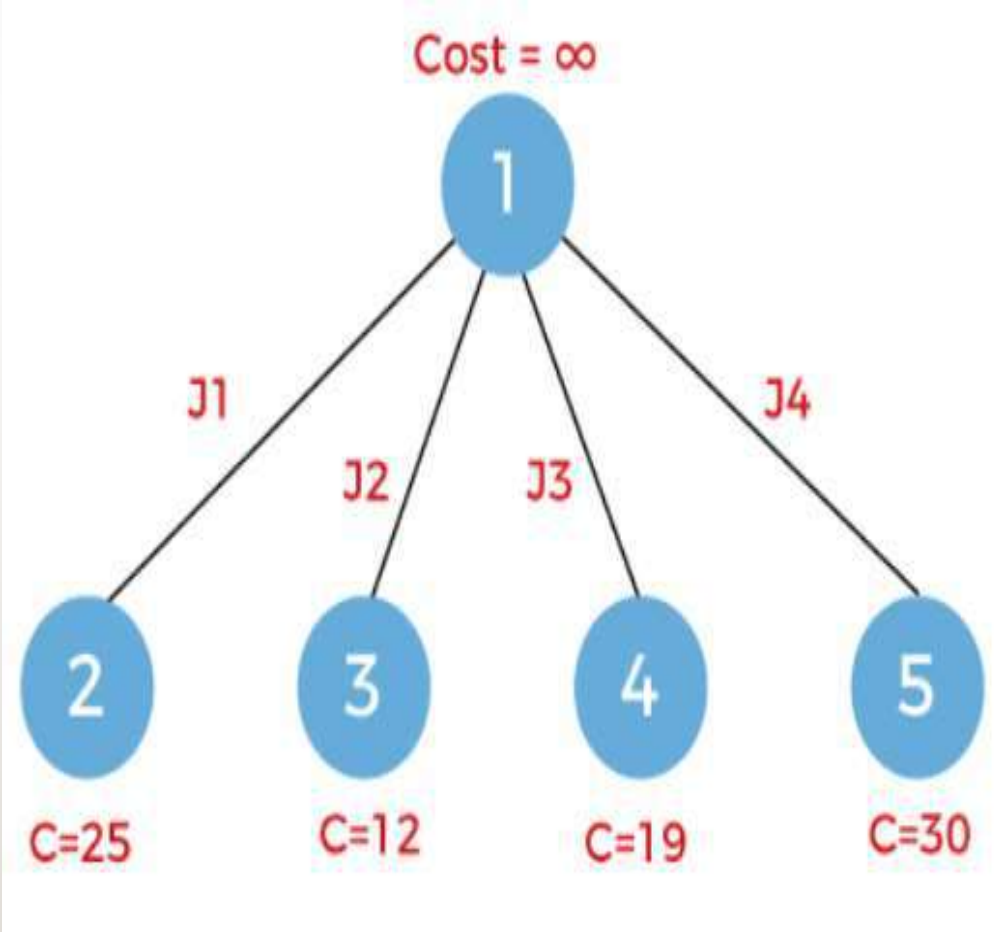
Using the stack and the LIFO concept, we explored all of the nodes in the preceding manner.

Method 3 - LC BRANCH & BOUND

Least cost branch and bound is another strategy that can be utilised to discover the solution. Nodes are investigated using this strategy based on their cost. The cost of the node can be defined using the problem, and the cost function can be defined using the given problem. We can define the node's cost after we define the cost function.

Consider node 1 with cost infinity, as given below:

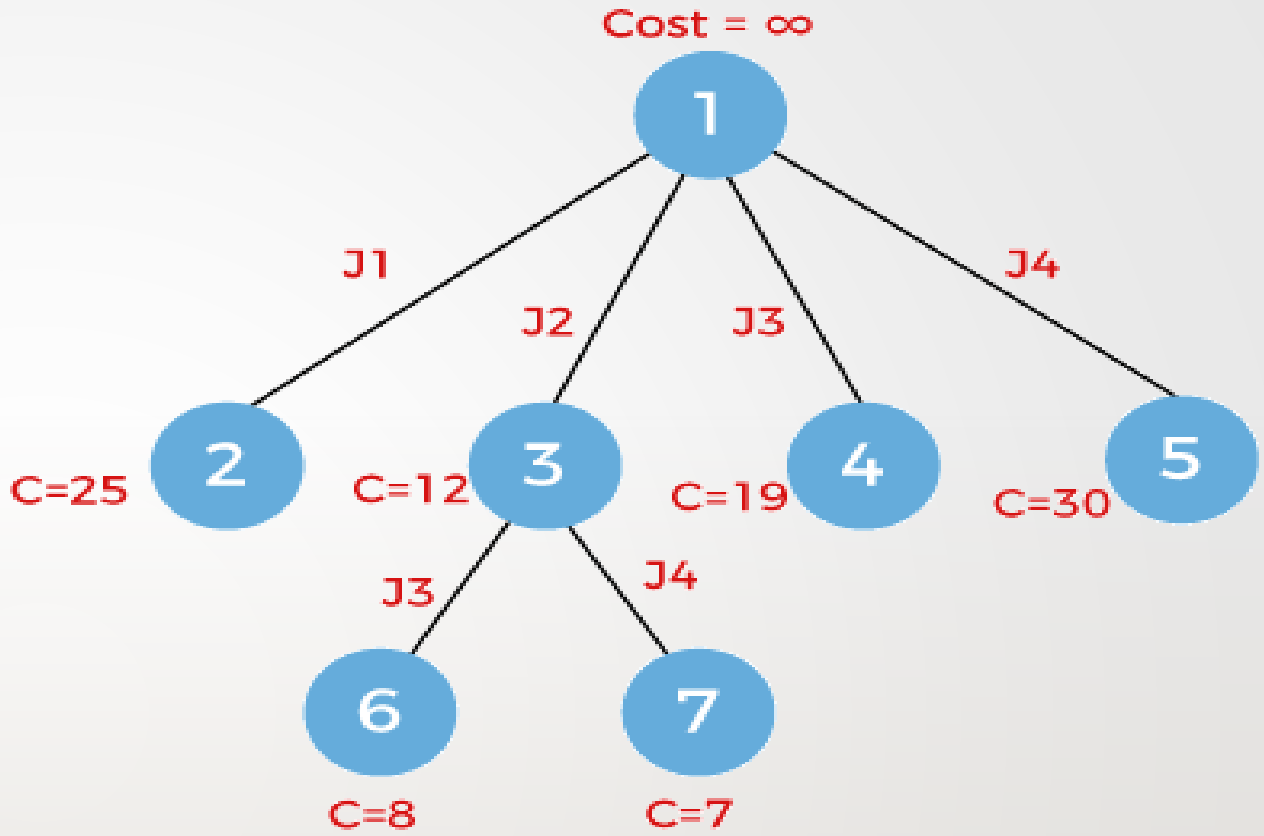
We shall now expand node 1. Node 1 will be divided into four nodes, numbered 2, 3, 4, and 5, as indicated below:



Assume the costs of nodes 2, 3, 4, and 5 are $25, $19, and $30, respectively.

Because it is the least cost branch n bound, we will investigate the node with the lowest cost. In the diagram above, we can see that node 3 has the lowest cost. As a result, we will investigate Node 3 with a cost of $12.

Because node 3 is working on job j2, it will be split into two nodes, 6 and 7, as indicated below:



Node 6 is working on job j3, while node 7 is working on job j4. The cost of node 6 is $8, whereas the cost of node 7 is $7. Now we need to choose the node with the lowest cost. Because node 7 has the lowest cost, we will investigate it. Because node 7 is already working on job j4, there is no room for expansion.