Unit 4

Backtracking:

The Backtracking technique is an algorithm used to solve the decision problem.

Backtracking is the modified process of the brute force approach where the technique efficiently searches for a solution to the problem among all available options.

The brute force approach is nothing but it finds all possible solutions to make satisfaction to the given problem.

This technique follow the brute force method and are used to generate the State Space tree.

A State Space tree is the tree that represents the nodes in all possible states of the problem from a root node to the terminal node.

A backtracking solution can be represented in the form of a tree and it is also called a State Space tree.

Backtracking follow the Depth-first search(DFS).

Backtracking is used to solve problem in which a sequence of objects is chosen from a
specified set so that the sequence satisfies some criterion.

The solution is based on finding one or more vectors that maximize, minimize, or satisfy
a criterion function P (x1, ... , xn).

Form a solution and check at every step if this has any chance of success. If the solution
at any point seems not promising, ignore it.

All solutions require a set of constraints divided into two categories: explicit and implicit
constraints.

Explicit constraints:

- Explicit constraints are rules that restrict each xi to take on values only from a given set.

Explicit constraints depend on the particular instance I of problem being solved. .

- All tuples that satisfy the explicit constraints define a possible solution space for I.

Implicit constraints:

- Implicit constraints are rules that determine which of the tuples in the solution space of I

satisfy the criterion function.

- Thus, implicit constraints describe the way in which the xi’ s must relate to each other

Example:

For 4-queens problem: Let us consider, N = 4. Then 4-Queens Problem is to place eight queens on an 4 x 4 chessboard so that no two “attack”, that is, no two of them are on the same row, column, or diagonal see Figure 1.

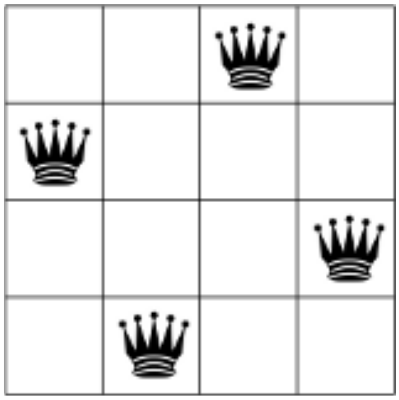
All solutions to the 4-queens problem can be represented as 4-tuples (x1, ... , x4), where xi is the column of the ith row where the ith queen is placed.

In this problem **Explicit** and **Implicit constraints** are as:

 Explicit constraints using 4-tuple formation, for this problem are S= {1, 2, 3, 4}.

 Implicit constraints for this problem are that no two queens can be the same (i.e., all

queens must be on different columns) and no two queens can be on the same diagonal



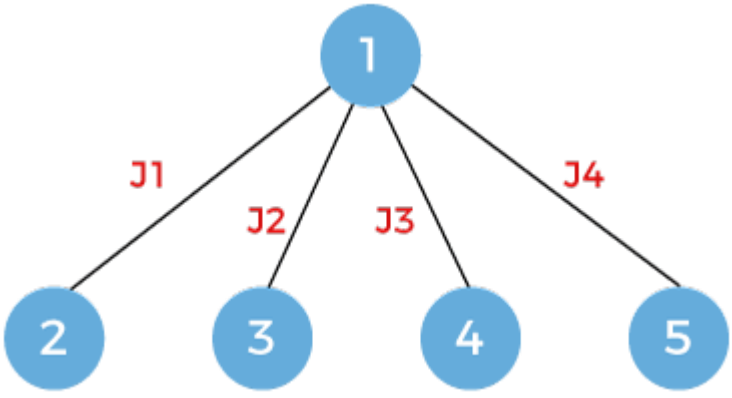
the Branch-N-Bound technique is an algorithm used to solve the optimization problem.

Branch and Bound

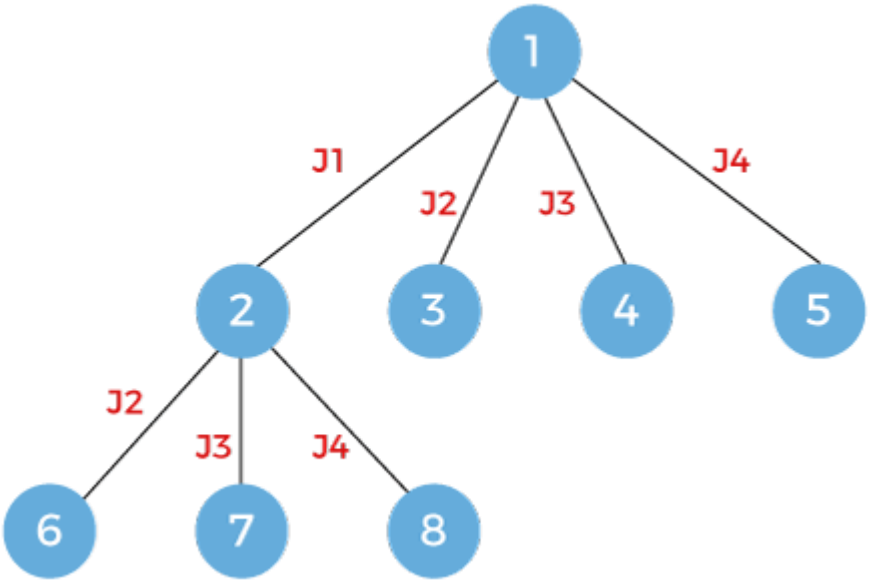
Branch and bound is one of the techniques used for problem solving. It is similar to the backtracking since it also uses the state space tree. It is used for solving the optimization problems and minimization problems.

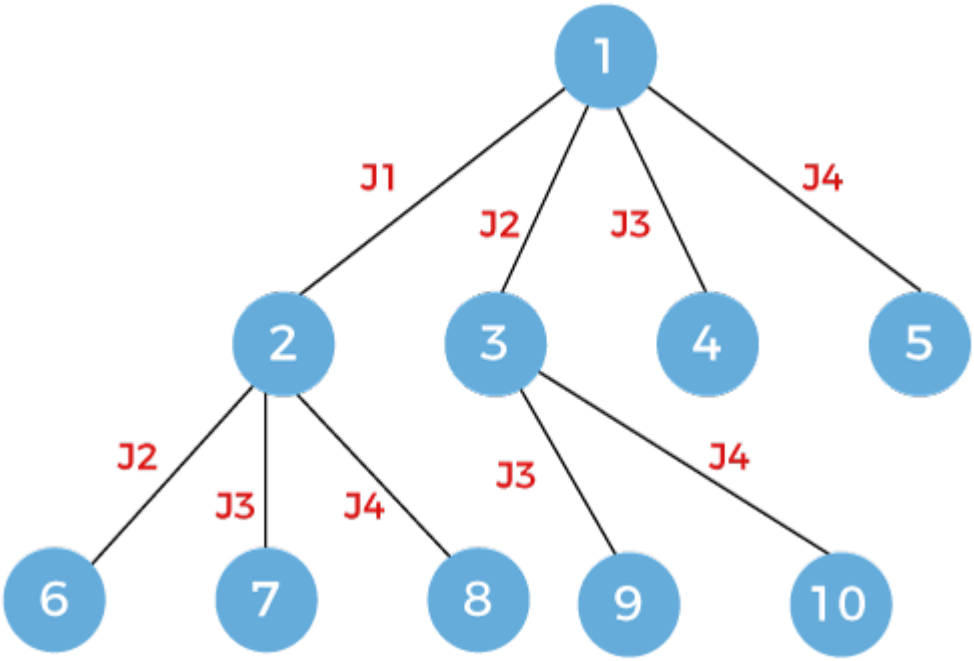
Jobs = {j1, j2, j3, j4}

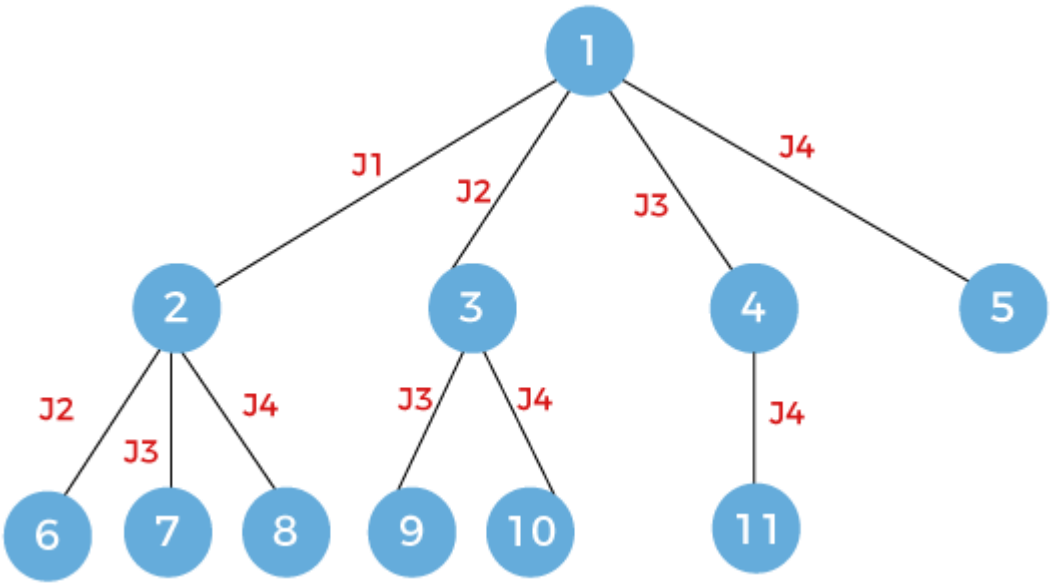
P = {10, 5, 8, 3}

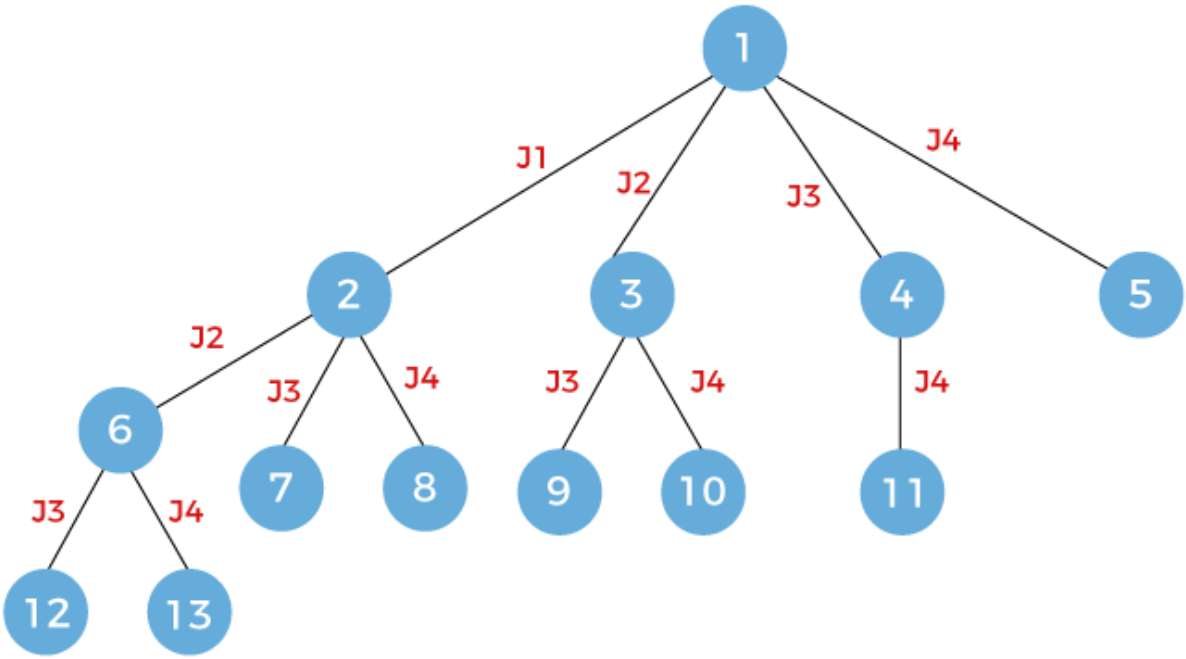


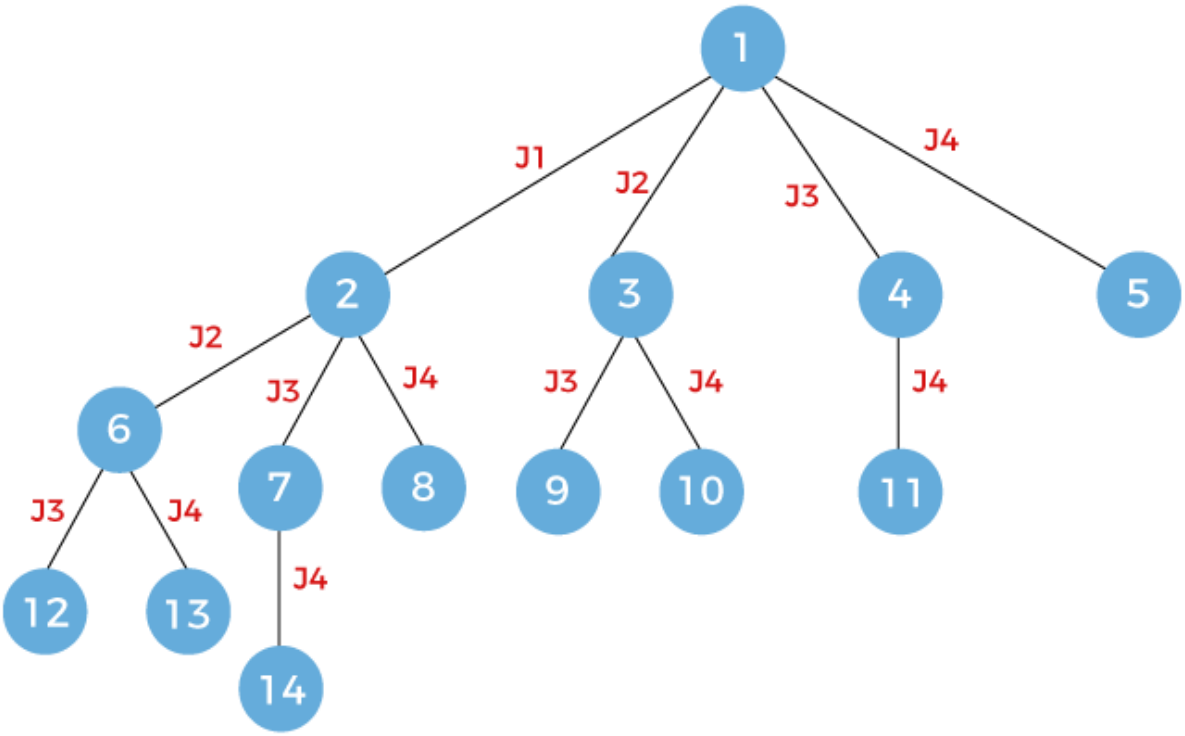
In this case, we first consider the first job, then second job, then third job and finally we consider the last job.


1
J1
J2
J3
J4
2
3
4
5
J2
J3
J4
6
7
8

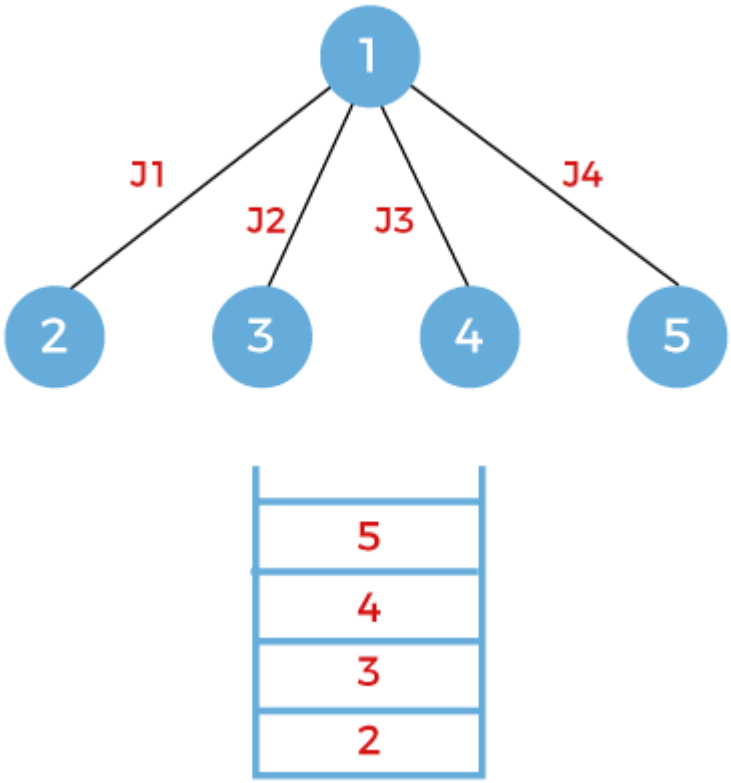

1
J1
J2
J3
J4
2
3
4
5
J2
J3
J4
J3
J4
6
7
8
9
10


1
J1
J2
J3
J4
2
3
4
5
J2
J3
J4
J3
J4
J4
6
7
8
9
10
11


1
J1
J2
J3
J4
2
3
4
5
J2
J3
J4
J3
J4
J4
6
7
8
9
10
11
J3
J4
12
13

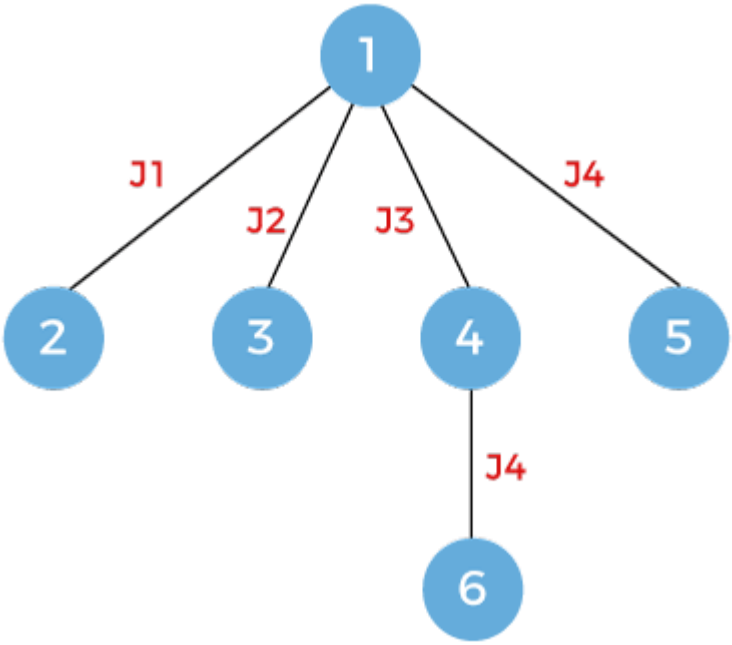


Second Method

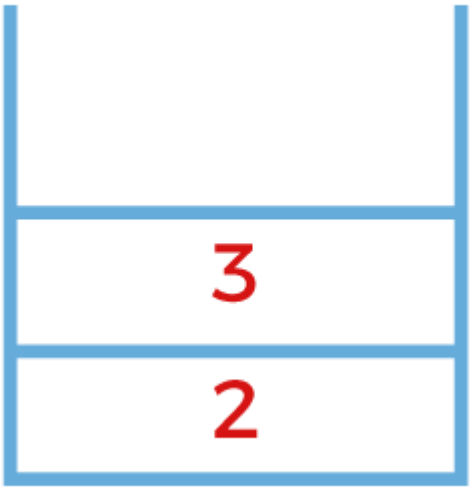


Now the expansion would be based on the node that appears on the top of the stack. Since the node 5 appears on the top of the stack, so we will expand the node 5. We will pop out the node 5 from the stack. Since the node 5 is in the last job, i.e., j4 so there is no further scope of expansion.

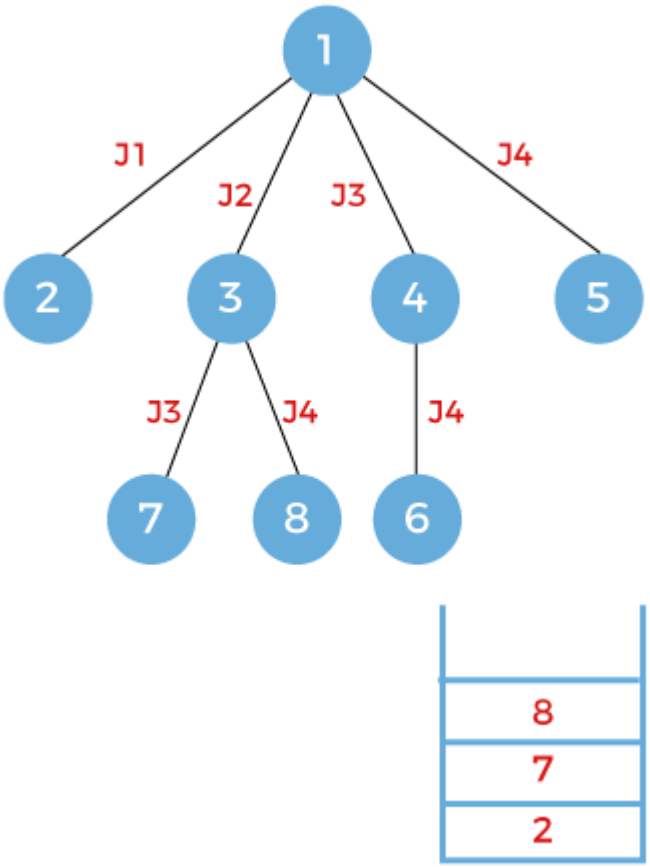
The next node that appears on the top of the stack is node 4. Pop out the node 4 and expand. On expansion, job j4 will be considered and node 6 will be added into the stack shown as below:



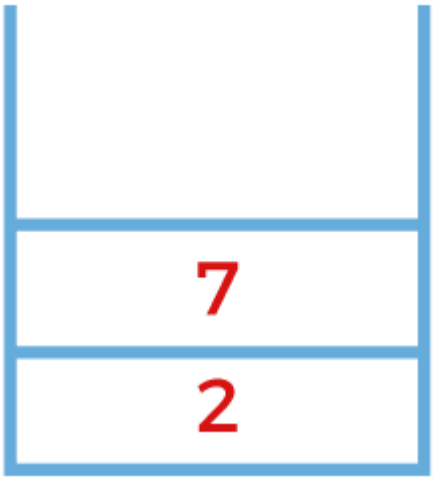
The next node is 6 which is to be expanded. Pop out the node 6 and expand. Since the node 6 is in the last job, i.e., j4 so there is no further scope of expansion.



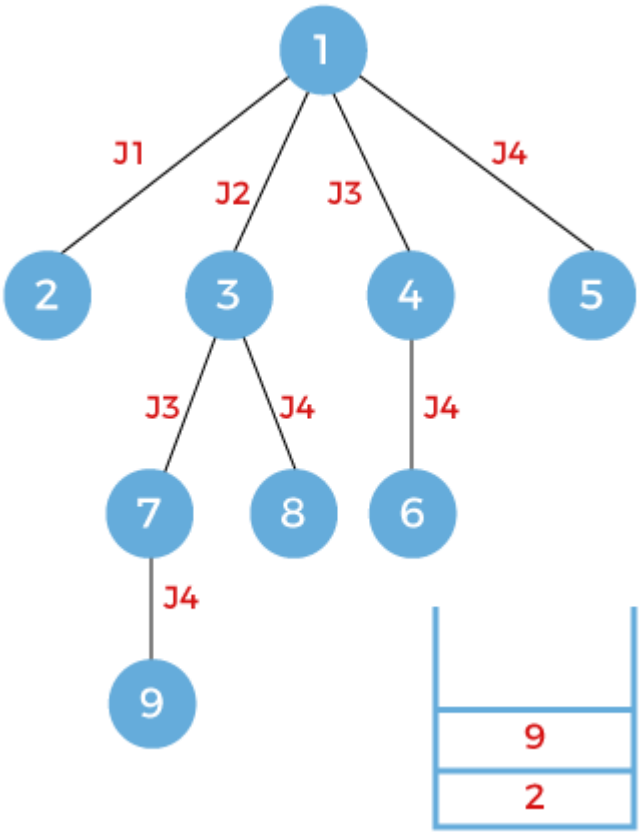
The next node to be expanded is node 3. Since the node 3 works on the job j2 so node 3 will be expanded to two nodes, i.e., 7 and 8 working on jobs 3 and 4 respectively. The nodes 7 and 8 will be pushed into the stack shown as below:



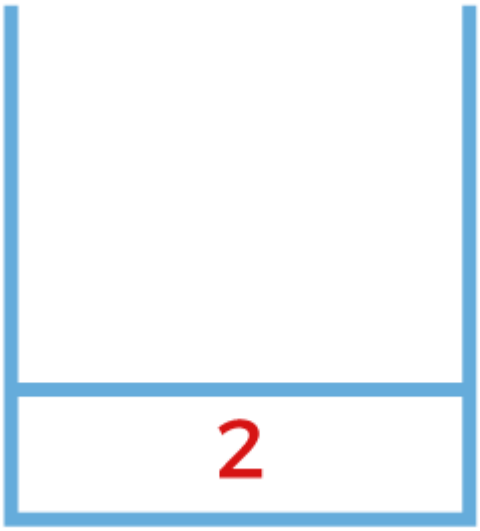
The next node that appears on the top of the stack is node 8. Pop out the node 8 and expand. Since the node 8 works on the job j4 so there is no further scope for the expansion.



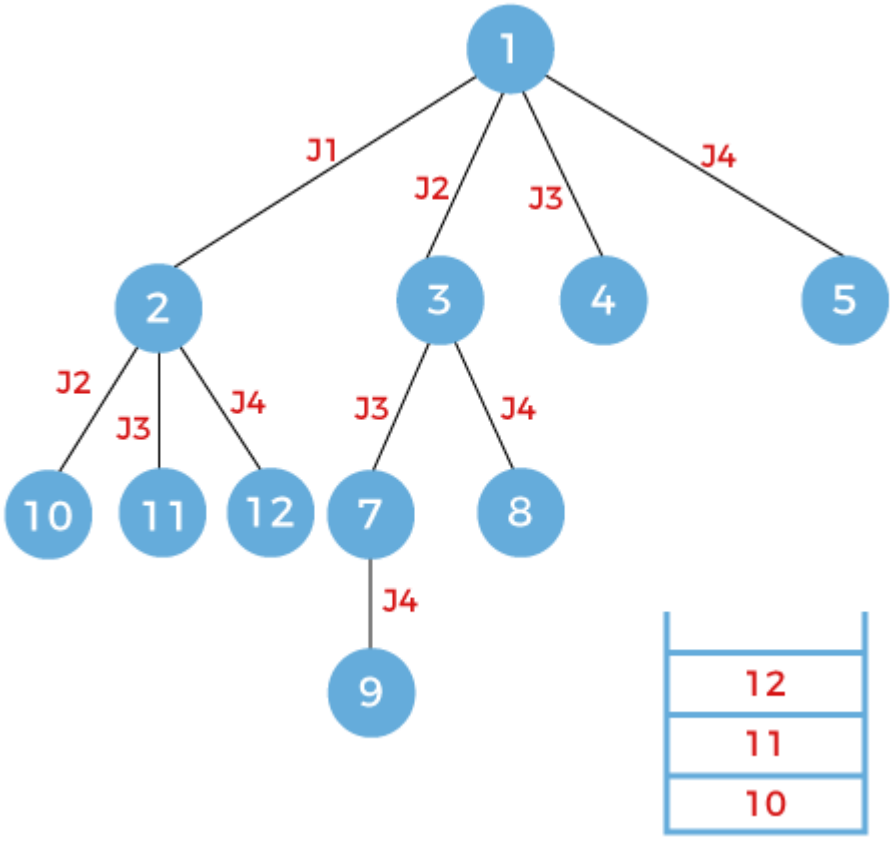
The next node that appears on the top of the stack is node 7. Pop out the node 7 and expand. Since the node 7 works on the job j3 so node 7 will be further expanded to node 9 that works on the job j4 as shown as below and the node 9 will be pushed into the stack.



The next node that appears on the top of the stack is node 9. Since the node 9 works on the job 4 so there is no further scope for the expansion.



The next node that appears on the top of the stack is node 2. Since the node 2 works on the job j1 so it means that the node 2 can be further expanded. It can be expanded upto three nodes named as 10, 11, 12 working on jobs j2, j3, and j4 respectively. There newly nodes will be pushed into the stack shown as below:


1
J1
J2
J3
J4
2
3
4
5
J2
J3
J4
J3
J4
10
11
12
7
8
J4
9
12
11
10