**What is Backtracking?**

Backtracking is nothing but the modified process of the **Brute force approach.** It is a technique where multiple solutions to a problem are available, and it searches for the solution to the problem among all the available options. Let's consider the example of brute force search as backtracking follows the brute force search approach.

Consider the box, and we have three objects of three different colors. One object is of red color, the second object is of green color, and the third object is of blue color. Now, we have to keep these objects inside the box, and we have multiple options. As per the Brute force approach, we have to consider all the options and identify the best option among all the possible options.

**Let's consider all the possible solutions to this problem.**

Suppose we first fill red object, after then green object and then blue object. This is the first solution to this problem. The possible solution is red, green, and blue.



The second solution is to keep the red object, then the blue object, and then the green object. The possible solution red, blue, and green.



The third solution can be to keep the green object, then the red object and then the blue object. The possible solution is green, red and blue.



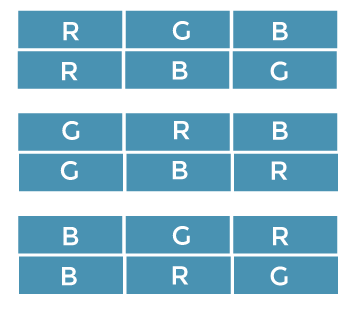
The fourth solution is to keep the green object, then the blue object and then the red object. The possible solution is green, blue and red.



The fifth solution is to keep the blue object, then green object and then red object. The possible solution is blue, green and red.



The sixth solution is to keep the blue object, then the red object and then the green object. The possible solution is blue, red and green.

The above are the possible solutions, and we have to identify the best possible solution out of all the possible solutions. This approach is known as a brute force approach. Backtracking is similar to the brute force approach, but it is a modified process of the brute force approach.

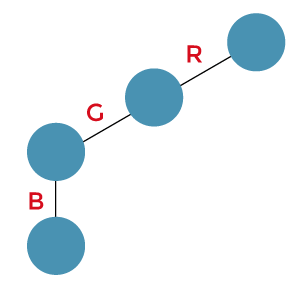
**Let's see how backtracking is different from the brute force approach.**

The above is the figure that shows the possible solutions to the above problem. Now we will see that how these solutions are represented in backtracking.

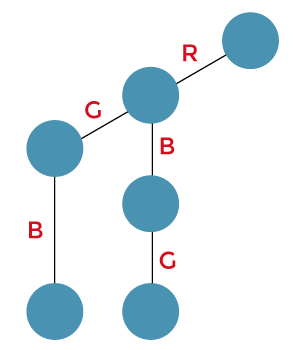
In backtracking, solutions are represented in the form of a tree and that tree is known as a state space tree. Since backtracking follows the DFS, the tree will be formed using DFS, which is known as a **State Space tree.**

**Let's create the tree.**

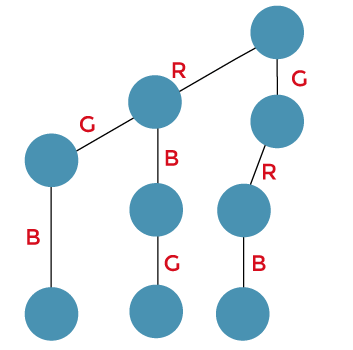
Consider the first solution, i.e., **red, green, blue,** and it can be represented in the form of a tree as shown as below.



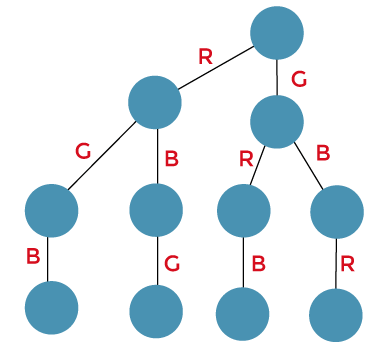
Consider the second solution, i.e., **red, blue, green.** Since there is no more object after blue so we will backtrack and move to the green. Instead of taking green, we first take blue and then green, shown as below:



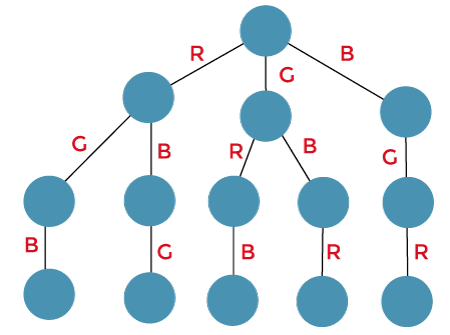
The next solution is **green, red and blue.** Since we cannot explore the green object, so we move back and reach the blue object. We cannot explore the blue object, so we again move back and reach to the red object. Instead of using the red object, we will use the green object. After the green object, we use the red object and then we use the blue object. We cannot explore the blue object further. Now the sequence green, red and blue is formed as shown as below:



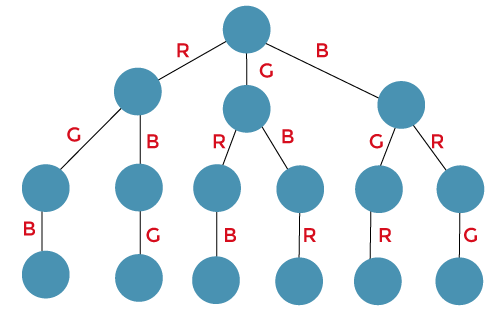
The next solution is **green, blue and red.** Since we cannot explore the blue object, we move back and reach the red object. Instead of using a red color object, we will use the blue color object and then we use the red color object. Now the sequence green, blue and red is formed.



The next solution is **blue, green and red.** Since we cannot explore the red object, so we backtrack and reach the blue object. We cannot explore the blue object so we backtrack and reach the green object. Instead of using the green object, we will use the blue color object then we use the green object and then we use the red color object. The sequence, i.e., blue, green and red, is formed as shown as below:



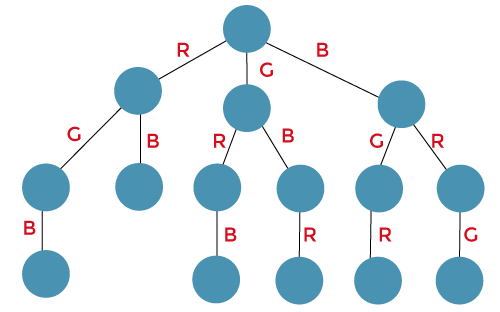
The next solution is **blue, red and green.** Since we cannot explore red object so we backtrack and reach to the green object. Instead of using the green object, we use the red object and then we use the green object.



The above is the state space tree that shows all the possible solutions related to the problem. Therefore, we can say that the state space tree can be used to represent all the solutions in backtracking. Using backtracking, we can solve the problem that has some constraints and we can find the solution based on these constraints. Suppose in the above example; the constraint is blue color object must not be in the middle (bounding function).

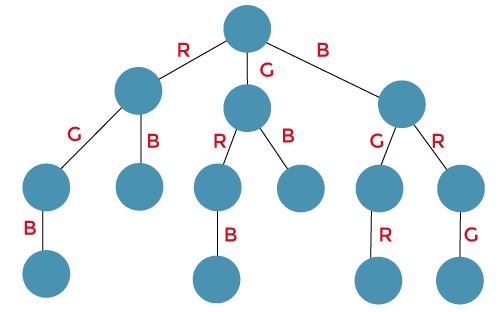
The first solution is **red, green and blue.** Since blue color object is not in the middle so we do need to perform any modification.

The second solution is **red, blue, and green.** Since blue color is in the middle so we will remove the green color as shown as below:



The next solution is **green, red, and blue.** Since blue color object is not in the middle so we do need to perform any modification.

The next solution is **green, blue and red.** Since blue color is in the middle so we will remove the red color as shown as below:



The above is the state space tree that does not have blue object in the middle of the solution.

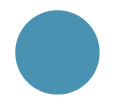
**Branch and Bound**

It is similar to backtracking. The concept branch and bound and backtracking follow the Brute force method and generate the state space tree. But both of them follows different approaches. The way to generate the tree is different.

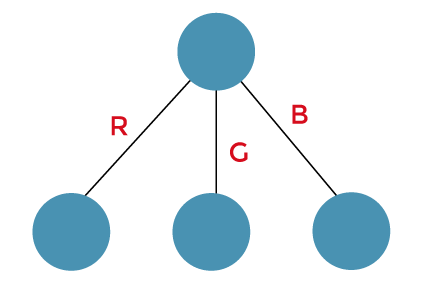
Backtracking follows the DFS, whereas the branch n bound follows the BFS to generate the tree. Now we will understand the branch n bound through an example. As it follows the BFS, so first all the nodes of the same level are added then we move to the next level.

**Consider the same example that we discussed in the backtracking.**

First, we take the root node.



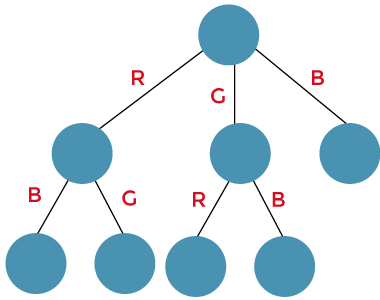
Now we have three possibilities that either we select red, green, or blue objects as shown below. In this case, we have completed the first level.



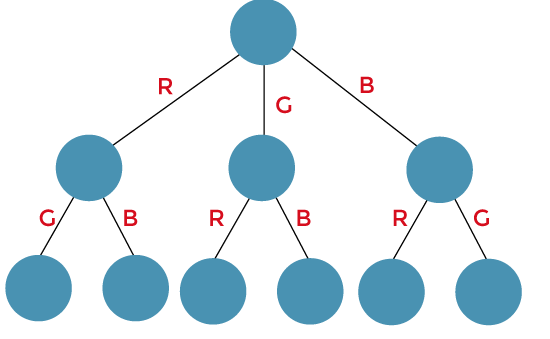
Now we move to the next level.

In the case of red object, we have two possibilities that either we select green or blue object shown as below:

In the case of green object, we have two possibilities that either we select red or blue object shown as below:

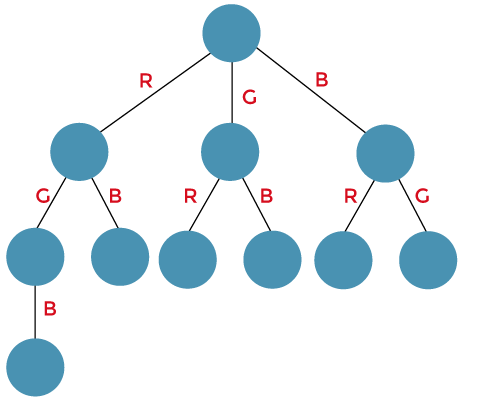


In the case of blue object, we have two possibilities that either we select red or green object shown as below:

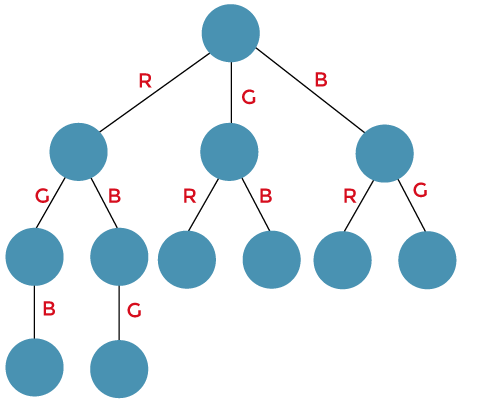


We move to the third level.

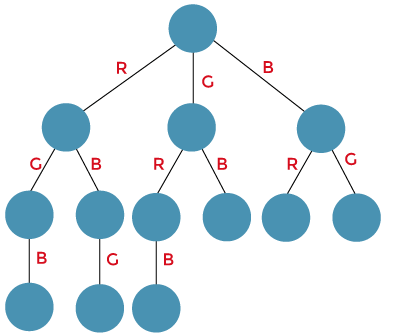
In the case of a green object, only one object, i.e., blue, can be added.



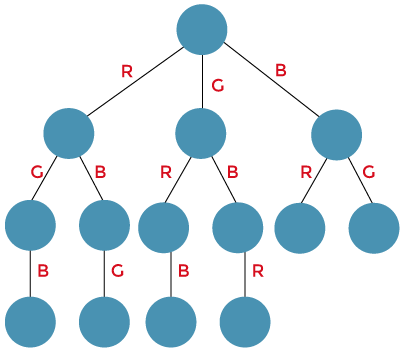
In the case of a blue object, only one object, i.e., green, can be added.



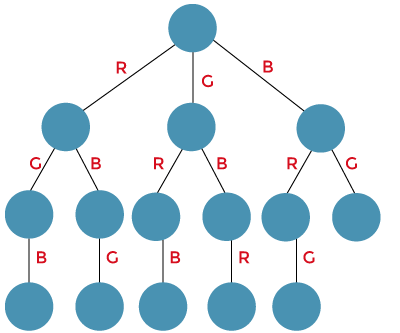
In the case of a red object, only one object, i.e., blue, can be added.



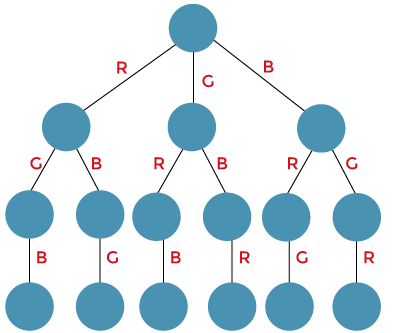
In the case of a blue object, only one object, i.e., red, can be added.



In the case of a green object, only one object, i.e., red, can be added.



In the case of a red object, only one object, i.e., green, can be added.



**Examples:**

The problems that can be solved by using backtracking are:

* 8 Queens problem
* Knapsack problem using backtracking

The problem that can be solved by using branch and bound is:

* Travelling Salesman Problem

|  |  |
| --- | --- |
| **Backtracking** | **Branch and bound** |
| Backtracking is a problem-solving technique so it solves the decision problem. | Branch n bound is a problem-solving technique so it solves the optimization problem. |
| When we find the solution using backtracking then some bad choices can be made. | When we find the solution using Branch n bound then it provides a better solution so there are no chances of making a bad choice. |
| Backtracking uses a Depth first search. | It is not necessary that branch n bound uses Depth first search. It can even use a Breadth-first search and best-first search. |
| The state space tree is searched until the solution of the problem is obtained. | The state space tree needs to be searched completely as the optimum solution can be present anywhere in the state space tree. |
| In backtracking, all the possible solutions are tried. If the solution does not satisfy the constraint, then we backtrack and look for another solution. | In branch and bound, based on search; bounding values are calculated. According to the bounding values, we either stop there or extend. |
| Applications of backtracking are n-Queens problem, Sum of subset. | Applications of branch and bound are knapsack problem, travelling salesman problem, etc. |
| Backtracking is more efficient than the Branch and bound. | Branch n bound is less efficient. |
| It contains the feasibility function. | It contains the bounding function. |
| Backtracking solves the given problem by first finding the solution of the subproblem and then recursively solves the other problems based on the solution of the first subproblem. | Branch and bound solves the given problem by dividing the problem into two atleast subproblems. |

### Differences between Branch n bound and Backtracking