**Branch and Bound Algorithm**

**Branch and bound** is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case.

The branch and bound algorithm technique solves these problems relatively quickly by pruning the search space using upper and lower bounds on the optimal solution.

The branch and bound algorithm consists of three main steps: branching, bounding, and fathoming.

**Branching**

This step involves dividing the search space into smaller and disjoint subsets by applying some branching rules. Each subset corresponds to a node in a decision tree, where the root node represents the entire search space and the leaf nodes represent the feasible solutions.

**Bounding:**

This step involves computing a bound for each node in the decision tree by using some bounding methods. The bound is an estimate of the best possible solution in the sub tree rooted at that node. The bounding methods can be based on heuristic methods, relaxation techniques, or duality theory.

**Fathoming**

This step involves comparing the bound of each node with the current best solution and pruning the node and its sub tree if the bound is worse than the current best solution. This is also called pruning or cutting.

The branch and bound algorithm can be applied to various problems such as 0/1 knapsack problem, 8 puzzle problem, job assignment problem, n queen problem, traveling salesman problem, etc.

The goal of a branch-and-bound algorithm is to find a value *x* that maximizes or minimizes the value of a real-valued function *f*(*x*), called an objective function, among some set *S* of admissible

branch(*I*) produces two or more instances that each represent a subset of *SI*. (Typically, the subsets are [disjoint](https://en.wikipedia.org/wiki/Disjoint_sets) to prevent the algorithm from visiting the same candidate solution twice, but this is not required. However, an optimal solution among *SI* must be contained in at least one of the subsets.)

bound(*I*) computes a lower bound on the value of any candidate solution in the space represented by *I*, that is, bound(*I*) ≤ *f*(*x*) for all *x* in *SI*.

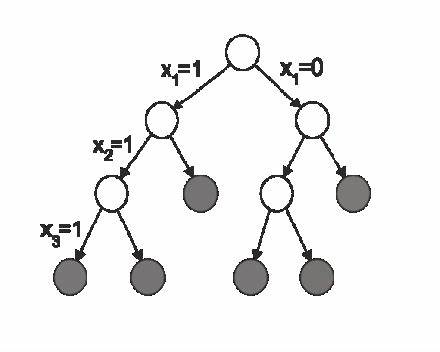
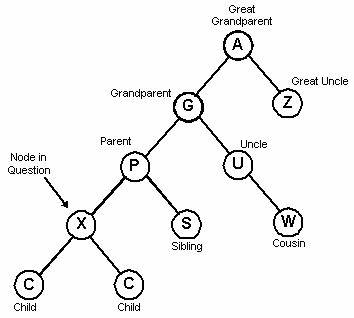
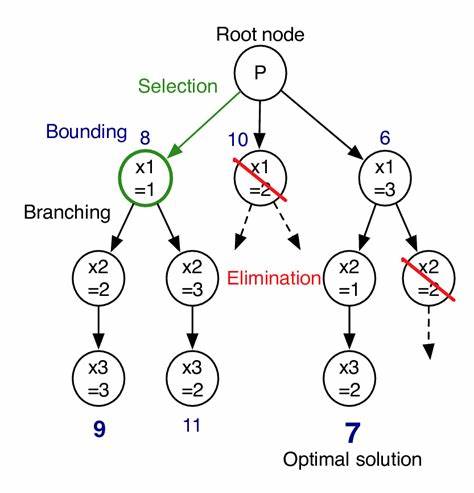
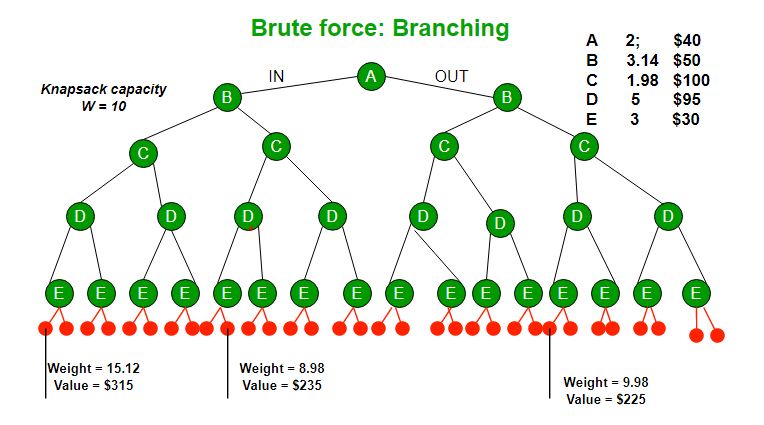
solution(*I*) determines whether *I* represents a single candidate solution. (Optionally, if it does not, the operation may choose to return some feasible solution from among *SI*.) If solution(*I*) returns a solution then *f*(solution(*I*)) provides an upper bound for the optimal objective value over the whole space of feasible solutions.

**HOW TO SOLVE PROBLEMS USING BRANCH AND BOUND ALGORITHM**

To solve a problem using a branch and bound algorithm, you need to follow these steps:

1. Define the problem as an optimization problem with a finite but large number of feasible solutions.
2. Find an upper bound and a lower bound for the optimal solution by using some heuristic methods or relaxation techniques.
3. Construct a rooted decision tree where each node represents a partial or complete solution. The root node represents the entire search space.
4. Select an active node from the tree and explore its children nodes by applying some branching rules. The branching rules should divide the search space into smaller and disjoint subsets.
5. For each child node, compute its bound by using some bounding methods. The bounding methods should estimate the best possible solution in the sub tree rooted at that node.
6. Compare the bound of each child node with the current best solution. If the bound is worse than the current best solution, prune the node and its sub tree. This is called fathoming.
7. Repeat steps 4 to 6 until there are no more active nodes in the tree or a termination criterion is met. The current best solution is the optimal solution.

**SOME IMAGES OF BRANCH AND BOUND ALGORITHM**



**SAMPLE PROBLEMS AND THEIR BRANCH AND BOUND ALGORITHM**

Some sample problem can be

* 0/1 KNAPSACK problem
* Job assignment problem
* Travelling sales man problem
* N queen problem

We will look at the first one how to solve problems using this method by providing the source code .