**Branch and Bound method, 0/1 Knapsack problem**

**Branch and bound**

**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.

We discussed different approaches to solve above problem and saw that the Branch and Bound solution is the best suited method when item weights are not integers.

In this post implementation of Branch and Bound method for 0/1 knapsack problem is discussed.

**How to find bound for every node for 0/1 Knapsack?**

The idea is to use the fact that the Greedy approach provides the best solution for Fractional Knapsack problem.

To check if a particular node can give us a better solution or not, we compute the optimal solution (through the node) using Greedy approach. If the solution computed by Greedy approach itself is more than the best so far, then we can’t get a better solution through the node.

**Complete Algorithm:**

1. Sort all items in decreasing order of ratio of value per unit weight so that an upper bound can be computed using Greedy Approach.
2. Initialize maximum profit, maxProfit = 0
3. Create an empty queue, Q.
4. Create a dummy node of decision tree and enqueue it to Q. Profit and weight of dummy node are 0.
5. Do following while Q is not empty.
   * Extract an item from Q. Let the extracted item be u.
   * Compute profit of next level node. If the profit is more than maxProfit, then update maxProfit.
   * Compute bound of next level node. If bound is more than maxProfit, then add next level node to Q.
   * Consider the case when next level node is not considered as part of solution and add a node to queue with level as next, but weight and profit without considering next level nodes.

**Illustration**:

Input:

// First thing in every pair is weight of item

// and second thing is value of item

Item arr[] = {{2, 40}, {3.14, 50}, {1.98, 100},

{5, 95}, {3, 30}};

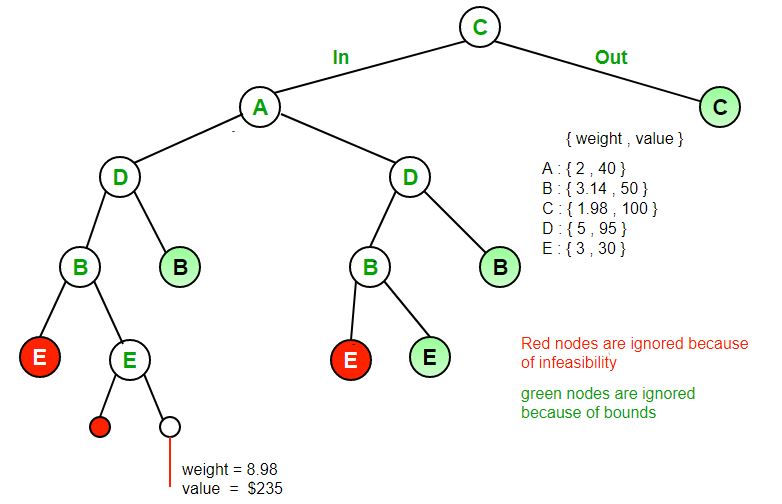
Knapsack Capacity W = 10

Output:

The maximum possible profit = 235

Below diagram shows illustration. Items are

considered sorted by value/weight.



**Note :**  The image doesn't strictly follow the algorithm/code as there is no dummy node in the

image.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.geeksforgeeks.org/implementation-of-0-1-knapsack-using-branch-and-bound/>

**Lecture Video:**

1. <https://youtu.be/dQr4wZCiJJ4>
2. <https://youtu.be/yV1d-b_NeK8>

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**