DAA-Practical - 4

CODE

```
import java.util.*;

class Item {
    int value, weight;
    double valuePerWeight;

    Item(int value, int weight) {
        this.value = value;
        this.weight = weight;
        this.valuePerWeight = (double) value / weight;
    }
}

class Node {
    int level;
    int profit;
    int weight;
    double bound;
}

public class KnapsackBranchBound {

    static double bound(Node u, int n, int W, List<Item> items) {
        if (u.weight >= W)
            return 0;

        double profitBound = u.profit;
        int j = u.level + 1;
        int totalWeight = u.weight;

        while (j < n && totalWeight + items.get(j).weight <= W) {
            totalWeight += items.get(j).weight;
            profitBound += items.get(j).value;
            j++;
        }

        if (j < n) {
            profitBound += (W - totalWeight) * items.get(j).valuePerWeight;
        }

        return profitBound;
```

```
}

static int knapsackBB(int W, int[] val, int[] wt, int n) {
    List<Item> items = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        items.add(new Item(val[i], wt[i]));
    }

    items.sort((a, b) -> Double.compare(b.valuePerWeight, a.valuePerWeight));

    Queue<Node> Q = new LinkedList<>();
    Node u = new Node();
    Node v = new Node();

    u.level = -1;
    u.profit = 0;
    u.weight = 0;
    u.bound = bound(u, n, W, items);

    int maxProfit = 0;
    Q.add(u);

    while (!Q.isEmpty()) {
        u = Q.poll();

        if (u.bound > maxProfit) {
            v = new Node();
            v.level = u.level + 1;
            v.weight = u.weight + items.get(v.level).weight;
            v.profit = u.profit + items.get(v.level).value;

            if (v.weight <= W && v.profit > maxProfit) {
                maxProfit = v.profit;
            }

            v.bound = bound(v, n, W, items);
            if (v.bound > maxProfit)
                Q.add(v);

            v = new Node();
            v.level = u.level + 1;
            v.weight = u.weight;
            v.profit = u.profit;
            v.bound = bound(v, n, W, items);
```

```
                if (v.bound > maxProfit)
                    Q.add(v);
            }
        }

        return maxProfit;
    }

    public static void main(String[] args) {
        int n = 4;
        int[] val = {60, 100, 120, 80};
        int[] wt = {10, 20, 30, 40};
        int W = 50;

        System.out.println("Maximum value using Branch and Bound: " + knapsackBB(W, val, wt,
n));
    }
}
```

OUTPUT

```
Maximum value using Branch and Bound: 220
```