

Knapsack Problem

```
import java.util.List;

public class KnapsackProblem01 {

    public static void main(String[] args) {

        int capacity = 8;
        int[] values= {1,2,5,6};
        int[] weights = {2,3,4,5};
        int n = values.length;

        int maxValue= knapsack(capacity,weights,values, n);
        System.out.println("Maximum value that can be obtained :"+ maxValue);

    }

    private static int knapsack(int capacity, int[] weights, int[] profits, int n) {
        int[][] t =new int[n+1][capacity+1];

        for(int rownum =0 ;rownum<=n; rownum++) {
            for(int colnum =0; colnum <=capacity ; colnum++) {
                if(rownum ==0 || colnum ==0) {
                    t[rownum][colnum] =0;
                }else if(weights[rownum-1] <= colnum) {
                    t[rownum][colnum] = Math.max(t[rownum-1][colnum],
profits[rownum -1] +
                    t[rownum -1][colnum -weights[rownum-1]]);

                }else {
                    t[rownum][colnum] = t[rownum-1][colnum];
                }
            }
        }

        List<Integer> itemsIncluded =
findItemsIncluded(t,weights,profits,n,capacity);
        System.out.println("Items included in the knapsack :" + itemsIncluded);
        return t[n][capacity];
    }

    private static List<Integer> findItemsIncluded(int[][] t, int[] weights, int[]
profits, int n, int capacity) {

        return null;
    }

}
```

```
Dijkstra.java LCS.java Kruskal.java UnionFind.java LongestCommo... *KnapsackPro... »s
18
19 private static int knapsack(int capacity, int[] weights, int[] profits, int n) {
20     int[][] t = new int[n+1][capacity+1];
21
22     for(int rownum = 0; rownum <= n; rownum++) {
23         for(int colnum = 0; colnum <= capacity; colnum++) {
24             if(rownum == 0 || colnum == 0) {
25                 t[rownum][colnum] = 0;
26             } else if(weights[rownum-1] <= colnum) {
27                 t[rownum][colnum] = Math.max(t[rownum-1][colnum], profits[rownum-1] +
28                     t[rownum-1][colnum - weights[rownum-1]]);
29             } else {
30                 t[rownum][colnum] = t[rownum-1][colnum];
31             }
32         }
33     }
34
35     List<Integer> itemsIncluded = findItemsIncluded(t, weights, profits, n, capacity);
36     System.out.println("Items included in the knapsack : " + itemsIncluded);
37     return t[n][capacity];
38 }
39
40 private static List<Integer> findItemsIncluded(int[][] t, int[] weights, int[] profits, int n, int
41     return null;
42 }
43
44 }
45
46 }
```

com.wipro.dynamicprog
KnapSackProblem01
main(String[]): void
knapsack(int, int[], int[], int): int
findItemsIncluded(int[], int[], int): List<Integer>

Markers Properties Terminal Console Coverage
<terminated> KnapSackProblem01 [Java Application] C:\Users\Nikita\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 4
Items included in the knapsack : null
Maximum value that can be obtained : 8