Lab1

Problem 1. Knapsack Optimization Problem

Please find attached java code. Problem1.java

```
import java.util.Arrays;
public class Problem1 {
   static int[] binaryDigits;

   public static void knapsack(int n, String[] s, int[] w, int[] v, int W){
      int solutionMaxWeight = 0;
      int solutionMaxValue = 0;
      String[] solutionItems = new String[n];

      int powerOfTwo = (int) Math.pow(2, n);
      System.out.println(powerOfTwo);

      for (int decimal_number = 1; decimal_number < powerOfTwo; decimal_number++) {

         System.out.print(binaryConversion(decimal_number, 0));
         System.out.print(Arrays.toString(binaryDigits));

         int sumWeight = 0;
         int sumValue = 0;

         for (int j=0; j<n; j++) {
            sumWeight += binaryDigits[j]*w[j];
            sumValue += binaryDigits[j]*v[j];
         }

         if (sumWeight<=W){
            if (sumValue>solutionMaxValue) {
               solutionMaxWeight = sumWeight;
               solutionMaxValue = sumValue;
               for (int i=0; i<n; i++){
                  if (binaryDigits[i]==1) solutionItems[i]=s[i];
                  else solutionItems[i]=" ";
               }
            }
         }
         System.out.print(" , " + sumWeight);
         System.out.println(" , " + sumValue);
      }

      System.out.println("Solution: Value=" + solutionMaxValue + " Weight = " + solutionMaxWeight);
      System.out.println(Arrays.toString(solutionItems));


   }
```

```
    public static void main(String args[]) {

        int n = 6; // Please change item number here
        int maxWeight = 60; // maximum weight
        String[] items = new String[]{ "#1", "#2", "#3", "#4", "#5", "#6", "#7", "#8", "#9", "#10"};
        int[] weightArray = new int[]{ 10, 21, 13, 24, 15, 21, 8, 17, 6, 3};
        int[] valueArray = new int[]{ 13, 17, 12, 5, 19, 4, 25, 30, 7, 22 };

        binaryDigits = new int[n];
        knapsack(n, items, weightArray, valueArray, maxWeight);

    }

    // Decimal to binary conversion using recursion
    static int binaryConversion(int decimal_number, int i) {

        if (decimal_number == 0) {
            if (i < binaryDigits.length) binaryDigits[i] = 0;
            return 0;
        } else {
            {
                binaryDigits[i] = decimal_number % 2;
                return (decimal_number % 2 + 10 * binaryConversion(decimal_number / 2, i + 1));
            }
        }

    }

}
```

Problem 2. Greedy Strategies

1. **Try arranging S in increasing order of weight.** Can you invent other values for S, v[], w[], W for which this strategy will not give a correct answer?
   For example: S={3, 1, 2}, v[]={1, 3, 4}, w[]={1, 2, 4}, W=5 does not give a correct answer. I don't think this approach is a greedy strategy to solve the problem.

2. **Try arranging S in decreasing order of value.** Can you invent other values for S, v[], w[], W for which this strategy will not give a correct answer?
   Again: S={2, 1, 3}, v[]={4, 3, 1}, w[]={1, 2, 4}, W=5 does not give a correct answer.

3. **Try arranging S in decreasing order of value per weight.** Can you invent other values for S, v[], w[], W for which this strategy will not give a correct answer?
   Again: S={1, 3, 2}, v[]={3, 1, 4}, w[]={ 2, 1, 4}, W=5 does not give a correct answer.

Problem 3.

1. It is possible that an optimal solution will not make use of item s_{n-1}
   S={1, 3, 2}, v[]={3, 2, 1}, w[]={ 4, 1, 2}, W=5

2. S_0' can be solution for smaller knapsack problem. Because S_0 was the solution for the former and weight restriction is reduced the same amount of w_{n-1}

-*-**