Assignment 10

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Suppose we want to make change for N cents, using the least number of coins of denominations {1, 5, 10} cents (Different from the US currency system!). Consider the following greedy strategy: suppose the amount left to change is M, take the largest coin that is no more than M; subtract this coin's value from M, and repeat.

Does this algorithm output an optimal solution? If not, give a counterexample. If yes, prove that this algorithm always outputs an optimal solution (a formal proof as what we have done in the video).

**2 points – Your answer**

This is an optimal solution

**5 points – Counterexample or proof**

M = 37

Greedy: a\*10 + b\*5 + c\*1 = n

a: 3

b: 1

c: 2

Optimal: a’\*10 + b’\*5 + c’\*1 = n

a’: 3

b’: 1

c’: 2

Since a = a’, b = b’, and c = c’ the greedy solution is the optimal solution. This work with any number

1. Consider the following problem.

Input: n segments of line with coordinates [x\_i, x\_j] on the x-axis.

Output: Find the minimum number of segments that cover the segment [0,M].

Note: segments can overlap.

For example:

Let M be 10, and we want to cover [0, 10].

There are 5 segments: [0, 2], [0, 3], [0, 9], [1, 6], [5, 12].

One way to cover [0, 10] is to use 3 segments: [0, 2], [1, 6], [5, 12].

The optimal solution is to use 2 segments to cover [0,10]:

[0, 9] and [5, 12]

Therefore, for this particular case, you should output 2.

Design a greedy algorithm to solve the problem.

1. describe the idea behind your algorithm in English (2 points);

We will assume that the items have a distance value for each that have been calculated. We then sort the items based off of their distance from smallest to largest. Then we enter a while loop which runs until the end is reached (M). there is an if statement that checks to see if the start value for the specific item is less than or equal to the tempStart. It also makes sure that the items second value is greater than the temp end. We can’t use greater than or equal to because there can be overlap then. The temp start and end values update and the i value increases by one. We check to see if M has been reach if so we are done otherwise we repeat.

(b) provide pseudocode (5 points);

Assume the items have dist already calculated

Int tempEnd = 0;

Int tempStart = 0;

Count = 0;

Int i = 0;

sort(items.dist);// sort the items based off distance

while (tempEnd<M && i < items.length){

if(items[i].x\_i <= tempStart && items[i].x\_j > tempEnd){ // allows overlap if it increases range

tempEnd = items[i].x\_j;

tempStart = items[i].x\_i;

count++;

i++;

}

}

Return count;

**Section 2: Java Implementation**

1. Implement the Coin Changing Problem in Java.

Note:

Find a file called Coinchange.java in assignment 10 folder.

Complete the method of greedycoinchange().

Test your method in the main method provided following the comments.

1. Implement the Fractional Knapsack Problem in Java.

Note:

Find a file called Fractionalknapsack.java in assignment 10 folder.

Complete the method of greedyfractionalknapsack().

Test your method in the main method provided following the comments

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 10 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_10'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**