Assignment 11

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

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

1. Given A[] = [ 12, 1, 3, 8, 2, 5], let x be 10. Run the subset sum algorithm on A[] to find out if there exists a subset of A[] with sum = x. Show the dynamic programming matrix Sum[i][j] that is needed to efficiently compute true or false for all possible i and j (5 points).

0 1 2 3 4 5 6 7 8 9 10

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

0| T F F F F F F F F F F

1| T F F F F F F F F F F

2| T T F F F F F F F F F

3| T T F T T F F F F F F

4| T T F T T F F F T T F

5| T T T T T T T F T T T

6| T T T T T T T T T T T

1. Consider the following problem.

Input: an array of positive integers A[].

Output: true if A can be partitioned into two subsets A1 and A2 such that the sum of the elements in each subset is equal, false otherwise.

For example, let A[] be {1, 1, 3, 4, 7}, then the output should be true. Because the given array can be partitioned into two subsets with equal sum: A1 = {1, 3, 4} and A2 = {1, 7}.

Outline a dynamic programming algorithm to solve this problem.

**(i) describe the idea behind your algorithm in English (2 points);**

The first thing to check for is to see if the sum of all the elements is even or odd. If it is odd then the split sides can’t be even so you can return false right away. Then it will initialize the table top row as base values , also initialize the first column. Then you can fill the table bottom up trying to get the values on each side to equal.

**(ii) provide the pseudocode (5 points).**

// Returns true if arr[] can be partitioned in two subsets of

static boolean findPartition (int arr[], int n)

{

int sum = 0;

int i, j;

for all elements

sum += arr[i];

if (sum%2 != 0)

return false;

boolean part[][]=new boolean[sum/2+1][n+1];

for the top row

part[0][i] = true;

for left column

part[i][0] = false;

// Fill the partition table in botton up manner

for (i = 1; i <= sum/2; i++)

{

for (j = 1; j <= n; j++)

{

part[i][j] = part[i][j-1];

if (i >= arr[j-1])

part[i][j] = part[i][j] ||

part[i - arr[j-1]][j-1];

}

}

return part[sum/2][n];

}

**(iii) analyze its running time (3 points).**

 O(n^2)

Full credit (10 points) will be awarded for a dynamic programming algorithm. Algorithms that are NOT dynamic programming will be scored out of 3 points.

1. Consider the following problem.

Given a set of coins with values (V1, V2, … Vm) and a target sum N, find the fewest coins required to equal N.

Outline a dynamic programming algorithm to solve this problem.

**(i) describe the idea behind your algorithm in English (2 points);**

This algorithm will look though all the possibilities of coins from 1 to V in order to find out what the smallest amount of coins. We have a base case in which the given V value is 0. Then we can set all of the table values as infinite. It will check to find the highest coin value that is less than the target value, allowing for the smallest amount of coins to be used.

**(ii) provide the pseudocode (5 points).**

static int minCoins(int coins[], int m, int V)

{

table[] = new int[V + 1];

table[0] = 0;

for all values in table

table[i] = Integer.MAX\_VALUE;

for (int i = 1; i <= V; i++)

{

for (int j = 0; j < m; j++)

if (coins[j] <= i)

{

int sub\_res = table[i - coins[j]];

if (sub\_res != Integer.MAX\_VALUE

&& sub\_res + 1 < table[i])

table[i] = sub\_res + 1;

}

}

return table[V];

}

**(iii) analyze its running time (3 points).**

O(mV).

**Note: This problem is discussed in the class.**

**Hint: This problem is very similar to the subset sum problem.**

Full credit (10 points) will be awarded for a dynamic programming algorithm. Algorithms that are NOT dynamic programming will be scored out of 3 points.

**Section 2: Java Implementation**

1. Implement the dynamic programming Subset Sum Algorithm in Java.

Note:

Find a file called SubsetSumDP.java in assignment 11 folder.

Complete the method of subsetSum().

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

**Full credit (30 points) will be awarded for a dynamic programming implementation of SubsetSum. Programs that are NOT dynamic programming will be scored out of 10 points.**

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 2 (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.**