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

2. 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);\**

This algorithm will run very similarly to the subset algorithm. The only difference is that we calculate our own search value. We do this by finding the total of the array and cutting it in half. If the total is not even then we may immediately return false. Else we will use the subset algorithm with the total of the array divided by 2 to acting as our search value. We know the total/2 will work because the array may potentially have 2 subsets necessary to split the array in half.

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

subSetEqual(A[])

int total = 0;

//finds the total value of everything in the list

for(int i; i < A.length; i++)

{

total = A[i] + total;

}//for

//if the total is odd than we know that there are not 2 array that will not be able to be equal to //one another

If(total % 2 != 0)

Return false;

Int search = total/2;

ArrayList[][] sums = new ArrayList[A.length][search];

For(int i =0; i< A.length; i++)

Sums[i][0] = true;

//empty subsets are equal to zero

For(int j = 0; j < search; j++)

Sums[0][j] = false;

//sum is not zero and the subset is empty

For (int i = 0; count < A.length; i++)

{

For(int j = 0; j < A.length; j++)

{

If(A[i-1] > j)

Sums[i][j] = sums[i- 1][j] || sums[i-1][j – A[i-1]]

Else

Sums[i][j] = sums[i- 1][j]

}//for

}//for

Return sums[n][search];

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

This algorithm will run in **BigO(n\*seach)**. This is because we originally loop through the array to get the total and within the 2D array we loop through the search value \* n. To be more precise the actual time is 2n\*search, but since Big O gets rid of constant values we can say it is **BigO(n\*seach)**.

**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**

3. 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.**

1. **Create a folder and name it 'FirstName\_LastName\_assignment\_11'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**