Jake Tantorski

Bowu Zhang

CMPT435

6 April 2020

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

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

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

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

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

5| T T T T T T T T T T

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

2. i) The idea behind this algorithm is that at first, you want to sum up all of the elements within the array. Next, you check to see if the sum/2 has a remainder using the modulo function. From there you want to make the entirely of the first row true. Next, you make the entirely of the first row false. Lastly, the table is filled with values in order to find out if there can be equal sums based on the input.

ii) Psuedocode

```
subSet( A[] , arrL) //where arrL = A.length
int sum = 0
int i, j
//find total sum of array
for i in [0, arrL]
    sum += A[i]
end for
//check to see if the total sum is divisable by two to see if you can have equal parts
if(sum%2 != 0)
    false
end if

new boolean arr[sum/2+1][arrL+1]
//Dynamic Programming For Loops
for i in [0,arrL]
    arr[0][i] = true
end for
for j in [ 1,sum/2]
    part[i][0] = false
end for
for i  in[1,sum/2]
    for j in [0,arrL]
        arr[i][j] = arr[i][j-1]
        if( i >= A[j-1])
            arr[i][j] = arr[i][j] || arr[i-A[j-1]][j-1]
        end if
    end for
end for
return arr[sum/2][arrL]
```

iii) The run time of this algorithm would be O(sum*arrL) due to the fact that both sum and arrL loop through a for loop in order to solve the algorithm shown by the highlighted areas.

3. i) The idea behind this algorithm is that you want to create a table that tells you the minimum coins needed for the input given. From there, you want to set all the values of that table to infinity. From there you will be able to use Dynamic Programming in order to figure out the coins required.

   ii) Pseudocode

```
coins(coin[], coinLength, n)

int storage []  = new int[n+1]

//check to see if the value is 0, therefore, it would be 0 coins

stroage[0] = 0

//use infinity as a placeholder for storage table

for i in [1,n]

        storage[i] = infinity

end for

for i in [1,n]

        for j in [0,coinLength]

                if(coin[j] <= i)

                        int x = storage[i - coin[j]];

                         if (x != infinity  && x + 1 < storage[i])

                                storage[i] = x + 1;

                        end if

                end if

        end for

end for

return storage[n]
```

   iii) The run of the code would be O(coinLength*n) due to the fact that you have to go through the entire length of the coins array and also the for loop cotaining the value in which is desired shown by the highlighted lines.