CSE 321 Homework #5

1-)

Firstly, For this part I can find number of sets that has equal sum zero but couldn't print them, just print how many sets there is that has zero sum. I use dynamic programming by using a 2D matrix. Initial length capacity for my matrix is 100x50. Then using by recursion and dynamic programming I fill this table.

```
26
27    arr = [2,3,-5,-8,6,-1];
28    print(Check_subset_zero(0, 0, arr, len(arr)));
```

For example, For given arr= [2,3,-5,-8,6,-1] my program output like this:

```
C:\Users\yasir\OneDrive\Masaüstü>python hw5_part1.py
4

C:\Users\yasir\OneDrive\Masaüstü>
```

It prints 5 because there is 5 possible subset of sum zero. They are

{2,3,-5}, {2,6,-8},{-5,6,-1},{3,6,-8,-1}

The complexity fort his algorithm is O(n*s) where s is sum of all elements and n is number of elements in given array.

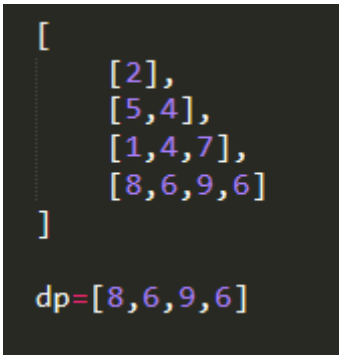
2-)

In this part, I find minumum path by starting from bottom to top approach. The main idea is that start from nodes on the bottom row, After that minimum path at the ith node of kth row would be the minimum of the pathsum of its two children + the node's itself value.

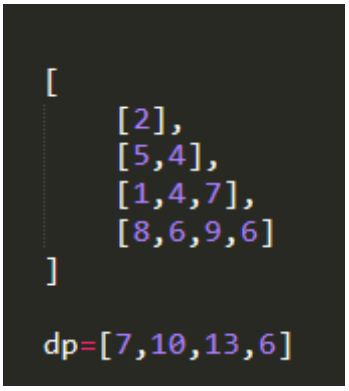
Main dynamic programming usage is that:

Dp[i] = A[k][i] + min(Dp[i],Dp[i+1])

Algorithm Works like that:



Firstly, this is our triangle and dp will be bottom row of triangle.

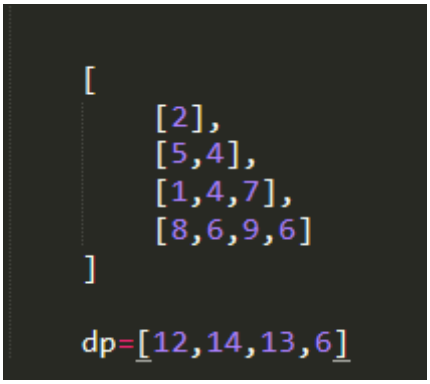


In second step our dp become like this.

I just use Dp[i] = A[k][i] + min(Dp[i],Dp[i+1]) formula

For example left most element in dp become 7 because dp[0] = 1 + min(8,6) = 7. And I find other dp elements like that then move from bottom to upper in my triangle.

Like this algorithm goes from bottem to top, in third step dp become like that:



Lastly it will become :

```
[
    [2],
    [5,4],
    [1,4,7],
    [8,6,9,6]
]

dp=[14,14,13,6]
```

Here just return first element in dp it will smallast possible path. So the answer will be 14 for this triangle.

```
C:\Users\yasir\OneDrive\Masaüstü>python hw5_part2.py
14

C:\Users\yasir\OneDrive\Masaüstü>
```

Time complexity fort his algorithm is O(n) where n is elements number in triangle. This is linear time because we access same element only one time in triangle.

3-)

This problem like same 0/1 knapsack problem that we have seen in class but in 0/1 knapsack problem we can select every items only once, in this problem we can select each item as much as we want.

For solving this problem dynamically I use 2 1D array instead of 2D array we use in 0/1 knapsack problem.

In algorithm, firstly I fill my dp with zeros. Then I use this dp formula: where i>= weights[j],

dp[i] = max(dp[i-weights[j]]+values[j],dp[i])

Here weights array contains my possible items weights and values array contains those items values.

Finaly the last element of my dp will be maximum value for given weights and I also hold an array selected Items.

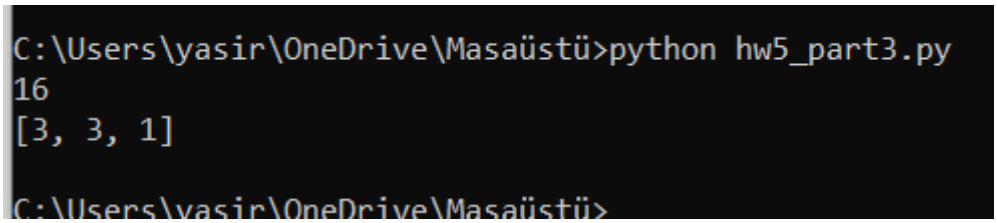
For

weights = [5,4,2]

values= [10,4,3]

maxW = 9

My output will be like this:



Here 16 is possible maximum choosable items total value. And [3,3,1] is selected items. It means for W=9 I can choose 2 times for item 3 and 1 times for item 1. Item 1 has 10 value and 5 weight, item 3 has 3 and 2 weight. So knapsack total weight will be 9 and total value will be 16 for those selected items.

Muhammed Yasir Fidan

161044056