CSE 321 Homework 5
REPORT

MİRAY YILDIZ

161044023

1) **In this question, the task is to print all subsets of given array with sum equal to 0.**

Explanation of my algorithm :

In the FindSubsets function, first, I create dynamic programming table. In this table, range is random(you can give any number you want). Then, recursively, I found all subsets, sum of subsets equal to 0. I appends this subsets to another array. So, I found all options.

In SumZeroSubsets function, first I found given arrays length and call FindSubsets function. Than, FindSubsets function finds all subsets and I print all subsets.

2) **In this question, I found the smallest sum path from the triangle apex to its base through a sequence of adjacent numbers.**

Explanation of my algorithm:

SmallestSumPath function finds smallest path in the triangle. For storage, I create an array.

```
arr = [None] * len(myarr)
```

Then, I found triangle array's length.

I calculated remaining rows, and I saved the paths to arr. (Bottom-up)

```
for i in range(len(myarr) - 2, -1, -1):
    for j in range(len(myarr[i])):
        arr[j] = myarr[i][j] + min(arr[j], arr[j + 1]);
```

Finally, I returned arr.

Arr's 0. Element is my smallest path.

```
      2
    5   4
  1   4   7
8   6   9   6
```

Output of my code:

```
Smallest path is :
14
```

3) **I used the Unbounded Knapsack algorithm for this question. The reason why I do not use the normal Knapsack algorithm is because in this algorithm we are allowed to use unlimited number of instances of an item.**

In my algorithm, Knapsack function returns the maximum value with knapsack of W capacity.

Firstly, I create

```
dp = [0 for i in range(W + 1)]
```

It is store maximum value.

Than, I use recursion and fill db :

```
for i in range(W + 1):
    for j in range(n):
        if (w[j] <= i):
            dp[i] = max(dp[i], dp[i - w[j]] + val[j])
```

Finally, i return dp.

In the above, we can see other alternatives. But we found maximum.

F(0)=0,

F(1)=0,

F(2)=max{F(2-2)+3}=3,

F(3)=max{F(3-2)+3}=3,

F(4)=max{F(4-2)+3, F(4-4)+4}=6,

F(5)=max{F(5-2)+3, F(5-4)+4, F(5-5)+10}=10,

F(6)=max{F(6-2)+3, F(6-4)+4, F(6-5)+10}=10,

F(7)=max{F(7-2)+3, F(7-4)+4, F(7-5)+10}=13,

F(8)=max{F(8-2)+3, F(8-4)+4, F(8-5)+10}=13,

F(9)=max{F(9-2)+3, F(9-4)+4, F(9-5)+10}=16.