-CSE 321-

Homework 5

Deadline: 22.01.2021

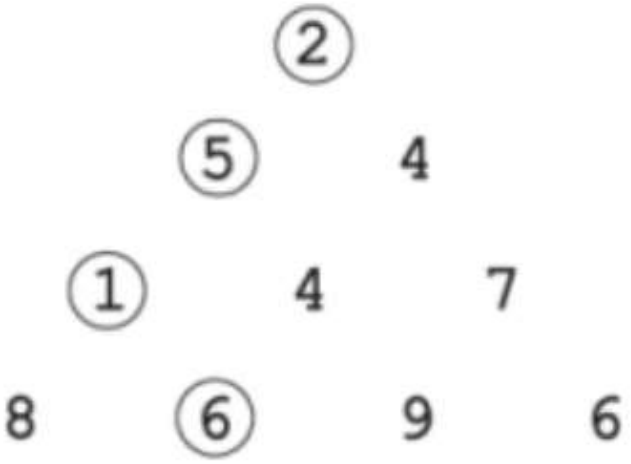
1) Suppose that you have an array A. A = {2, 3, -5, -8, 6, -1}. Propose a dynamic programming algorithm that checks whether there is a subset with total sum of elements equal to zero. Your algorithm should display the elements of each subset whenever it finds them. Implement your algorithm with Python and explain it in your report file.

- It is solved using the concept of dynamic programming.
- I create a boolean subset[][] and fill it in bottom up manner.
- The value of subset[i][j] will be true if there is a subset of set[] with sum equal to , otherwise false.
- Finally, we return subset[n][sum]

- Worst case time complexity: **Θ(n*sum)**
- Space complexity: **Θ(sum)**

p.s : But I couldn't print the subsets

2) Suppose that you have a set of integers arranged in the form of a triangle like below. Your aim is to find the smallest sum path from the triangle apex to its base through a sequence of adjacent numbers. The sequence in the following example is shown by the circles. Design a dynamic programming algorithm for this problem. Implement your algorithm with Python and explain it in your report file.



Explanation:

- Declare the triangle in 2D array form.
- Then call the function SmallestSumPath (A).
- Create a 1D array for memoization.
- Fill the array with first row and last row.
- Fill the remaining array finding the minimum of current and next element of memoization and adding the current cost in bottom up manner.
- Return the top element.

3) Suppose that you have a knapsack problem. You are given N items of weights $w1$, $w2$,… wn and their values are $v1$, $v2$, …, vn,respectively. The knapsack has capacity W. Your goal is to find the most valuable subset of the items that fit into the knapsack. You can pick from each item as many as you can. Design a dynamic programming algorithm and explain each step in your report file. Implement your algorithm with Python and explain it in your report file. Apply your algorithm with these values: $w1$ = 5, $w2$ = 4, $w3$ = 2; $v1$ = 10, $v2$ = 4, $v3$ = 3; W=9.

- It is solved using the concept of knapsack with dynamic programming.
- First of all declare the table to store the intermediate dynamic programming calculations.
- Building the table arr[][] in bottom-up manner.
- Then store the result of knapsack in the variable.
- Then check this variable if the item is included then print it, if this weight is already included, its values is reduced.

outputs:

```
Found a subset
Smallest sum path from the triangle :  14
Elements of the Knapsack :
4 5
```