CSE-321 Homework 5 Report

Created	@Jan 22, 2021
Introduction to Algorithms	HomeWork Lecture
Number	171044036
Property	Baran Hasan Bozduman

▼ **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.

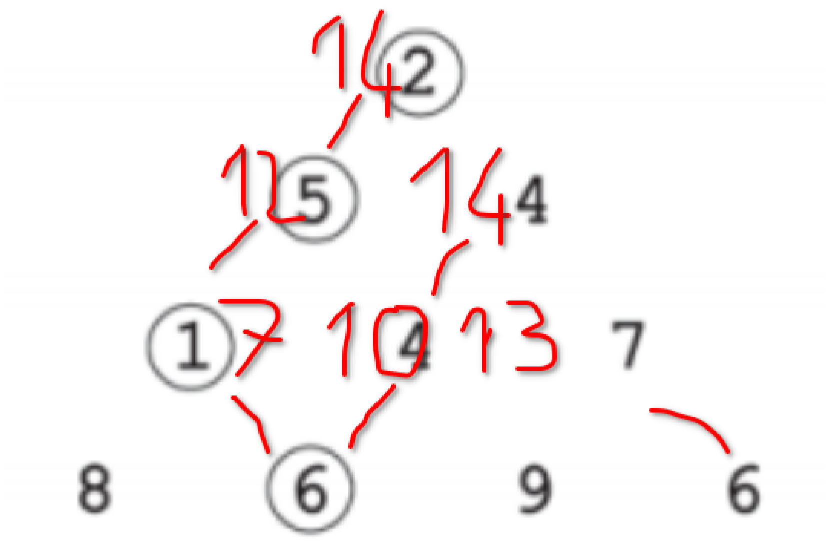
Since my algorithm did not for sum of zero i did not send it only works with positive numbers

▼ **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.

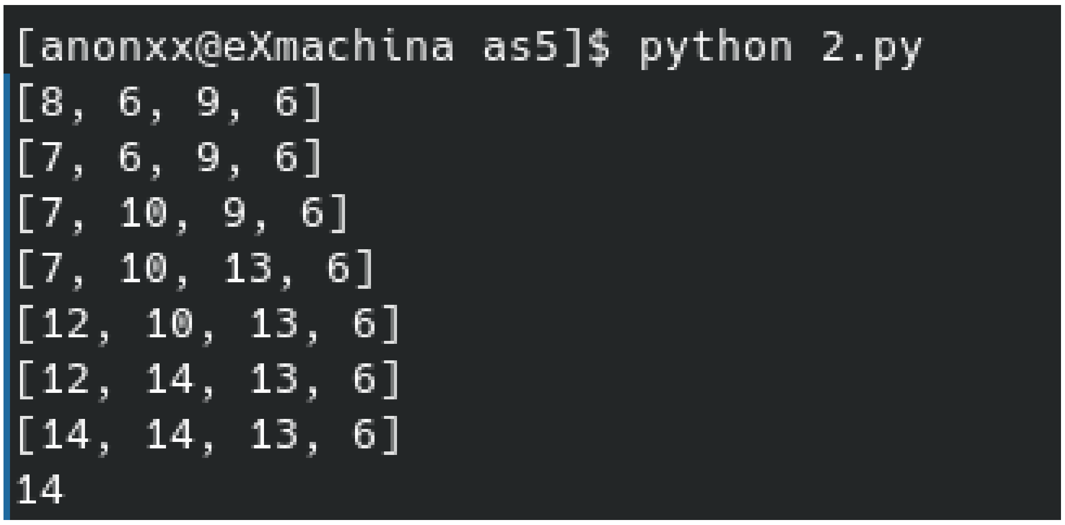
It is a common algorithm actually. I used bottom to top approach that means firstly i determine the smallest numbers in bottom then i calculated upper by decreasing possibilities so it gains us dynamic programming

1. I determine an array to calculate row by row
2. I assign bottom array to determined array
3. It adds the possible adjacent and determines the minimum summation.
4. It goes until top

5. When it reaches the top it return first index because it will be the only result.



As you can see in the table below it does not compute the adjacent which is not related with the number that algorithm chose

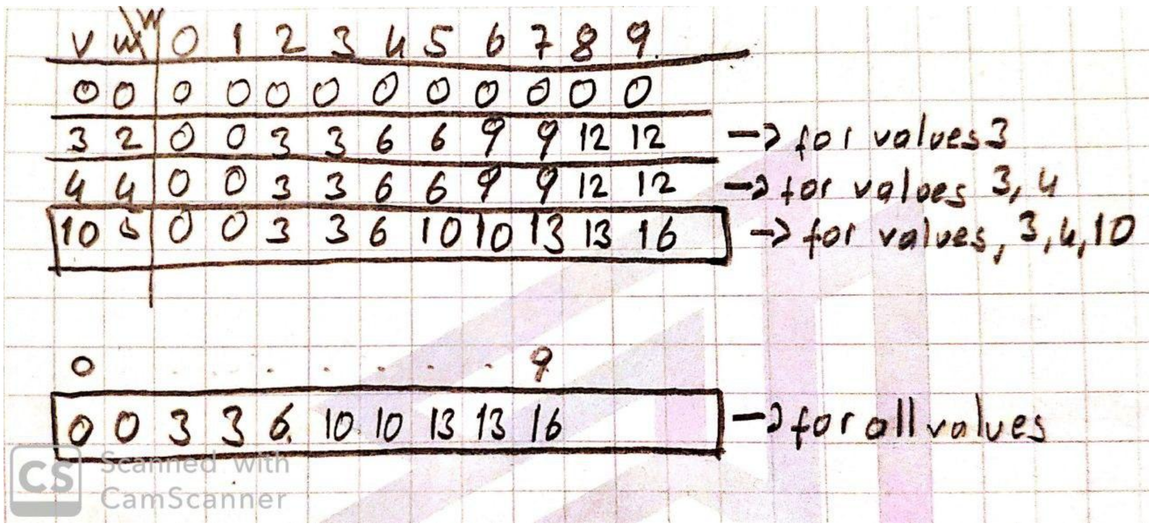


```
[anonxx@eXmachina as5]$ python 2.py
[8, 6, 9, 6]
[7, 6, 9, 6]
[7, 10, 9, 6]
[7, 10, 13, 6]
[12, 10, 13, 6]
[12, 14, 13, 6]
[14, 14, 13, 6]
14
```

▼ **3. Suppose that you have a knapsack problem. You are given N items of weights w 1, w 2,... w n 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; v 1 = 10, v 2 = 4, v 3 = 3; W=9.

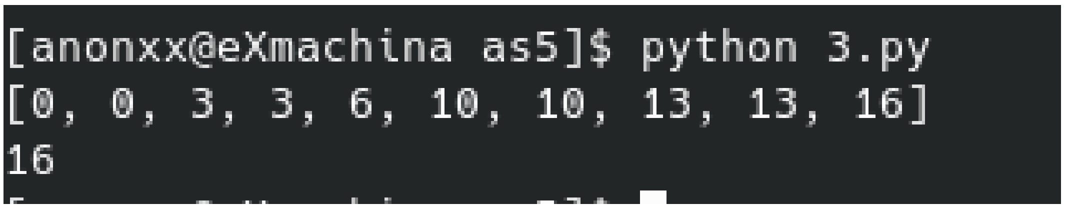
The problem "*You can pick from each item as*
many as you can" that asking for from us calls unbound knapsack problem normally we have to use one for each of them but since our algorithm should be unbound we can choose as many as we want.

lets show the table: Its easier to show in 2d table finally it reach bottom row



v	w \ w	0	1	2	3	4	5	6	7	8	9	
0	0	0	0	0	0	0	0	0	0	0	0	
3	2	0	0	3	3	6	6	9	9	12	12	$\rightarrow$ for values 3
4	4	0	0	3	3	6	6	9	9	12	12	$\rightarrow$ for values 3, 4
10	5	0	0	3	3	6	10	10	13	13	16	$\rightarrow$ for values, 3, 4, 10

0	.	.	.	.	.	.	.	.	9	
0	0	3	3	6	10	10	13	13	16	$\rightarrow$ for all values



```
[anonxx@eXmachina as5]$ python 3.py
[0, 0, 3, 3, 6, 10, 10, 13, 13, 16]
16
```

As you see it goes downward we put the values and weights into the table and first for 0 weight we put zeros to first row and first column then we put zeros into the 1 weight column because there is not any weight with 1. so I started 1 row which has valid values such as the row of 2 weight row

- ▼ For 2 weight
  1. we place the 3 which is the value of weight 2
  2. we place the 3 which is the value of weight 2 above because there is not enough weight to place weight 4
  3. we place the 3 which is the value of weight 2 because there is not enough weight to place weight5
- ▼ For 3 weight
  1. we place the 3 which is the value of weight 2
  2. we place the 3 which is the value of weight 2 above because there is not enough weight to place weight 4
  3. we place the 3 which is the value of weight 2 because there is not enough weight to place weight5
- ▼ For 4 Weight

    1. We place two weight2 which gives us 6
    2. Since the value of weight 4 is smaller we place theabove one which is 6
    3. Since the 5 weight bigeer than 4 we put above one which is biggest value until now
- ▼ For 5 Weight
    1. we put 6 value which is gives us to two 2 weight value
    2. we put 6 value which is two weights value is bigger than 4 weight value
    3. we put 10 which is weight 5 is bigger than aboves
- ▼ For 6 Weight
    1. we put 9 which is the value of 3 times weight 2
    2. We put 9 because its bigger than value of weight 4
    3. we put 10 which is the weight of 5 and its enough for replacement
- ▼ For 7 weight
    1. we put 9 which is the 3 times weight 2
    2. we put 9 which is bigger than weight of 4
    3. We put 13 which is one of weight 5 value plus one of weight 2 value
- ▼ For 8 Weight
    1. we put 12 which is the value of 4 times weight 2
    2. We put 12 because its bigger than two times weight four value
    3. we put 13 one of them is value of weight 5 plus weight 2 value
- ▼ For 9 weight
    1. 12 for two times 2 weight value
    2. 12 because two times weight 4 value is smaller
    3. we put 16 one of them is weight 5 value plus two times weight two value

So we reach max 16 for weight 9 and including all values.