Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

YUNUS ÇEVİK
141044080

Course Assistant: Mehmet Burak KOCA

Part – 1

Dynamic Programming is mainly an optimization over plain recursion. Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we do not have to recomupute them when needed later. This situation makes easier some process. In part 1 question, travel planning algorithm is asked with penalty score. Travel start on the road at mile post 0 as a reference point. Along the way there are n hotels, at mile posts a1 < a2 < · · · < an, where each ai is measured from the starting point. The passenger can stop to hotels but passenger can choose which of the hotels it stop at. The passenger would ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels).

If passenger travels x miles during a day, passenger would have penalty score depending to square of (200-x). The passenger must stop at the final hotel. But, passenger must plan how travel must be for minimum penalty score.

A = [190, 220, 410, 580, 640, 770, 950, 1100, 1350] will use as distances of hotels to starting point. This dynamic programming algorithm calculates distance of between two hotels and decide to stop in suitable hotels.

In this situation, the passenger must stop in a1, a3, a4, a6, a7, a8 and a9 hotels.

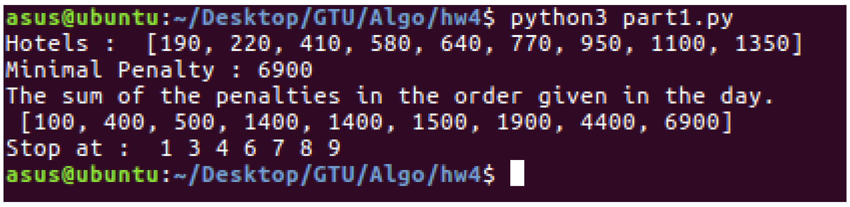
a1 - a0= 190 (square of (200-190))= 100
a2 - a1= 30
a3 - a2= 190 (square of (200-(30+190)))= 400
a4 - a3= 170 (square of (200-170))= 900
a5 - a4= 60
a6 - a5= 130 (square of (200-(130+60)))= 100
a7 - a6= 180 (square of (200-(180)))= 400
a8 - a7= 150 (square of (200-150))= 2500
a9 - a8= 250 (square of (200-250))= 2500

The minimum penalty score is 100+400+900+100+400+2500+2500= 6900.

Complexity

The length of the list where the hotels are located runs outside the external for loop. This portion lasts for T (n) = O (n). The internal for loop runs from 0, depending on the index value of the external for loop. In this cycle, the list of max hotels will be as long as the length of the list, this section takes as long as T (n) = O (n). In this section where the nested loops are present, the comlexity is
T (n) = O (n) * O (n) ==> T (n) = O (n ^ 2).
Time comlexity is T (n) = O (n ^ 2) Quadratic Time depending on the dynamic programing algorithm.

Result



```
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ python3 part1.py
Hotels :  [190, 220, 410, 580, 640, 770, 950, 1100, 1350]
Minimal Penalty : 6900
The sum of the penalties in the order given in the day.
 [100, 400, 500, 1400, 1400, 1500, 1900, 4400, 6900]
Stop at :  1 3 4 6 7 8 9
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ 
```

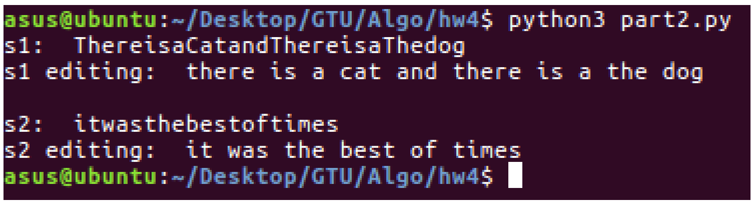
Part – 2

A string of n characters s[1 . . . n] is given which be a corrupted text document in which all punctuation has vanished (so that it looks something like “itwasthebestoftimes...”). The text document must be reconstructed the document using a dictionary, which is vailable in the form of a a Boolean function dict(·). From character to character navigates in text with two "for loop" and every time we need to check that exist or not the word in dict (). However, there may be some problems in the words in the text "the" and "there", "there" can accept as "the" in the dict. The method of avoiding from this problem is that the first "for loop" navigates to show the starting point of text and the second "for loop" navigates to show the end point of text. Even if "the" is found in the second for loop, the text advances, and if there is a word like "there", it is considered to be the last. In this way, adjacent text is separated from one another. The running time of this algorithm has been at most $O(n^2)$ assuming calls to dict take unit time.

Complexity

The outer for loop runs as long as the length of the string value. This portion lasts for T (n) = O (n). The inner for loop runs from 1 up to + 1 in length. Since the length of the string will be as long as it is in this cycle, it takes T (n) = O (n) in this section. In this section with nested loops, T (n) = O (n) * O (n) ==> T (n) = O (n ^ 2) Quadratic Time works as comlexity.

Result



```
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ python3 part2.py
s1:  ThereisaCatandThereisaThedog
s1 editing:  there is a cat and there is a the dog

s2:  itwasthebestoftimes
s2 editing:  it was the best of times
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ 
```

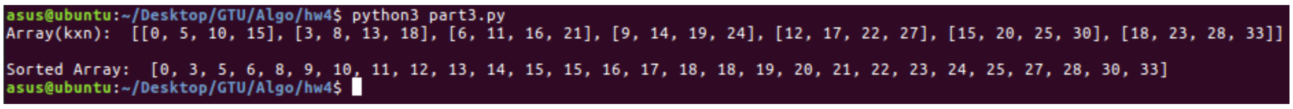
Part – 3

Suppose there are k sorted arrays, each with n elements, and these arrays must be combined them into a single sorted array that has k x n elements by divide-and-conquer algorithm.
First of all, we need to make the kxn array into a single array, then sorting must be with one of the sort algorithms that are appropriate to the divide and conquer algorithm. "Merge Sort" sorting algorithm was used that complexity is O (nlogn) in this algorithm.

Complexity

We get a non-sequential list by combining the ordered arrays in a single blank list. For this part T (n) = O (n) Linear Time works as complexity. We then sort the non-sequential sequence with a sort algorithm according to the divide and conquer algorithm. The sorting algorithm works as T (n) = O (nlogn). T (n) = O (n) + O (nlogn) ==> T (n) = O (nlogn) lasts.

Result



```
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ python3 part3.py
Array(kxn):  [[0, 5, 10, 15], [3, 8, 13, 18], [6, 11, 16, 21], [9, 14, 19, 24], [12, 17, 22, 27], [15, 20, 25, 30], [18, 23, 28, 33]]

Sorted Array:  [0, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 15, 16, 17, 18, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 30, 33]
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ 
```

Part – 4

In part 4 question, Alice wants to throw a party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know. The greedy algorithm must be proposed that takes as input the list of n people and the list of pairs who know each other, and outputs the best choice of party invitees. The people list is {'Arif', 'Arin', 'Arkan', 'Arkut', 'Arman', 'Baha', 'Bahadir', 'Bahattin', 'Bahri', 'Baki', 'Cabbar', 'Cafer', 'Cahit', 'Can', 'Canalp', 'Dalan', 'Dalay', 'Dalya', 'Damra', 'Darcan', 'Efehan', 'Efekan', 'Efgan', 'Efrahim', 'Efran'}.
List of people who know each other is below.

'Arif'	:{'Arin', 'Arkan', 'Arkut', 'Arman', 'Baha', 'Bahadir', 'Dalya', 'Efekan'},
'Arin'	:{'Arif', 'Arkut', 'Arman', 'Bahattin', 'Dalya', 'Efekan'},
'Arkan'	:{'Bahattin', 'Bahri','Cahit', 'Efehan','Arif', 'Arin'},
'Arkut'	:{'Cahit', 'Baha', 'Bahadir', 'Efehan', 'Efrahim', 'Efran', 'Arin', 'Bahattin', 'Dalya'},
'Arman'	:{'Baha', 'Efran', 'Damra', 'Arif'},
'Baha'	:{'Arif', 'Arkut', 'Arman','Baki', 'Efgan', 'Dalan', 'Bahri', 'Bahattin', 'Dalya'},
'Bahadir'	:{'Arif', 'Arkut', 'Efrahim ', 'Baki'},
'Bahattin'	:{'Arin', 'Arkan', 'Dalay', 'Dalya', 'Efehan', 'Bahri', 'Baha'},

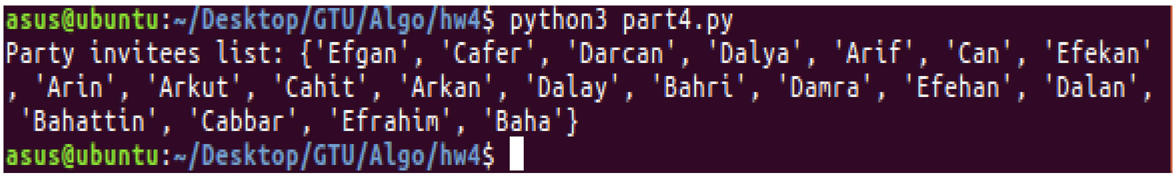
'Bahri'	:{'Arkan', 'Cabbar', 'Darcan', 'Cafer', 'Baha', 'Dalya', 'Efekan', 'Efehan'},
'Baki'	:{'Baha', 'Bahadir'},
'Cabbar'	:{'Bahri', 'Efehan', 'Cafer', 'Efekan', 'Cahit', 'Baha', 'Can'},
'Cafer'	:{'Bahri', 'Cabbar', 'Dalya', 'Dalay', 'Can', 'Dalan', 'Efehan', 'Baha'},
'Cahit'	:{'Arkan', 'Arkut', 'Cabbar', 'Dalya', 'Dalay', 'Dalan', 'Efehan', 'Baha'},
'Can'	:{'Efehan', 'Cafer', 'Efekan', 'Efran', 'Efrahim', 'Baha'},
'Dalan'	:{'Baha', 'Efgan', 'Darcan', 'Cahit', 'Cafer', 'Dalay', 'Bahri'},
'Dalay'	:{'Bahattin', 'Cafer', 'Cahit', 'Dalan', 'Damra', 'Efekan', 'Bahri', 'Can'},
'Dalya'	:{'Arin', 'Bahattin', 'Cafer', 'Cahit', 'Efgan', 'Efrahim', 'Bahri', 'Can'},
'Damra'	:{'Arman', 'Bahri', 'Darcan', 'Dalay', 'Efrahim', 'Efran', 'Can'},
'Darcan'	:{'Bahri', 'Damra', 'Efran', 'Dalan', 'Efrahim', 'Can'},
'Efehan'	:{'Arkan', 'Arkut', 'Bahattin', 'Cabbar', 'Can', 'Efgan','Darcan'},
'Efekan'	:{'Arin', 'Cabbar', 'Can', 'Dalay', 'Efran','Darcan'},
'Efgan'	:{'Baha', 'Bahadir', 'Dalan', 'Dalya', 'Efekan','Darcan', 'Can'},
'Efrahim'	:{'Arkut', 'Bahadir', 'Can', 'Dalya', 'Damra','Darcan'},
'Efran'	:{'Arkut', 'Arman', 'Baki', 'Efekan','Darcan'}

This algorithm looks at how many people know each other. Accordingly, if a person does not recognize at least 5 people, it is removed from the list of guests. This person is removed from the group of friends and the lists are updated. According to the current list, if a person does not recognize at least 5 people, that person is removed from the list. Finally, everyone on the list recognizes at least 5 people and does not recognize at least 5 people.

Complexity

In an endless loop, the base case runs up to the maximum T (n) = O (n), provided that the array is empty. There are 3 for loop in it, two of which are T (n) = O (n) + O (n), as well as the vertexes of the "friends" graph. The other is T (n) = O (n), which lasts as long as the number of elements that a directory contains. In short, while for loop, 3 for loop T (n) = O (n) + O (n) + O (n) is working. The function is T (n) = O (n) * (O (n) + O (n) + O (n)) ==> T (n) = O (n) * O (n) ==> T (n) = O (n ^ 2) It works as Quadratic time.

Result



```
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ python3 part4.py
Party invitees list: {'Efgan', 'Cafer', 'Darcan', 'Dalya', 'Arif', 'Can', 'Efekan'
, 'Arin', 'Arkut', 'Cahit', 'Arkan', 'Dalay', 'Bahri', 'Damra', 'Efehan', 'Dalan',
 'Bahattin', 'Cabbar', 'Efrahim', 'Baha'}
asus@ubuntu:~/Desktop/GTU/Algo/hw4$
```

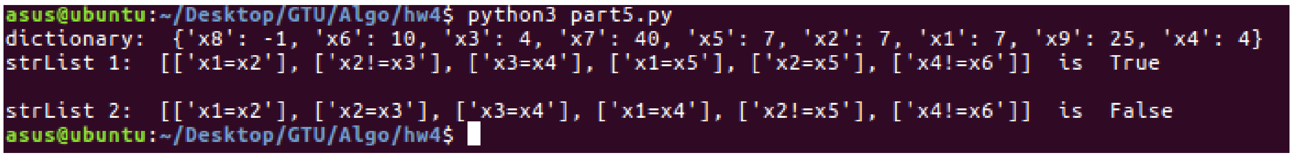
Part – 5

There are an array of strings that are defined as equality constraints and inequality constraints, and there are a set of dictionary in which the variables are located. While the variables are "key" in this dictionary, the values are in the "value" section. Strings in the array with conditions are navigated with a "for loop" and whether conditions are successful or unsuccessful according to the values in the dictionary. In order to get the most optimal result in the Greedy algorithm, if one of the conditions does not even suitable, "False" is returned and the function is exited. If all conditions are suitable, then "True" is returned.

Complexity

For the number of elements of a array of string arrays, the for loop runs. T (n) = O (n) Linear Time works because this loop is up to the maximum number of n elements. Comparisons with the dictionary are made for the loop. These are T (n) = O (1) Constant Time. The important thing here is the for loop.
T (n) = O (n) Works up to the Linear Time comlexity.

Result



```
asus@ubuntu:~/Desktop/GTU/Algo/hw4$ python3 part5.py
dictionary:  {'x8': -1, 'x6': 10, 'x3': 4, 'x7': 40, 'x5': 7, 'x2': 7, 'x1': 7, 'x9': 25, 'x4': 4}
strList 1:  [['x1=x2'], ['x2!=x3'], ['x3=x4'], ['x1=x5'], ['x2=x5'], ['x4!=x6']]  is  True

strList 2:  [['x1=x2'], ['x2=x3'], ['x3=x4'], ['x1=x4'], ['x2!=x5'], ['x4!=x6']]  is  False
asus@ubuntu:~/Desktop/GTU/Algo/hw4$
```