CSE 321 HOMEWORK 4

SEVGİ BAYANSALDUZ – 151044076

- **PART1**

```
untitled2.py | part1.py | part2.py | part3.py | part4.py | part5.py

# -*- coding: utf-8 -*-
"""
Created on Fri Dec 28 21:29:42 2018

@author: sevgi
"""
def calculateOptimalRoute(hotelList):
    path = [-1]*(len(hotelList))
    penalty = [0]*(len(hotelList))

    for i in range (len(hotelList)):
        penalty[i] = (200 - hotelList[i])**2#Calculate the penalty of each stop (200 – ai)^2
        path[i] = -1
        for j in range(i):
            temp = penalty[j] +((200 - (hotelList[i] - hotelList[j]))**2);#Calculate total penalty
            if (temp < penalty[i]):#choose the minimum penalty
                penalty[i] = temp;
                path[i] = j#save the path into the array path
    output=[]
    for i in range (len(path)):#Take the hotels distance according to elements of path and add distan
        if (path[i]>=0 and not(hotelList[path[i]] in output)):
            output.append(hotelList[path[i]])
    output.append(hotelList[len(hotelList)-1])
    return output;

print("Optimum path:"+str(calculateOptimalRoute([20, 40, 60, 940, 1500])))
print("Optimum path:"+str(calculateOptimalRoute([190, 420, 550, 660, 670])))
print("Optimum path:"+str(calculateOptimalRoute( [190, 220, 410, 580, 640, 770, 950, 1100, 1350] )))
```

This method takes an array for distances of hotel and calculates the minimum penalty.(penalty= (200 – x) ^2)

First two array will be created for penalties and path.The array path will be inialize with -1.
For i to n, penalty of each stop will be calculated.Then for j to i,minimum penalty will be chosen and index of hotels will be saved as a path into the array path.

After path is calculated,the method will create the output with the second for loop (which goes i to len(path)).In this loop ,the mathed will used the array path elements as index of hotelList,then will create the output.

Time Complexity: **(n=len(hotelList))**

$$\left(\sum_{i=0}^{n}\sum_{j=0}^{i} c\right)+\sum_{l=0}^{n}1 = \theta(n^2)$$

Output:

```
In [5]: runfile('C:/Users/sevgi/Desktop/hw4/part1.py', wdir='C:/Users/
sevgi/Desktop/hw4')
Optimum path:[60, 940, 1500]
Optimum path:[190, 420, 670]
Optimum path:[190, 410, 580, 770, 950, 1100, 1350]
```

- **PART2**

```
@author: sevgi
"""

dictionary={"it","was","the","best","of","times","you","can",
            "add","new","words","here","are"}

def isValid(string):
    #if length is 0 return false
    if(len(string)==0):
        return False;
    s2="";
    for i in range(len(string)+1):
          #s2 is a word ,delete s2 and add the next letter into it
        if(s2 in dictionary):
            if(i!=len(string)):
                s2=string[i].lower()
            else:
                s2=""#if the func reach the end of string ,add nothing
        else:#s2 is not  a word, concatanete s2  with the next letter
            if(i!=len(string)):
                s2=s2+string[i].lower()
    if(s2==""):
        return True
    else:
        return False

#given string
print(isValid("itwasthebestoftimes"))
print(isValid("youarethebest"))
print(isValid("besttimesks"))
```

This code takes a string.It traverses the string.Adds each letter,which is traversed, to the s2 array. The s2 array will be deleted and the next letter will be added to the s2 , if the s2 is a word.The function keeps to traverse string until there is no letter in the string.After traversing is done ,if the length of s2 equals to zero ,the function returns True.Otherwise it return False.

COMPLEXİTY: **n=len(string)**

$$\sum_{i=0}^{n}1 = \theta(n)$$

Output:

```
In [4]: runfile('C:/Users/sevgi/De
True
True
False
```

- **PART3**

```python
#This algorithm will divide the list into two parts ,until the best case occur.
#Then will merge the list element two by two.
#For instance ;we have 4 sorted arrays with this form : list[[],[],[],[]].This method will take li
#list[1] and will merge them ; it will merge list[3] and list[4] in the meantime.Then it will merg
def mergesort(list,size):
    #best case:if list has only  one element return list.
    if len(list) < 2:
        return list
    middle = len(list)//2
    #divide the list into two parts
    left = mergesort(list[:middle],size)
    right = mergesort(list[middle:],size)
    if(not(middle==size//2)):# make list the result of merge if method does not reach the end.
        return [merge(left[0], right[0])]#we send the first element of the left and right into the
                                         #of right and left are [[]] so we need to remove first bra
    return  merge(left[0], right[0])#if method reachs the end return the result of merge.

#the function merge is not change.
def merge(left, right):
    result = []
    i, j = 0, 0
    while ((i<len(left) ) and (j < len(right))):
        if left[i] < right[j]:
            result.append(left[i])
            i+= 1
        else:
            result.append(right[j])
            j+= 1
    while i < len(left):
        result.append(left[i])
        i=i+1
    while j < len(right):
        result.append(right[j])
        j+= 1
    return result


#standard form for input list.
list =[[2,7,9,11,15],[1,3,5,8,10],[1,3,5,6,10],[1,3,5,7,10],[31,44,55,60,85],[11,18,19,20,20]]
print("Given array is")
print(list);
print("Sorted array is")
print(mergesort(list,len(list)))
print("\n")
list =[[2,7,9,11,15],[1,3,5,8,10],[1,3,5,6,10],[1,3,5,7,10],[31,44,55,60,85]]
print("Given array is")
print(list);
```

I used merge sort algorithm for this problem This algorithm will divide the list into two parts, until the best case occur. Then will merge the list element two by two. For instance ;we have 4 sorted arrays with this form : list[[],[],[],[]]. This method will take list[0] andList[1] and will merge them ; it will merge list[3] and list[4] in the meantime. Then it will merge the result of previous merges.

COMPLEXİTY:

The recurrence relation of this algorithm is:

$T(n) = 2T(k/2) + \theta(n) = \theta(n.\log k)$

Output:

```
In [3]: runfile('C:/Users/sevgi/Desktop/hw4/
part3.py', wdir='C:/Users/sevgi/Desktop/hw4')
Given array is
[[2, 7, 9, 11, 15], [1, 3, 5, 8, 10], [1, 3, 5, 6,
10], [1, 3, 5, 7, 10], [31, 44, 55, 60, 85], [11, 18,
19, 20, 20]]
Sorted array is
[1, 1, 1, 2, 3, 3, 3, 5, 5, 5, 6, 7, 7, 8, 9, 10, 10,
10, 11, 11, 15, 18, 19, 20, 20, 31, 44, 55, 60, 85]


Given array is
[[2, 7, 9, 11, 15], [1, 3, 5, 8, 10], [1, 3, 5, 6,
10], [1, 3, 5, 7, 10], [31, 44, 55, 60, 85]]
Sorted array is
[1, 1, 1, 2, 3, 3, 3, 5, 5, 5, 6, 7, 7, 8, 9, 10, 10,
10, 11, 15, 31, 44, 55, 60, 85]
```

- **PART4**

```python
@author: sevgi
"""
# takes as input the list of n people and the list of pairs who know each other, and outputs the best cho
def partyInvitees(people,pairs):
    output=[]
    known=[]
    #take a person and calculate that how many people he/she knows.
    for i in range (len(people)):
        count=0#reset count
        for j in range(len(pairs)):#search this person in pairs
            if(people[i] in pairs[j]):#if this person is in  pair[j] ,reduce count by 1
                count=count+1
        known.append(count)#append 'count' into the array known.(this person is in the ith index and know
    for i in range(len(people)):#Take a person and make a decision to invite this person to the party or
        #if this person should have at least five other people whom they know
        #and five other people whom they don't know,add this person into the array output .
        if(known[i]>=5 and (len(people)+1 -known[i]) >=5):
            output.append(people[i])#this array includes people who will ne inviteed to the party.
    return output

#Standard form for people
people=["Jane","Bill","Susan","Tom","Jim","Mary","Will","Mick","Rory","Lindsay",
        "Kate","Karen","Joe","Martha","Arthur","Taylor","Connor"]

#standard form for pairs
pairs=[("Jane","Bill"),("Jane","Susan"),("Jane","Tom"),("Jane","Jim"),("Jane","Will"),("Jane","Connor"),
       ("Bill","Kate"),("Bill","Susan"),("Bill","Martha"),("Bill","Arthur"),("Bill","Karen"),
       ("Susan","Will"),("Susan","Tom"),("Susan","Jim"),("Susan","Taylor"),
       ("Tom","Will"),("Tom","Rory"),("Tom","Kate"),
       ("Jim","Joe"),
       ("Mary","Rory"),("Mary","Kate"),("Mary","Lindsay"),
       ("Will","Kate"),("Will","Rory"),
       ("Mick","Rory"),("Mick","Kate"),("Mick","Connor"),("Mick","Arthur"),("Mick","Martha"),
```

This algoritm takes as input the list of n people and the list of pairs who know each other, and returns the best choice of party invitees.

Then takes a person from the people array,for I to n, and search this person in pairs. if this person is in any pair , increase count by 1 for each pair which this person is in.After found the number of people who this person knows, add 'count' into the array known.

After calculating how many people they know for each person,algorithm takes array known and a person, and make a decision to invite this person to the party or not. if this person should have at least five other people whom they know and five other people whom they don't know,add this person into the array output .

COMPLEXİTY:

n=len(People)

$$\left(\sum_{i=0}^{n}\sum_{j=0}^{n} 1\right) = \theta(n^2)$$

OUTPUT:

```
In [2]: runfile('C:/Users/sevgi/Desktop/hw4/part4.py', wdir='C:/Users/
sevgi/Desktop/hw4')
['Jane', 'Bill', 'Susan', 'Tom', 'Will', 'Mick', 'Rory', 'Kate', 'Joe',
'Arthur']
```

- **PART5**

part1.py | part2.py | part3.py | part4.py | part5.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Dec 28 11:40:01 2018

@author: sevgi
"""
#takes as input constraints B over variable A and decides
#whether the constraints can be satisfied
def isSatisfiable(A,B):
    #check each consraints in B
    for i in range(len(B)):
        #second index holds '!' or '=',thus make comparison
        #according to the content of the secon element of the B[i
        if(B[i][2]=='='):#check equality constraints
            if(A[int(B[i][1])]!=A[int(B[i][4])]):
                return False
        else:#check inequality constraints
            if(A[int(B[i][1])]==A[int(B[i][5])]):
                return False
    return True

input1=[2,5,2,3]#standard form for variables
input2=[("x0=x2"),("x1!=x3")]#standard form for constraints
print("variables :"+str(input1))
print("constraints :"+str(input2))
print("output:"+str(isSatisfiable(input1,input2)))
print("\n")

input2=[("x1=x2"),("x2!=x3")]
print("variables :"+str(input1))
print("constraints :"+str(input2))
print("output:"+str(isSatisfiable(input1,input2)))
```

This method takes as input constraints B over variable A and decides whether the constraints can be satisfied.

Constaints are kept in an array in this form:

B= [("x0=x2"),("x1!=x3")]

And variables`s form is:

A= [2,5,2,3]

Based on these forms,while browsing the A list, we checker whether the rules in B are applied.

COMPLEXİTY:

n=len(B)

$$\left(\sum_{i=0}^{n}\right) = \theta(n)$$