CSE 321 Homework 4

141044084 – Seyit Ahmet KARACA

1)Algorithm Analysis

İf distance between all hotels is more than 200m occurs worst case.And worst case is O(n^2). İf there is only one hotel , occurs best case and is O(1).

Algorithm explanation

First of all , deckared 2 array with n size. First one holds penalties , second one holds which hotel you must to stop. Penalty calculation is initialize ith penalty (200 – hotel i)^2.

j is zero to i and if j th penalty + (200- j penalty – i th penalty)^2 is smaller then i th penalty ,then assign j th penalty + (200- j penalty – i th penalty)^2 to i th penalty and assign j + 1 to i th stop array. Last element of penalty array is desciribes minimum penalty. Calculation of result of method is array wich where to must stop is index = lenght stop array , final[index] = stopat[index] where index >= 0.

Cezaları ve hangi otelde durulması gerektiğini tutan iki tane array tutuldu. Cezaların tutulduğu arrayin i.ci elemanına (200 – hotel i)^2 atandı. 0 dan hotel sayısı kadar bir döngü oluşturuldu (değişkeni i) ve içerisinde 0'dan i'ye kadar giden bir döngü daha oluşturuldu. Bunun amacı ise dıştaki döngüde atanan i.ci cezayı önceki cezalarla karşılaştırıp optimum olanı bulmaktır. J.ci ceza+ (200- jth ceza– i.ci ceza)^2 < i.ceza dan küçük olma şartı doğru olursa i.cezaya J.ci ceza+ (200- jth ceza– i.ci ceza)^2 'nın sonucu atanır ve hangi durakta durulacağını belirten arraye j + 1 atanır. En sonunda tekrar eden elemanları yok etmek için is index = stopat arrayinin uzunluğu, final[index] = stopat[index] , index >= 0 ' iken formülü ile sonuç arrayi oluşturulur.

Method :

optimalSequence(_hotel) : küçükten büyüğe doğru sıralanmış tam sayılar içeren bir liste parametre alıyor.
En az ceza ile hangi hotellerde durması gerektiğini liste olarak veriyor.

inputHotel = [190, 220, 410, 580, 640, 770, 950, 1100, 1350]

optimalSopAtHotels = optimalSequence(inputHotel)

2) Algorithm Analysis

İf sentences size is 0 , best case occurs and it is O(1) else worst case is O(n^2).

Algorithm explanation

There is a word list and a sentence string. i is between 0 to lenght of sentences , algotihm is scanning last i letter and compare last i letter's first j letter. J is size of words.

When any matches puts 1 in the i th element in array. İn the begin , result array is initialized all elements with 0.

Method :

Reconstitute(sentence , liste) : bir string ile kelimelerin olduğu bir liste parametre olarak alıyor.
String uzunluğunda sıfır ve birlerden oluşan bir liste geri döndürüyor.Kelimeler cümlede var ise ilk harflerinin olduğu indexler 1 diğerleri 0 oluyor.

dic = ["the","best","of","there"]

cumle = "thebestoftherethe"

Reconstitute(cumle,dic)

3) Algorithm Analysis

Merge sort is O(nlogn) and union operate is O(k*n)

Algorithm explanation

Declared a kxn list array. İt's combined into one list and execute merge sort.

Method :

question3(ksa): Parametre olarak k*n boyutunda her biri kendi içerisinde sıralı liste alıyor.
Hepsini tek bir listede küçükten büyüğe sıralı olarak geri veriyor.

4) Algorithm Analysis

İf Alice matches person to first friend, best case occurs and it's O(n) , n is number of person invited to party. İf Alice matches persons to last friend , worst case occurs and it's O(n^2).

Algorithm explanation

Partiye katılacakların listesini bir kümede tuttum.Bu küme elemanlarını bir dictionary veri yapısında sıralarını yazarak kimleri tanıdığı bilgilerini liste olarak yazdım.

Greedy algoritma oluşturmam gerektiği için dictionary'deki elemanları tek tek alıp arkadaşlarına baktım.Bakılan insan ve tanıdığı arkadaşı davet edilmişler listesinde yok ise bu ikisini listeye ekliyorum. Herkesin 5 tanıdığı 5 tanımadığı insan olması gerektiğinden en az 11 kişi yazdım. 1 kişi açıkta kalıyor o da mecburiyetten açıkta kalıyor.

Method çiftlerin indexlerini ve isimlerini tutan iki tane liste geri döndürüyor.

Method:

aliceParty(personList,friendList) : isimlerin bulunduğu bir liste ile her bir kişinin tanıdığı 5 kişinin indexlerini tutan bir liste parametre olarak alıyor.

Paremetre olarak verilen listeler:

P = partiye devet edilecek potansiyel insanlar

P = ["seyit","aydın","safa","samet","yusuf","burki","furkan","cengo","fero","balli","sinan"]

PF = her bir kişinin tanıdığı insanların indexleri

PF = [[9,10,6,1,2],

[6,7,8,9,10],

[7,8,9,10,0],

[10,0,1,2,7],

[8,9,10,0,1],

[0,1,2,3,4],

[3,4,5,0,7],

[5,6,3,8,9],

[1,2,3,4,5],

[4,5,6,7,8],

[2,3,4,5,6]]

Çıktı olarak eşleşmerin indexleri ve isimleri veriyor.

Çıktısı:

[[9, 0], [6, 1], [7, 2], [10, 3], [8, 4]]

[['balli', 'seyit'], ['furkan', 'aydın'], ['cengo', 'safa'], ['sinan', 'samet'], ['fero', 'yusuf']]

5) Algorithm Analysis

İf there is one variable , this is best case and it's O(1). Otherwise O(n*m).

Algorithm explanation

I declared 3 list with variables and in a for loop found which variable is not satisfied.

Method:

question5(_equations) : eşitliklerin değişkenlerinin bulunduğu bir liste parametre olarak alıyor.

Çıktı olarak hangi değişkenlerin uygun olmadığının indexlerini veriyor.

Verilen liste :

equations=[[1,1,1,1,1],

[1,1,1,2,1],

[1,1,1,3,1]]

Çıktı: [[1, 3], [2, 3]]