

İslam Göktan Selçuk -- 141044071

PART-1

İç içe olan iki loop'un çarpımı $O(n^2)$ 'yi verir.

```
# Her durak için optimum penalty bulunur ve
# path ile birlikte kaydedilir.
for i in range(1, n):
    # 1. durak için penalty hesabı
    min = result[0] + (200 - (arr[i]-0))**2
    minIndex = 0
    path[i] = []
    # i. durak için optimum penalty bulunur.
    for j in range(1, i):
        temp = result[j] + (200 - (arr[i]-arr[j]))**2
        if temp < min:
            min = temp
            minIndex = j
    # i. durak için path ve penalty kaydedilir.
    temp = path[minIndex]
    path[i] = temp[:]
```

$O(n(n+1)/2) = O(N^2)$

PART-2

Dışarıdaki loop $O(n)$ zaman alır. İçerideki her i için i kadar döner. İkisinin çarpımı $O(n^2)$ 'yi verir.

```
for i in range(0, n):
    for j in range(0, i):
        if s[j+1:i+1] in dict:
            # Sentence'in j indeksine kadar olan kısmın valid olup
            # olmadığı ve j ile i arasındaki string'in dictionary'de olup
            # olmadığı kontrol edilir.
            if validIndexes[j] and dict[s[j+1:i+1]]:
                # Eger verilen aralıkta bir kelime varsa listede True
                # olarak işaretlenir.
                validIndexes[i] = True
```

$O(n) = O(n^2)$

PART-3

Merge işlemi $O(N)$ zamanda yapılır ve recursive çağrı bunu N defa yapar. Algoritmanın toplam çalışma süresi $O(n^2)$ olur.

```
# Verilen array listesini sıralı tek bir listeye donusturur.
def sortArrays(arrays):
    numberOfArrays = len(arrays)
    # Eger array sayısı birden fazlaysa isleme devam et.
    if numberOfArrays > 1:
        # 0. ve 1. array'leri birleştirir.
        arrays[0] = mergeArrays(arrays[0], arrays[1])
        # Array'leri birleştirdikten sonra 1. indeksteki array silinir.
        arrays.remove(arrays[1])
        # Array silindikten sonra tekrar fonksiyon çağırılır.
        sortArrays(arrays)
    return arrays[0]
```

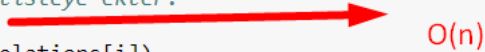
$O(n)$

PART-4

Algoritmanın toplam çalışma süresi $O(N)$ 'dir.

```
# Her kisinin kac tane tanidigi ve tanimadigi
# oldugunu kaydederek listeye ekler.
for i in range(0, n):
    friends[i] = len(relations[i])
    strangers[i] = n - (friends[i] + 1)
    # Kosulu saglayanlar incitedList'e kaydedilir.
    if friends[i] >= 5 and strangers[i] >= 5:
        invitedList.append(people[i])

print(friends)
```



PART-5

Toplam 3 tane $O(n)$ 'lik loop vardır ve bunların toplamı sonucu $O(3N)$ 'dir. Algoritmanın çalışma zamanı $O(N)$ olur.

```
arr[i] = i

indexes = {}
j = 0
# Tum degiskenler'e dictionary kullanilarak bir indeks atanir.
for x in variables:
    indexes[x] = j
    j += 1

# Esit olan degiskenler array uzerinden isaretlenir.
for x in constraints:
    tokens = x.split('=')
    if len(tokens) == 2:
        createUnion(arr, n, indexes[tokens[0]], indexes[tokens[1]])
# Array'de esit olarak isaretlenen aynı degere sahip olan degiskenler icin
# esit olmadiklarini iceren bir constraints bulunursa fonkstion false return eder.
for x in constraints:
    tokens = x.split('#')
    if len(tokens) == 2:
        if arr[indexes[tokens[0]]] == arr[indexes[tokens[1]]]:
            print("Constraints is not satisfied!")
            return False

print("Constraints is satisfied.")
return True
```