Lütfullah Türker
141044050

CSE – 321 HW 5

Q1)

Bu soru için önce elime kağıt kalem alıp, nasıl bir yol nasıl bir formül izlersem doğru yaklaşımı elde ederim diye düşündüm ve bulduğum çözümleri birkaç farklı Job listesi için denedim. En doğru sonucu W_i / T_i değerlerine bakarak buldum.

Algoritmamı W_i / T_i değeri en yüksek olandan en düşük olana doğru sıralama yaparak kurdum. Çünkü az zamanda yapılan yüksek weight li bir iş optimum sonucu elde etmemize yarar. Mesela (1,10) olan bir iş sona bırakılsaydı tüm t lerin toplamı kadar bir sayıyla ve üstelik çok yüksek sayılı bir weight ile çarpılacaktı. Bu da optimum sonucu vermezdi. Neden T_i / W_i ye göre sıralamadığımı da açıklayacak olursam, T_i değeri her job ilerledikçe bir önceki job'daki değeri ile toplanacak. Fakat W_i değeri için böyle bir durum yok.

Complexity için ;

Algoritmamda Job array ini W_i / T_i değerlerini tutan bir temp array e atadım. (O(n))

Bu temp arrayini gezerken her döngüde maximum değer bularak sort yaptım. Yani maximumu bulma (O(n)). Tüm array i gezerek maximumları bulup büyükten küçüğe sıralama ($O(n^2)$). Dolayısıyla complexity **$O(n + n^2) = O(n^2)$** dir.

Lütfullah Türker
141044050

Q2)

A)

```
for i= 1 to n
        if Ni < Si then
                Output "NY in Month i"
        else
                Output "SF in Month i"
end
```

Ödev PDF'indeki örnekteki gibi n=4 ve M=10 kabul edelim.

Örnek tablo aşağıdadır.

	Ocak	Şubat	Mart	Nisan
NY	12	18	10	12
SF	18	12	15	9

Yukarıdaki Algoritma'nın vereceği sonuç :

[NY,SF,NY,SF]

12+12+10+9+(3 * 10) = 73 → 3*10 işlemi 3 kere şehir değişikliği yapıldığı için M değerinin 3 ile çarpımı kadar değişiklik cost'udur ve toplam cost'a eklenir.

Olması gereken sonuç:

[NY,NY,NY,NY]

12 + 18 + 10 + 12 = 52 → Her ay NY şehrinde kaldığı durumdur. Dolayısıyla herhangi bir şehir değiştirme cost'u olmaz.Yani M değeri eklenmez. Optimum sonuçtur.

Sonuç olarak verilen algoritma hatalıdır. Optimum sonucu vermez.

Lütfullah Türker
141044050

B)

Soruda S ve N arrayleri 1. Indexden başladığı için kodda da öyle olması için arraylerin ilk elemanlarını 0 yaptım.

Hangi ilden başladığımıza göre optimum cost değişeceği için NY den başlandığı durum için bir optimum cost ve SF'den başlandığı durum için bir optimum cost ayrı olarak hesaplanır. İki hesaplama da bitince bu optimum değerlerden minimum olanı bize olabilecek minimum cost u verir.

Ters mantıkta düşünürsek, n. ayda NY'de isek optimum cost için 2 seçenek vardır.

Bulunulan ayın costu (N[n]) +

- n-1 ay için NY'de biten optimum cost. Veya
- n-1 ay için SF'de biten optimum cost + M (n. ayda NY de olduğumuz için M eklenmeli)

Bu işlemleri ilk aydan n. aya kadar NY ve SF için hesaplarsak n. ayda NY de biten ve SF de biten 2 optimum cost elde ederiz. Bunlardan küçük olanı bize optimum sonucu verir.

Algoritmamda sadece bir for döngüsü içinde 2 şehir için de optimum değerler hesaplanır. Ve minimum olanı optimum değer olarak return edilir. Yani complexity'si **O(n)**'dir.