Gözde DOĞAN
131044019

CSE 321 – Introduction to Algorithm Design
Homework 3

1. Devrim ile başa gelen Devlet Başkanı Cemal Gürsel yerli bir otomobil yapılmasını ister. Türk Milletinin bunu gerçekleştirebileceğini, gerçekleştirebilecek güce sahip olduğunu düşünerek yaptığı bu istek sonucunda medyadan, bürokrasiden yerli otomobil üretmenin mümkün olmadığını, her şeyin eksik olduğunu, pahalı olduğunu söyleyen bir sürü karşı söylem almasına rağmen vazgeçmez. Dış ülkenin karalamaları, bunun sadece hayal olduğunu söyleyen bir sürü kişiye, siyasi adamına rağmen bunu yapmayı kabul edecek insanlara başvurulur. Sanayi bakanının yaptığı toplantıda Türkiye Cumhuriyeri Devlet Demir Yolları'na bu isteğini bildirir ve odadaki diğer insanların bunu yapamayacağını söylemesine rağmen isteğinde diretir. Türkiye Cumhuriyeti Devlet Demir Yolları bunu kabul eder. Bunun üzerine yirmi üç mühendis Türkiye Cumhuriyeti Devlet Demir Yolları Eskişehir'deki Cer Atölyesine toplanır ve bu fikir açıklanır. Yetiştirileceği tarih 29 Ekim olarak belirlenen bu araba için 130 günün çok az olduğu düşünülse de işlemlere başlanır. Her mühendis ailesinden, yaşamından vazgeçer bu işe kendini verir. İçlerinden birinin getirdiği araba sökülerek incelenir ve bundan faydalanılır. Canla başla çalışan mühendisler tasarımı halleder ve bu arabanın adının Devrim olacağı belirlenir. Bu araba “Türk insanının makus talihine bir meydan okuma” olarak görülyordur. Araba yapım konusunda Recep Usta denilen bir döküm ustasından da yardım alınır. Araba parçalarını Recep Usta yardımı ile döküp hallederler. Medya bu süreç boyunca Türk Milletinin bunu yapamayacağı üzerine bir sürü demeç yayınlayıp durur hatta sokaktaki insanların bile bu şekilde görmesi, paralarının sokağa atıldığını düşünmeleri mühendislerin şevkini kırmaya yönelik bu davranışlar göz ardı edilir, çalışmalar devam eder. Bir gün Sami Bey Gündüz Beyi arayarak bütçe kesintisine gidilmek zorunda olunduğunu, çok para harcandığını söyleyerek bütçeyi üçte birine indirir. Bütçe kesintisini yanında mühendisler bir de elektrik kesintisi ile uğraşırlar. Parçalar gerekli olan aletler olmadan gerçekleştirilmeye devam edilir. Geçmişteki uçak fabrikasının dönemin başkanı tarafından kapatılması ve hatta sebebinin “Biz kim uçak yapmak kim” gibi bir yaklaşım ile kapatılması bazı mühendisleri umutsuzluğa sürüklese de mühendisler çalışmalara devam eder. Devrim sonucunda kararlaştırılan idam cezaları ile ilgili konuşmadan çalışmaya devam eden mühendisler bu yetmezmiş gibi bir de geç haber verilen 29 Ekime 2 araba yetiştirilmesi haberi ile karşı karşıya kalırlar. İlk şaşkınlık ardından bunun içinde çalışırlar. Arabalar 29 Ekime yetiştirilir ve halka sunulur. Cemal Gürsel arabaya bindiğinde benzin bitmesi ile araba durur ve herkes şaşkınlık içinde kalır. Devrim arabası yolda kalmıştır. Yaşı en küçük olan mühendis Necip yıllar sonra bu arabayı ürettikleri yere döndüğünde ve ürettikleri devrim arabasının sergilendiği gördüğünde duygulanır ve gururlanır da bir devrim hikayesi böyle sonuçlanmış olsa da...

2. Permütasyon yöntemi kullanarak her olasılığı buldum ve ardından bu olasılıklardan minumum olanı seçtim. Asistan sayısı kadar elemanlı bir liste oluşturdum ve bu liste üzerinde permütasyon uyguladım. Asistan sayısı ders sayısından büyük olabilirdi, bunun içinde zamanı hesaplarken ders sayısından büyük elamanlar yerine bölüm işlerinin yapılma saati (6) ile topladım.
3 tane metot yazdım, bunlar;

- FindOptimalAssistantship metodunda calculateTime ve permutation metotlarını kullandım.

  ```
  for l in permutation(tempL):
      time = calculateTime(l, len(L), len(L[0]), L)
          if minTime >= time:
                  …
  ```

  Çalışma zamanı permutation metodunun çalışma zamanı * calculateTime metodunun çalışma zamanıdır.

- CalculateTime metodunda gelen listedeki eleman sayisi kadar toplama işlemi yapılıyor. Gelen listenin eleman sayısı da asistan sayısı(r) kadardır. CalculateTime metodu lineer zamanda çalışır. Complexity'si O(r)'dir.

- Permutation metodunda gelen listenin elamanlarının tek tek sıralanması işlemi farklıo şekillerde gerçekleştiriliyor. Yani gelen listenin ilk elemanını listeye yerleştirip ardından listenin geri kalanı için metodu çağırarak burdan gelen listeyi de bu listeye ekleyerek sıralama işlemini gerçekleştirmiş oluyoruz. Listenin eleman sayısı asistan sayısıdır, yani r'dir. Bu nedenle Worst Case de r! kadar çalışıyor. Ama best case complexity'si 1 veya 0 elaman olma durumundan kaynaklı lineer zaman'dır.
  Worst Case = O(r!), Best Case = O(1)
  Permutasyon algoritması için, https://stackoverflow.com/questions/13109274/python-recursion-permutations, http://www.geeksforgeeks.org/generate-all-the-permutation-of-a-list-in-python/ sitelerinden yararlanıldı. (Anladıktan sonra kendim yazmaya çalıştım.)

FindOptimalAssistantship metodunun çalışma zamanı ise, permutation metodu r! kadar çalışacağı için for döngüsü r! kadar dönecek ve içinde calculateTime metodu çalışıyor, calculateTime metodu da yukarıda açıklanmıştı, r kadar dönecek, burdan FindOptimalAssistantship metodunun çalışma zamanı r.r! olarak bulunur. Yani bu algoritmanın çalışma zamanı = O(r.r!) 'dir.

3. BreadthFirstSearch algoritmasını kullanarak her vertexten başlayıp pathleri hesapladım, burdaki pathlerin yol ve bilgisyar labaratuvarı kurmak için gereken fiyatı hesapladım, bu hesaplamalardan en minumum olanı buldum.

2 metot yazdım;

- BFS algortimasının karmaşıklığı zaten (|V|+|E|)'dir.
- FindMinimumCostToLabifyGTU algoritması da her vertex üzerinde BFS uygulayarak işlemleri gerçekleştirir. |V|*(|V|+|E|) = $|V|^2$ +|V|*|E| dir. Vertex sayısı her zaman edge sayısından büyük olacağı için algoritmanın karmaşıklığı $|V|^2$ olarak bulunur.

Breadth First Search algoritması Veri Yapıalrı dersinde öğrenilen algoritma kullanılarak gerçekleştirildi.

4.

Insertion Sort:

Pseudocode insertionSort (L[1:n])

```
    for i ← 2 to n do
        current ← L[i]
        position ← i-1
        while (position >= 1) and (current < L[position]) do
            position ← position-1
        end while
        L[position+1] ← current
    end for
end
```

<u>Insertion Sort uygulaması;</u>

List = 12, 34, 54, 2, 3

n=5

(Listenin ilk elemanının index'i 1, son elemanının index'i 5 olarak düşünülerek işlem yapıldı)

for döngüsünde;

i = 2, current = 34, position = 1

while döngüsünde 1>=1 sağlanır ama 34<12 sağlanamadığı için döngüye girilmez.

L[position+1] = L[2] = 34

List = 12, 34, 54, 2, 3

i = 3, current = 54, position = 2

while döngüsünde 2>=1 sağlanır ama 54<34 sağlanamadığı için döngüye girilmez.

L[position+1] = L[3] = 54

List = 12, 34, 54, 2, 3

i = 4, current =2, position = 3

while döngüsünde

3>=1 sağlanır ve 2<54 sağlanır, döngüye girilir.

L[position+1] = L[4] = 54

position = position - 1 = 2

2>=1 sağlanır ve 2<34 sağlanır, döngüye girilir.

L[position+1] = L[3] = 34

position = position - 1 = 1

1>=1 sağlanır ve 2<12 sağlanır, döngüye girilir.

L[position+1] = L[2] = 12

position = position - 1 = 0

0>=1 olmadığı için dmngüye girilmez.

L[position+1] = L[1] = 2

List = 2, 12, 34, 54, 3

i = 5, current = 3, position = 4

while döngüsünde

4>=1 ve 3<54 şartları sağlandığı için girilir.

L[position+1] = L[5] = 54

position = position - 1 = 3

3>=1 sağlanır ve 3<34 sağlanır, döngüye girilir.

L[position+1] = L[4] = 34

position = position - 1 = 2

2>=1 sağlanır ve 3<12 sağlanır, döngüye girilir.

L[position+1] = L[3] = 12

position = position - 1 = 1

1>=1 sağlanır ve 3<2 sağlanmadığı için döngüye girilmez.

L[position+1] = L[2] = 3

List = 2, 3, 12, 34, 54

i = 6 olduğunda for döngüdünden çıkılır. Ve liste sıralanmış olur.

- Insertion sort current elemandan başlayıp ondan önceki bütün elemanlar ile kıyaskayarak gerekli ise swap yaparak listenin başına kadar gider. Bunu her index için tekrarlar. Bu nedenle de for döngüsüne n(input size) kadar girilir. While döngüsü ise her seferinde ilk index'e kadar gitme ve current elemanın index'teki eleman ile karşılaştırma durumuna göre çalışacağı için en kötü durumda n, en iyi durumda 0 defa tekrarlanır. Burdan da Best case = n, Worst case = n^2 olur
- Insertion sort online bir algoritmadır.
- Insertion sort stability 'i sağlar.
- Insertion sort in-place bir algoritmadır.

Shell Sort ;

```
Pseudocode shellSort(L[0....n-1])
        gap ← n/2
        while gap > 0 do
                for  i ← gap to n do
                        temp ← L[i];
                        j ← i
                        while  (j >= gap) and (L[j - gap] > temp)
                                L[j] = L[j – gap];
                                j ← j-gap
                        end while
                        L[j] ← temp;
                end for
                gap ← gap/2
        end while
end
```

Shell Sort Uygulaması;

List = 12, 34, 54, 2, 3

n=5 (Listenin ilk elemanının index'i 0, son elemanının index'i 4 olarak düşünülerek işlem yapıldı)

gap = 5/2 = 2;

i = 2, temp = 54

j = 2;

2>=2 doğru ama 12>54 doğru olmadığı için while döngüsüne girilmez.

L[j] = L[2] = 54

List = 12, 34, 54, 2, 3

i = 3, temp = 2

j = 3

3>=2 ve 34>2 şartları sağlandığı için while döngüsüne girilir.

L[j] = L[j-gap] => L[3] = L[1] => L[3]=34

j = j - gap = 3-2 = 1

j=1

1>=2 şartı sağlanmadığı için döngüye girilmez.

L[j] = L[1] = 2

List = 12, 2, 54, 34, 3

i=4, temp = 3

j=4

4>=2 şartı sağaır ama 54>3 şartı sağlnmadığı için döngüye girilmez.

L[j] = L[4] = 3

List = 12, 2, 54, 34, 3

gap = gap / 2 = 2/2 = 1

i=1, temp = 2

j=1

1>=1 ve 12>2 şartları sağlanır, döngüye girilir.

L[j] = L[j-gap] => L[1] = L[0] => L[1] = 12

j = j-gap = 1-1 = 0

j=0

0>=1 şartı sağlanmadığı için döngüye girilmez.

L[j] = L[0] = 2

List = 2, 12, 54, 34, 3

i=2, temp = 54

j=2

2>=1 şartı sağlanır ama 12>54 şartı sağlanmadığı için döngüye girilmez.

L[2] = 54

List = 2, 12, 54, 34, 3

i=3, temp = 34

j=3

3>=1 ve 54>34 şartları sağlanır, döngüye girilir.

L[j] = L[j-gap] => L[3] = L[2] => L[3] = 54

j = j-gap = 3-1 = 2

j=2

2>=1 şartı sağlanır ama 12>34 şartı sağlanmadığı için döngüye girilmez.

L[j] = L[2] = 34

List = 2, 12, 34, 54, 3

i=4, temp = 3

j=4

4>=1 ve 54>3 şartları sağlanır, döngüye girilir

L[j] = L[j-gap] => L[4] = L[3] => L[4] = 54

j = j-gap = 4-1 = 3

j=3

3>=1 ve 34>3 şartları sağlanır, döngüye girilir

L[j] = L[j-gap] => L[3] = L[2] => L[3] = 34

j = j-gap = 3-1 = 2

j=2

2>=1 ve 12>3 şartları sağlanır, döngüye girilir

L[j] = L[j-gap] => L[2] = L[1] => L[2] = 12

j = j-gap = 2-1 = 1

j=1

1>=1 şartı sağlanır ama 2>3 şartı sağlanmadığı için döngüye girilmez.

L[1] = 3

List = 2, 3, 12, 34, 54

gap =1/2 < 0 olduğu için while döngüsünden de çıkılır.
List = 2, 3, 12, 34, 54 şeklinde sıralnmış olur.

- Shell sort temp elemandan başlayıp ondan önceki bütün elemanlar ile kıyaslayarak gerekli ise swap yaparak listenin başına kadar gider. Bunu her gap aralıktaki index için tekrarlar. Bu da for döngüsünü n/gap kadar her zaman dönmesini sağlar. İçerdeki while döngüsü ise 2 şarta bağlıdır. Bu nedenle de Worst case = n^2 olur. Best case ise içerdeki while döngüsüne bağlı olur, Best case = $n^{7/6}$ 'dır.
- Shell sort online bir algoritmadır.
- Shell sort stability 'i sağlar.
- Shell sort in-place bir algoritmadır.

→Yukarıdaki açıklamalardan da anlaşılacağı üzere insertion sort ile shell sort'un genel özellikleri aynıdır. İkisi de in-place, stability ve online algoritmalardır. Worst case'leri aynıdır. Best case'lerinde shell sort'un daha kötü olduğu görülmektedir. Ama average case'ler kıyaslandığında shell sortun daha iyi olduğu anlaşılır.