The Imitation Game

Film 3 bölümden oluşmaktadır. Alan Turing'in çocukluğu, krptolojiyle olan çalışmaları ve tutuklanması bu bölümleri oluşturmaktadır. Alan Turing'in gençliğini Alex Lawther adında bir kişi oynamaktadır. Alan yalnız bir öğrenciydi. Tek arkadaşı Christopher idi. O arkadaşı da gittikleri bir tatilden sonra geri dönmemişti. Bu haberi okudukları okulun müdürü tarafından almıştı ve o günden sonra hiçbirşey eskisi gibi olmamıştı.

Bu olaylardan sonra Turing büyümüştü. Artık Turing rolünü İngiliz sinemalarının başarılı oyuncusu Benedict Cumberbatch üstlendi. O zamanlar üniversitede matematikçi olan Alan bir şekilde orduya katıldı. Katılmasının sebebi İkinci Dünya savaşında Nazilerin kullandığı Enigma isimli makinanın kodunun kırılması idi. Enigma'nın kırılmasına o zamanlar imkansız olarak bakılmaktaydı. Çünkü her 24 saatte bir Enigma yeni bir şifre üretmekteydi. Bunun için ordu içerisinde ülkenin en iyi matematikçileri ve kriptografi üzerine ciddi çalışmaları olan içlerinde Turing'in de bulunduğu bir grup kuruldu. Turing gurubun içindeki en zeki en hırslı ve anlaşılması en zor kişi idi. Bu nedenle grubun içerisinden birkaçı eksilince yeni matematikçiler bulmak amacıyla kendince bir seçim yapan Turing Joan Clarke ile tanışır. Joan, üstün zekası ile Turing'in hayranlığını ve dostluğunu kazanır.

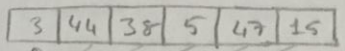
Turing Enigma'nın kodunu kırmak için bir makina tasarlamaya başlar. İsmini de çocukluk arkadaşı olan Christopher koyar. Bu makine ilk bilgisayar olarak da bilinir. Nihayet günlerce üzerinde çalıştığı makinanın yapım aşamasını bitiri. Sıra test etmeye gelir. Fakat makine çalışmaz. Kendisini işe alan kişiler tarafınca işten kovulmak üzereyken arkadaşları da eğer Turing giderse kendilerinin de gideceğini söyler ve zar zor 1 ay daha zaman kazanırlar. Turing yaptığı hatayı Joan ile birlikte bulur ve düzeltirler. Makine çalıştıktan sonra Enigma'nın şifresini kırarlar. Her gün yeni şifreler üretildiği için her gün kodu kırarlar. Fakat Almanların kodu kırdıklarını bilmelerini istemezler. Çünkü eğer öğrenirlerse Enigma'nın çalışma rensibini değişiteceklerini düşünürler. Bu bir devlet sırrı olarak uzun süreler saklanır.

CSE 321 - Homework 02

Seyfullah Beceriklı
121044028

1) Selection Sort

```
for j=1 to n-1
    smallest = j
    for i=j+1 to n
        if list[i] < list[smallest]
            smallest = i
    Swap list[j] ⇔ list[smallest]
```



There is no small element from 3

3 44 38 5 47 15

38<44 so go
5<44 so go
There is no small element from 5
so swap

3 5 38 44 47 15

15<38 swap

3 5 15 44 47 38

38<44 swap

3 5 15 38 47 44

44<47 swap

3 5 15 38 44 47

Insertion Sort

```
for i=1 to length(list)
    x = list[i]
    j = i
    while j > A[j-1] > x
        A[j] = A[j-1]
        j = j-1
    A[j] = x
```

3 44 38 5 47 15

3<44 38<44 but 3<38
swap

3 38 44 5 47 15

5<44
swap

3 38 5 44 47 15

5<38
swap

3 5 38 44 47 15

index

3 5 38 44 47 15

47!<44
15<44 swap

index

3 5 38 15 47 44

15<38 swap

index

3 5 15 38 47 44

5!<15 stop.
44<47 swap

3 5 15 38 44 47

38!<44 stop

Bubble Sort

```
x = length(list)
for i=1 to x
    swaped = false
    for j=1 to x
        if list[j] > list[j+1]
            swap(list[j], list[j+1])
            swaped = true
    if (! swaped)
        break
return list
```

3 44 38 5 47 15

38<44 change

3 38 44 5 47 15

5<44 chag

3 38 5 44 47 15

15<47 change

3 38 5 44 15 47

5<38

3 5 38 44 15 47

15<44

3 5 38 15 44 47

3 5 38 15 44 47

15<38

3 5 15 38 44 47

So sorted

Quick Sort

```
position = length(list) / 2
if high > low
    Rearrange(list[low:high], position)
    QuickSort(list[low:position-1])
    QuickSort(list[position+1:high])

Rearrange(list[low:high], position)
    right = low
    left = high+1
    x = list[low]
    while right<left
        repeat right = right+1 until list[right]>x
        repeat left = left-1 until list[left]<x
        if right<left
            swap(list[left], list[right])
    position = left
    list[low] = list[position]
    list[position] = x
```

2) a) Is selection sort stable?

Stabil değildir. Örneğin 4, 2, 3, 4, 1 dizisini sıralayalım. En küçük elemanı bulacağız. Burada 1 en küçük eleman. 1 en başa gelecek ve ilk sırada bulunan 4 te en sona gelecek. 1, 2, 3, 4, 4 şeklinde sıralanmış olacak. Fakat sıralanmamış dizide ilk sırada bulunan 4 sıralanmış dizide 3. index teki 4 ten sonra gelmiş oldu.

b) Is bubble sort stable?

Stabildir. Çünkü bubble sort aynı elemanların birbirine göre posisyonları değişmemektir. Temel işlem olan kıyaslama büyüktür veya küçüktür operatörü kullanır. Buda aynı elemanlar için yer değiştirme operasyonunu başlatmaz.

c) Is it possible to implement selection sort for linked list with the same value $\Theta(n^2)$ efficiency as the array?

Evet mümkündür. İki işlemde de en küçük elemanı bulup yer değiştirmek aynı verim ile gerçekleştirilebilir.

d) Evet mümkündür. Fakat sıralanan kısmı eleman ekleyerken taramamız gerekmektedir.

3) İlk önce ilk iki eleman kıyaslanır. Sağdaki beyaz ve soldaki siyah yer değiştir. Daha sonra ikili grup halinde ilk 4 disk ele alınır. İlk ikilideki sağda kalan siyah ile diğer ikilideki sağda kalan beyaz yer değiştirir. Ardından 3lü grup olacak şekilde ilk 6 disk ele alınır. İlk 3lüdeki sağda kalan siyah ile ikinci 3lüdeki sağda olan beyaz yer değiş. En son 4'lü grup şeklinde bütün diskler ele alınır. İlk dörtlüdeki sağda olan siyah ile ikinci 4lüdeki sağda olan beyaz yer değiştirir.

4) n tane 0'dan oluşan bir string olsun ve $\underbrace{0\ldots01}_{m-1}$ şeklinde bir modelimiz olsun. Bu model en berbat durumda olur. Brute-force string eşleştirme algoritması $m(n-m+1)$ karakter karşılaştırması yapacaktır.

5) a) Aranan kelimenin sayısı 0 olarak ilklendirilir. Soldan sağa doğru arama yapılır. Eğer "A" harfi ile karşılaşılırsa ondan sonra gelen "B" harflerini sayar ve 0 olan aranan kelime sayısını "B" harfi sayısı kadar arttırır. Bu işlem string bitene kadar devam eder. n tane "A" harfinden oluşan bir string için en kötü durumda karşılaştırma yapılan karakter sayısı

$n+(n-1)+\ldots+2 = n(n+1)/2 - 1$ dir.

Bu da $\Theta(n^2)$ ye tekabül eder.

b) Aranan kelimenin ve "A" harflerinin sayısını 0 olarak ilklendiririz. Yine soldan sağa doğru string'i taramaya başlarız. Eğer bir A harfi bulunursa A sayısını 1 arttırırız. Eğer B harfi bulunursa o an olan A sayısını aranan kelime sayısına ekleriz. String bittikten sonra aranan kelime sayısı return edilir.

6) İki çocuk bot ile karşıya geçer. Çocuklardan birisi karşı tarafta kalır. Diğeri bot ile askerlerin olduğu tarafa geçer. Çocuk bottan iner ve asker bot ile diğer tarafta olan çocuğun yanına geçer. Asker orada iner ve orda olan çocuk bot ile diğer askerlerin olduğu tarafa geçer. Her asker için olacak bu rutin 4 geçiş asker sayısı kadar tekrarlanır. 4n kez bot geçişi olur.

7) Gruptan bir kişi seçeriz. Grubun diğer elemanlarını tanıyor mu diye rastgele sorarız. En kötü durumda en son sorduğumuz kişiyi tanıyor ise aradığımız kişi en son tanıyor mudiye sorduğumuz kişidir. $(n-1)$ soru. Gruptan seçtiğimiz kişi rastgele sorduğumuz birisini tanıyor ise soru sorduğumuz kişi aradığımız kişi değildir. Bu işlemi sorduğumuz kişi en son sorduğumuz kişiyi tanıyor ise aradığımız kişi en son sorduğumuz kişidir. $(n-1)$ soru.

Bir diğer durumda soru sorduğumuz kişi diğer hiçbir kişiyi tanımıyor ise oluşan durumdur. O da $n-1$ soru. Toplamda $3n-3$ soru sorulmuş olur en kötü durumda. Durumun karmaşıklığı $O(n)$ dir.

8) a) Decrease and conquer algoritması. Sıralı olmayan en büyük krep bulunur. Hemen altından spatula ile çevrilir. Eğer en büyük krep üstte ise birşey yapılmaz. Daha sonra krepların en altından spatula ile alt üst çevrilir. Böylece en büyük kek alta gelmiş olur. Sonra sondaki kekler sıralı olduğundan sondan alt üst çevirme olayı sıralı kekler öncesinden gerçekleştirilir.

b) 1, 2, 10, 7, 8, 3