Onur Demir 18011078

1. "Sequential Search" algoritmasının zaman karmaşıklığını, matematiksel analiz yardımı ile hesaplayıp Big-Oh, Big-Omega ve Big-Theta gösterimlerinden en uygun olanı ile aşağıdaki üç durum için ifade ediniz. **(20 Puan)**

   a. Best Case
   b. Average Case
   c. Worst Case

Sequential search algoritması aşağıdaki pseudo kod ile ifade edilebilir. (Tekrar yoksa.)

	cost	times
For i from 0 to n-1:	c_1	$1 \to n+1$
İf Li = T:	c_2	$1 \to n$
return i;	c_3	$1 \to n$
return -1	c_4	$3n+2 = \text{worst}$

Best $\to$ worst
3 $\quad$ $3n+2$

a.En iyi çalışma zamanında(best case) bulunacak eleman aranacak elemanın ilk eleman olması durumudur. $\Theta(1)$ İle ifade edilebilir. (Sırasıyla Big-O ve Big Omega analizi için ; c_3 de f(n) = 3 g(n) = 1 için f(n)<=c*g(n) c>=3 için ve c_3 de f(n) = 3 g(n) = 1 için f(n)>=c*g(n) c=2 için)

b.En kötü çalışma zamanı(worst case) elemanın bulunamamasıdır. $\Theta(n)$ İle ifade edilebilir. (Sırasıyla Big-O ve Big Omega analizi için ; c_4 de f(n) =3n+2 g(n)=n için f(n)<=c*g(n) c=4,n>2 için ve c_4 de f(n)=3n+2 g(n)=n için f(n) >= c*g(n) c=3,n>0 için) Eğer sequential search tekrarlı elemanı arıyor ise tekrar eden eleman sayısı kadar çıkarılır. Bu durumda $\Theta(n-k)$ olur.

c.Ortalama çalışma zamanı için tüm durumlar incelenmelidir. Bu durumda her eleman tek tek aranmalıdır ve aranma sayısına bölünmelidir.

$$\frac{k_1+k_2+k_3+\dots+k_n}{n} = \sum_{i=1}^{n} \frac{k_i}{n}$$

n: toplam arama sayısı
k_i: i. indexteki elemanı aramak için geçen cost
temel operatör karşılaştırmadır ve costu 1 dir buna göre

$$f(n) = \frac{1+2+\dots+n}{n} = \frac{n.(n+1)}{2} \cdot \frac{1}{n} = \frac{n+1}{2} \text{ (tekrarsız.)}$$

eğer k kadar eleman tekrar ediyorsa;

$$f(n) = \frac{n+1}{2(k+1)}$$ olur. $g(n) = n/k$ için

$$c_1 g(n) \le f(n) \le c_2 g(n)$$ $c_1 = 1/4$ $c_2 = 2$ için

$\Theta\left(\frac{n}{k}\right)$ şartını sağlar.

Buna göre algoritma çalışma zamanı $\Omega(1)$ ve $O(n)$

2. $\frac{1}{2}n(n-1) \in \Theta(n^2)$ ifadesinin doğruluğunu ispat eden çözümünüzü adım adım yazınız.**(20 Puan)**

$g(n) = n^2$, $f(n) = \frac{1}{2} n.(n-1)$ iken,

$c_1.g(n) \leq f(n) \leq c_2 g(n)$ şartı sağlanmalıdır.

$c_1 = \frac{1}{4}$, $c_2 = 1$ için;

$\frac{1}{4}n^2 \leq \frac{1}{2}n^2 - \frac{1}{2} \leq n^2$ $n \geq 2$ için sağladığından

$f(n) \in \Theta(n^2)$ denebilir.

3. Aşağıda verilen ifadeleri çözümleyerek "n" cinsinden sonucu yazınız. **(20 Puan)**

a. $\sum_{i=3}^{n+1} i$

b. $f(n) = \sum_{i=0}^{n-1} i(i+1)$

a. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \Rightarrow \sum_{i=3}^{n+1} i = \sum_{i=1}^{n+1} i - \sum_{i=1}^{2} i = \frac{(n+1)(n+2)}{2} - 3$

$\frac{n^2+3n-4}{2} = \frac{(n+4)(n-1)}{2}$ //

b. $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$, $\sum_{i=0}^{n} i^2 = \frac{n(2n+1)(n+1)}{6} \Rightarrow \sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} i = f(n)$

$f(n) = \frac{(n-1)(\overbrace{2(n-1)+1}^{2n-1})(n)}{6} + \frac{(n-1)n}{2} = \frac{(2n^2-n)(n-1)}{6} + \frac{3n^2-3n}{6}$

$= \frac{2n^3-3n^2+3n^2+n-3n}{6} = \frac{2n^3-2n}{6} = \frac{1}{3}(n^3-n)$

$= \frac{1}{3}n(n-1)(n+1)$ //

4. Aşağıda verilen rekürans bağıntısını "backward substitution" yardımı ile çözünüz. **(20 Puan)**
$x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)

$x(2^k) = x(2^{k-1}) + 2^k$ substitude $x(2^{k-1}) = x(2^{k-2}) + 2^{k-1}$

$x(2^{k-1}) = x(2^{k-2}) + 2^{k-1}$

$x(2^{k-2}) = x(2^{k-3}) + 2^{k-2}$

...

$x(2) = x(2^0) + 2^1$

...

$x(2^k) = \sum_{i=1}^{k} 2^i = 2(2^k - 1)$

$\boxed{x(n) = 2(n-1)}$

5. Verilen d tabanındaki m basamaklı bir sayıyı "Decimal" olarak ifade eden algoritmanın sözde kodunu yazıp karmaşıklığını ifade ediniz. **(20 Puan)**

~~ör~~ $(101)_2$, $(9AC)_{16}$ ...

$\underbrace{1.\boxed{2^0} + \boxed{2^1}.0 + \boxed{2^2}.1}$

Asıl basic işlem üs alma(çarpma)

her basamakta 3 temel işlem var. Üstler(basamaklar) tek adımda ifade edilemiyorsa ia bir döngüde ifade edilecektir.

```
Algorithm ToDecimal(A[0...n-1], K)                      cost
   " K: Base "
   " A: Number "
   number ← 0                                             1
   for i ← n-1 to 0 do                                    n+1
      digit ← 1                                           n
      for j ← 0 to i-n-1 do                               n(n+1)
         digit ← digit * K                                2n^2
      number ← number + digit * A[i]                      3n
   return number                                          1
```

$+$

$3n^2 + 6n + 3$

Time → $\Theta(n^2)$

Space → $\Theta(1)$

Algorithm ToDecimal_Opt(A[0 -- n-1], K)

		cost
	"K: Base"	
	"A: Number"	
	number ← 0	1
	pow ← 1	1
	for i ← n-1 to 0 do	n+1
	number ← pow * A[i] + number	3n
	pow ← pow * K	2n
	return number	1
		+
		6n + 4

Time: $\Theta(n)$: Tek durumlu işlem tanımlı

Space: $\Theta(1)$