Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 2 REPORT

Gözde DOĞAN
131044019

Course Assistant: Nur Banu ALBAYRAK

Contents

Contents 2
1.QUESTION 1 3
1.Question1_1 3
2.Question1_2 4
2.QUESTION 2 5
1.Question2_1 5
2.Question2_2 5
3.QUESTION 3 6
4.QUESTION 4 7
5.QUESTION 5 9
1. Question5_1 9
2. Question5_2 11
3. Question5_3 13
6.QUESTION 6 16
1. LinkedList(V1): 16
2. Array(V2): 18
3. ArrayList(V3): 20
7.PROBLEM SOLUTIONS APPROACH 23
8.TEST CASES 24
9.RUNNING AND RESULTS 26

1. QUESTION 1

1. Question1_1

1) for(int i=0; i<n-1; i++){
    for(int j = i+1; j<n; j++){
        3 simple statement
    }
}

→ statement ler constant time yani 1 kabul edilir!

3 simple statement = 3

$$\underbrace{\sum_{i=0}^{n-1}}_{\text{Distaki}} \underbrace{\sum_{j=i+1}^{n}}_{\text{icteki for}} 3 = 3\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} 1 = 3\sum_{i=0}^{n-1} n - \sum_{i=1}^{n-1} i$$

$$= 3.\left[n.(n-1) - \frac{(n-1).n}{2}\right] = 3.\left(n^2-2 - \frac{n^2-2}{2}\right)$$

$$= \frac{3}{2}(2n^2-4-n^2+2) = \frac{3n^2-6}{2} = \Theta(n^2)$$ worst case!

→ Best case de $n=1$ kabul edilir, distaki for 1, icteki for 0 defe döner. Statementlar da 3 olursa,

$3.0.1 = 0$ = constant time = $\Theta(1)$ olur!

$$T(n) = O(n^2)$$
$$= \Omega(1)$$

2. Question1_2

2) public static int length (String str){
if (str == null || str.equals("")
return 0;
else
return 1+length (str.substring(1));
}

→ equals ve substring metotlarının içi verilmediği için çalışma süreleri constant time olarak alındı!

Best case: str stringinin null veya boş olması!
Yani if'e girmesi!

str == null constant time
str.equals("") constant time
|| işlemi constant time

$T_{best}(n) = \Omega(1)$
$= \Theta(1)$

Worst case: str stringinin uzunluğuna n diyelim!
else'e girecek!
return işlemi constant time
+ işlemi constant time
length metodunun çağırılması constant time!
Ama recursive metot olduğu için;

$T_{worst}(n) = T(n-1) + \text{constant time}$
$T(n-1) = T(n-2) + \text{constant time}$

$T(n-k+1) = T(n-k) + \text{constant time}$ $k = n-1$ kabul edelim!
$T(1) = T(0) + \text{constant time}$ ile girmesi

$T_{worst}(n) = \text{constant time} + n.\text{constant time}$
$T_{worst}(n) = (n+1)\ \text{constant time}$: Linear time
$T_{worst}(n) = O(n)$
$= \Theta(n)$

2. QUESTION 2

1. Question2_1

1) SOME-FUNCTION (A)
n ← length[A]
for j ← 1 to n-1
do smallest ← j
for i ← j+1 to n
do if A[i] < A[smallest]
then smallest ← i
exchange A[j] ↔ A[smallest]

```
n = A.length;
for(j=1; j<n-1; j++){
  smallest = j;
  for(i=j+1; i<n; i++){
    if(A[i]< A[smallest]){
      smallest = i;
    }
  }
  temp = A[j];
  A[j] = A[smallest];
  A[smallest] = temp;
}
```

→ Arrayi sıralıyor! Büyükten küçüğe!

2. Question2_2

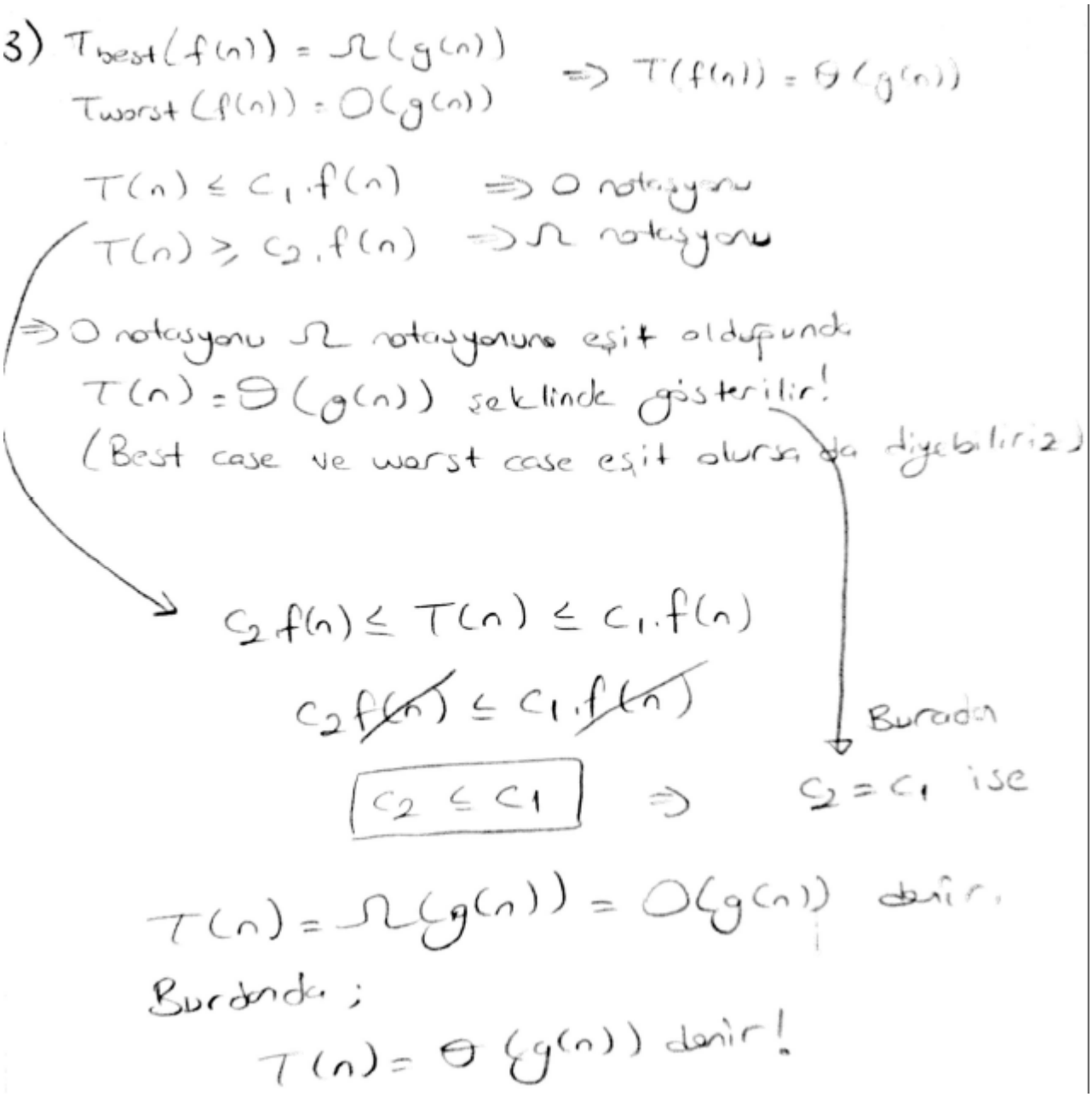
2) Best case:
Array sıralıdır!
Dıştaki for n-1, içteki for 0, assignmentlar constant time.

$T_{best}(n) = \Theta(1 + (n-1).(1 + (1.1) + 3))$
$= \Theta(1 + (n-1)(5))$
$= \Theta(5n-4)$
$= \Theta(n)$ ← linear zaman alır!
$= \Omega(n)$

worst case:
Array küçükten büyüğe sıralıdır!
Dıştaki for n-1, içteki for n, assignment lar constant time

$T_{worst}(n) = \Theta(1 + (n-1).(1 + n.1 + 3))$
$= \Theta(1 + (n-1).(n+4))$
$= \Theta(1 + n^2 + 3n - 4)$
$= \Theta(n^2 + 3n - 3)$
$= \Theta(n^2)$ ← quadratic time

3. QUESTION 3


3) $T_{best}(f(n)) = \Omega(g(n))$
$T_{worst}(f(n)) = O(g(n))$
$\Rightarrow T(f(n)) = \Theta(g(n))$
$T(n) \leq c_1 . f(n)$ $\Rightarrow$ O notasyonu
$T(n) \geq c_2 . f(n)$ $\Rightarrow \Omega$ notasyonu
$\Rightarrow$ O notasyonu Ω notasyonuna eşit olduğunda
$T(n) = \Theta(g(n))$ şeklinde gösterilir!
(Best case ve worst case eşit olursa da diyebiliriz)
$c_2 f(n) \leq T(n) \leq c_1 . f(n)$
$c_2 f(n) \leq c_1 . f(n)$
Buradan
$c_2 \leq c_1$ $\Rightarrow$ $c_2 = c_1$ ise
$T(n) = \Omega(g(n)) = O(g(n))$ dir.
Burdan;
$T(n) = \Theta(g(n))$ dir!

4. QUESTION 4

```
public static <T extends Comparable<T>> void
  Sort (T[] table) {

     insert (table, 0, table.length-1 , 0 );

  }
private static <T extends Comparable<T>>  void
 insert (T[] table, int first , int last , int counter ){

     if ( counter == last )
          return table;
      else {
         int index = first + 1;
         for(int i=first ; i< last-1 ; i++){
             if (table.get(i).compareTo (table.get(i+1)) >0)
                    index = i+1;
          }
           T temp = table.get (first);
           table.set (first, table.get (index));
           table.set (index , temp );
       }
      return insert (table, first+1 , last , counter +1 );

 }
```

→ length = n kabul edildi!

→ get , set , compareTo metotlar, icerikleri bilinmediği için constant time kabul e dildi!

insert metodu:

best case:

counter == last ← true ise!

OR

else şartında first değerinin last değerinden 1 eksik olması, for döngüsünün 1 kere dönmesi!

→ counter == last ; constant time!

→ first + 1 == last ; döngü 1 kere döndü, döngü içindeki if 1 kere çalıştı, insert metodu tekrar açıldı, if'e girip çıktı ; constant time!

$T_{best}(n) = \Omega(1)$ ← constant time
$= \Theta(1)$

worst case:

last == length ↑ n dedik! true ise ve first == 0 true ise

$T_{worst}(n)$ = initialize + for'un çalışması + swap + recursive kol

$= 1 + n \cdot a(1) + 3 +$ recursive kol

döngü n defa döner. ↑ if'in çalışması

$= 3 + n +$ recursive kol

(length-1) n-1 defa çağırılır!

$T_{worst}(n) = T(n-1) + 3 + n$
$T(n-1) = T(n-2) + 3 + n$
$\vdots$
$T(1) = T(0) + 3 + n$

$T_{worst}(n) = T(0) + n.(3+n)$
$T_{worst}(n) = \Theta(1) + n^2 + 3n$
$T_{worst}(n) = \max(\Theta(1), \Theta(n^2+3n))$
$T_{worst}(n) = \Theta(n^2)$ ← quadratic time
$= O(n^2)$

5. QUESTION 5

1. Question5_1

1) $f(n) = n^{0,1}$, $g(n) = \log n$

→ O notasyonu:

$$n^{0,1} \leq c \cdot \log n$$

$$\frac{n^{0,1}}{\log n} \leq c$$

$n = 1$ iken

$$1 \leq c$$

$n = k$ iken

$$\frac{k^{0,1}}{\log k} \leq c \text{ kabul edelim}$$

$n = k+1$ iken

$$\frac{(k+1)^{0,1}}{\log(k+1)} \leq c$$

$$\frac{(k+1)^{0,1}}{\log k} \leq c \Rightarrow k^{0,1} > (k+1)^{0,1}$$ olduğu için bu ifade doğrudur.

$$T(f(n)) = O(g(n))$$

→ Ω notasyonu

$$\frac{n^{0,1}}{\log n} \geq c$$

$n = 1$ iken

$$1 \geq c$$

$n = k$ iken

$$\frac{k^{0,1}}{\log k} \geq c \quad \text{kabul edelim}$$

$n = k+1$ iken

$$\frac{(k+1)^{0,1}}{\log(k+1)} \geq c \Rightarrow \frac{(k+1)^{0,1}}{\log k} \geq c$$

$k^{0,1} > (k+1)^{0,1}$ oldğu için

bu ifade yanlıştır.

$$T(f(n)) \neq \Omega(g(n))$$

→ Θ notasyonu

$$n^{0,1} \leq c \log n \leq n^{0,1}$$

Burası Ω notasyonundan gördüğü üzere yanlış!

Bu nedenle

$$T(f(n)) \neq \Theta(g(n))$$

2. Question5_2

2)

$f(n) = n!$, $g(n) = 2^n$

→ O notasyonu

$n! \leq c \cdot 2^n$

$\frac{n!}{2^n} \leq c$

n=1 iken

$1/2 \leq c$

n=k iken

$\frac{k!}{2^k} \leq c$ kabul edelim!

n=k+1 iken

$\frac{(k+1)k!}{2 \cdot 2^k} \leq c$ k>1 iken yanlıştır.

$T(f(n)) \neq O(g(n))$

→ Ω notasyonu;

$$n! \geq c \cdot 2^n$$

$$\frac{n!}{2^n} \geq c$$

$n=1$ iken

$$\frac{1}{2} \geq c$$

$n=k$ iken

$$\frac{k!}{2^k} \geq c$$ kabul edelim.

$n=k+1$ iken

$$\frac{(k+1).k!}{2.2^k} \geq c$$ $k>1$ iken doğrudur.

$$T(f(n)) = \Omega(g(n))$$

→ Θ notasyonu

$$\underbrace{n! \leq c \cdot 2^n}_{\text{Doğru}} \leq \underbrace{n!}_{\text{Doğru}}$$

O notasyonundan farklılığı üzere yanlış!

$$T(f(n)) \neq \Theta(g(n))$$

3. Question5_3

3)

$f(n) = (\log n)^{\log n}$, $g(n) = 2^{(\log n)^2}$

→ O notasyonu;

$(\log n)^{\log n} \leq c \cdot 2^{(\log n)^2}$

$(\log n)^{\log n} \leq c \cdot n^2$

$\frac{(\log n)^{\log n}}{n^2} \leq c$

$n = 1$ iken

$1 \leq c$

$n = k$ iken

$\frac{(\log k)^{\log k}}{k^2} \leq c$ kabul edelim

$n = k+1$ iken

$\frac{\log(k+1)^{\log(k+1)}}{(k+1)^2} \leq c \Rightarrow$ $\log(k+1) > \log k$

$(k+1)^2 > k^2$

yorlıştır.

$T(f(n)) \neq O(g(n))$

→ Ω notasyonu

$$\frac{(\log n)^{\log n}}{n^2} \geq c$$

$n = 1$ iken

$$1 \geq c$$

$n = k$ iken

$$\frac{(\log k)^{\log k}}{k^2} \geq c \quad \text{kabul edelim}$$

$n = k+1$ iken

$$\frac{[\log(k+1)]^{\log(k+1)}}{(k+1)^2} \geq c$$

$\log(k+1) > \log k$

$(k+1)^2 > k^2$ doğrudur!

$$T(f(n)) = \Omega(g(n))$$

→ Θ notasyonu;

$$(\log n)^{\log n} \leq c_1 \cdot n^2 \leq (\log n)^{\log n}$$

Yanlış — Ω notasyonundan daha büyük

Bu nedenle;

$$T(f(n)) \neq \Theta(g(n))$$

6. QUESTION 6

1. LinkedList(V1):

- linkedList'in getter, setter, equals ve toString metotlarının çalışma süresi constant time kabul edildi.

readFileAndFillArrays()

best case: Dosyanın boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Dosyada n satır(Dıştaki loop, çalışma süresi linear time(n)), satırda da n virgül olması(condition içinde ilk virgül öncesini aldıktan sonra geri kalanlar da next ile alındığı için satırda n virgül olması bir şey ifade etmez yani çalışma süresi constant time olur. Ama okunan her satır bir linkedList listesine add metodu ile eklenir ve linkedList'te add metodunun çalışma süresi linear time(n)dır.). Dıştaki loop'un çalışma süresi ile içteki loop'un çalışma süresi çarpılır(n*n = n^2) ve readFileAndFillArrays metodunun çalışma süresi quadratic time(n^2) olur!
O(n^2) = θ(n^2)

public void printToFileArrays()

best case: 3 listenin de boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Listede n eleman olduğu durum3 tane for döngüsü var, böylece çalışma süresi (n)+(n)+(n) = 3n, kat sayıların bir katkısı olmadığı için linear time bulunur. O(n) = θ(n)

public void showMenu()

best case: Kullanıcının çıkış yapması(printToFileArrays(çalışma süresi linear time, yani n) metodu çalışır), Ω(n) = θ(n), linear time

worst case: Kullanıcının doğru giriş yapması giriş yapması (logIn metodu çalışır, logIn metodunun çalışma süresi θ(n) dir. Doğru giriş yapılması tekrar input(giriş yapılmasının) istenmesi demektir. Tekrar girişlerin sayısını da n kabul edersek çalışma süresi n*n = n^2 olur. Yani quadratic time.) O(n^2) = θ(n^2)

public void logIn()

best case: Kullancının doğru input girmesi(listArray metodu çalışır 2 defa (n+n), kullanıcının doğru input girmesi("s" OR "u") kontrol edilir(1), conditionlardan true olana ya da false olana girmesi (ikisinin de çalışma süresi aynı, (select kontrolü(1)*(for(n) + if(n))))) Buradan da çalışma süresi 2n+1+1*(2n) = 4n+1 olarak bulunur. Sabit sayılar ve kat sayıların katkısı olmadığından çıkarılırsa çalışma süresi linear time(n) olur. O(n) = θ(n)

worst case: Kullanıcının n defa yanlış input girmesi, O(n) = θ(n) , linear time

T(n) = θ(n) olur!

private void showStaffMenu()

best case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmemesi durumudur. Böylece switch'in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir. do{}while'dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

private void showUserMenu(int index)

best case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmemesi durumudur. Böylece switch'in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir. do{}while'dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

private int polling(LinkedList list)

best case: Kullanıcının doğru input girmesi, Ω(1) = θ(1), constant time

worst case: Kullanıcının n defa yanlış input girmesi, O(n) = θ(n) , linear time

private void listArray(LinkedList list)

best case: Listenin boyutunun 0 olması, Ω(1) = θ(1), constant time

worst case: Listenin n elemana sahip olmasıBöylece çalışma süresi n yani linear time bulunur. O(n) = θ(n)

2. Array(V2):

readFileAndFillArrays()

best case: Dosyanın boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Dosyada n satır(Dıştaki loop, çalışma süresi linear time(n)), satırda da n virgül olması(condition içinde ilk virgül öncesini aldıktan sonra geri kalanlar da next ile alındığı için satırda n virgül olması bir şey ifade etmez yani çalışma süresi constant time olur. Ama her condtion da array boyutunun küçük olması durumu worst case olacağı için reallocate yapılacak, reallocate de Arrays.copyOf metodu(çalışma süresi n'dir) kullanıyor. Böylelikle içteki loop(hasNextElement)'un çalışma süresi linear time olur(n)). Dıştaki loop'un çalışma süresi ile içteki loop'un çalışma süresi çarpılır(n*n = n^2) ve readFileAndFillArrays metodunun çalışma süresi quadratic time(n^2) olur! O(n^2) = θ(n^2)

public void printToFileArrays()

best case: listelerin boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Listede n eleman olduğu durum, O(n) = θ(n) , linear time

private Array[] reallocate(Array[])

best case = worst case: Arrays.copyOf metodu(çalışma süresi n'dir) 2 defa kullanıyor.

Çalışma süresi n+n = 2n dir. Ama sabit sayının bir anlamı olmayacağı için çalışma süresi linear time(n) bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

public void showMenu()

best case: Kullanıcının çıkış yapması(printToFileArrays(çalışma süresi linear time, yani n) metodu çalışır), Ω(n) = θ(n), linear time

worst case: Kullanıcının doğru giriş yapması giriş yapması (logIn metodu çalışır, logIn metodunun çalışma süresi aşağıda da görüldüğü gibi θ(n) dir. Ama doğru giriş yapması sürekli tekrar giriş isteyecek ve biz bu tekrar girişlerin sayısını da n kabul edelim. Buradan da çalışma süresi n*n = n^2 olur. Yani quadratic time.) O(n^2) = θ(n^2)

public void logIn()

best case: Kullancının doğru input girmesi(listArray metodu çalışır 2 defa (n+n), kullanıcının doğru input girmesi("s" OR "u") kontrol edilir(1), conditionlardan true olana ya da false olana girmesi (ikisinin de çalışma süresi aynı, (select kontrolü(1)*(for(n) + if(n))))) Buradan

da çalışma süresi 2n+1+1*(2n) = 4n+1 olarak bulunur. Sabit sayılar ve kat sayıların katkısı olmadığından çıkarılırsa çalışma süresi linear time(n) olur. O(n) = θ(n)

worst case: Kullanıcının n defa yanlış input girmesi, O(n) = θ(n) , linear time

T(n) = θ(n) olur!

private void showStaffMenu()

best case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmemesi durumudur. Böylece switch'in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir. do{}while'dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

private void showUserMenu(int index)

best case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmemesi durumudur. Böylece switch'in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir. do{}while'dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch'in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

private int polling(LinkedList list)

best case: Kullanıcının doğru input girmesi, Ω(1) = θ(1), constant time

worst case: Kullanıcının n defa yanlış input girmesi, O(n) = θ(n) , linear time

private void listArray(LinkedList list)

best case = worst case: Listenin n elemana sahip olması, O(n) = θ(n) , linear time

T(n) = θ(n) olur!

3. ArrayList(V3):

readFileAndFillArrays()

best case: Dosyanın boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Dosyada n satır(Dıştaki loop, çalışma süresi linear time(n)), satırda da n virgül olması(condition içinde ilk virgül öncesini aldıktan sonra geri kalanlar da next ile alındığı için satırda n virgül olması bir şey ifade etmez yani çalışma süresi constant time olur.) Her condtion da array boyutunun küçük olması durumunda reallocate yapılacak, reallocate de addAll metodu(çalışma süresi n'dir) kullanıyor. Böylelikle içteki loop'un çalışma süresi linear time olur(n). Dıştaki loop'un çalışma süresi ile içteki loop'un çalışma süresi çarpılır(n*n = n^2) ve readFileAndFillArrays metodunun çalışma süresi quadratic time(n^2) olur!
O(n^2) = θ(n^2)

public void printToFileArrays()

best case: listelerin boş olduğu durum, Ω(1) = θ(1), constant time

worst case: Listede n eleman olduğu durum, O(n) = θ(n) , linear time

private Array[] reallocate(Array[])

best case = worst case: addAll metodu(çalışma süresi n'dir) 2 defa kullanıyor.

Çalışma süresi n+n = 2n dir. Ama sabit sayının bir anlamı olmayacağı için çalışma süresi linear time(n) bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

public void showMenu()

best case: Kullanıcının çıkış yapması(printToFileArrays(çalışma süresi linear time, yani n) metodu çalışır), Ω(n) = θ(n), linear time

worst case: Kullanıcının doğru giriş yapması giriş yapması (logIn metodu çalışır, logIn metodunun çalışma süresi aşağıda da görüldüğü gibi θ(n) dir. Ama doğru giriş yapması sürekli tekrar giriş isteyecek ve biz bu tekrar girişlerin sayısını da n kabul edelim. Buradan da çalışma süresi n*n = n^2 olur. Yani quadratic time.) O(n^2) = θ(n^2)

public void logIn()

best case: Kullancının doğru input girmesi(listArray metodu çalışır 2 defa (n+n), kullanıcının doğru input girmesi(“s” OR “u”) kontrol edilir(1), conditionlardan true olana ya da false olana girmesi (ikisinin de çalışma süresi aynı, (select kontrolü(1)*(for(n) + if(n))))) Buradan da çalışma süresi 2n+1+1*(2n) = 4n+1 olarak bulunur. Sabit sayılar ve kat sayıların katkısı olmadığından çıkarılırsa çalışma süresi linear time(n) olur. O(n) = θ(n)

worst case: Kullanıcının n defa yanlış input girmesi, O(n) = θ(n) , linear time

T(n) = θ(n) olur!

private void showStaffMenu()

best case: do{}while(choose!=4) döngüsü içindeki switch’in default koluna girmemesi durumudur. Böylece switch’in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir. (addBook(linear time): kitap adı yazarı alınır, listeye bakılır(linear time), yoksa add(constant time) metodu çalışır. addUser(linear time): user ID ve parola alınır, listeye bakılır(linear time), yoksa add(constant time) metodu çalışır. removeBook(linear time): polling metodu çalışır(linear time), remove metodu(linear time) çalışır.) do{}while’dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch’in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. O(n) = θ(n)

T(n) = θ(n) olur!

private void showUserMenu(int index)

best case: do{}while(choose!=4) döngüsü içindeki switch’in default koluna girmemesi durumudur. Böylece switch’in işlem yapan herhangi bir koluna girilmiş olunur ve işlem bitince de do{}while döngüsünden çıkılır. Bu kollardan her birinin çalışma süresi linear time(n)dir.(barrowBook(linear time): polling(linear time) metodu çalışır. Uygunsa ödünç verilir, değilse hata. returnBook(linear time): polling(linear time) metodu çalışır. Doğru kitapsa geri alınır, değilse hata.) do{}while’dan da çıkılacağı için yani bir kere döneceği için çalışma süresi constant time olur. showUserMenu metodunun çalışma süresi de n*1 = O(n) = θ(n) olur.

worst case: do{}while(choose!=4) döngüsü içindeki switch’in default koluna girmesi durumudur. Böylece sürekli input istenir(do{}while(choose!=4) döngüsü çalışır ve n defa çalıştığı

kabul edilirse). İnputun n defa isteneceği düşünülürse çalışma süresi linear time(n) olarak bulunur. $O(n) = \theta(n)$

T(n) = θ(n) olur!

private int polling(LinkedList list)

best case: Kullanıcının doğru input girmesi, $\Omega(1) = \theta(1)$, constant time

worst case: Kullanıcının n defa yanlış input girmesi, $O(n) = \theta(n)$, linear time

private void listArray(LinkedList list)

best case = worst case: Listenin n elemana sahip olması, $O(n) = \theta(n)$, linear time

T(n) = θ(n) olur!

7. PROBLEM SOLUTIONS APPROACH

Array(V2), ArrayList(V3) ve LinkedList(V1) kullanılarak yazılan programlar da kullanılan yapı, birkaç metod, bazı metodların içeriği ve kullanılan yapının zorunlu kıldığı birkaç değişiklik dışında bir farklılık yok. Farklılıkların bazıları;

- ArrayList(V3) ve LinkedList(V1) için getter ve setter metotlarının olması ama Array(V2)'de böyle bir metodun olmamasından kaynaklı elemana ulaşma biçimi farklı.
- LinkedList(V1) için bir capacity tutma ihtiyacı yok iken diğer versiyonlar için capacity değerinin olması.
- Array(V2)'de bir size tutma ihtiyacı var iken, ArrayList(V3) ve LinkedList(V1) için size'a gerek olmaması(Zaten kendileri tutuyor)
- Array(V2)'de remove yaparken size'ı bir azaltmak ve remove edilen elemanın yerine sıradaki elemanları kaydırmak gerekirken, ArrayList(V3) ve LinkedList(V1)'de sadece remove metodu çağırıldı ve bu dediklerimi remove metodu kendi içinde gerçekleştirmiş oldu.(Zaten LinkedList(V1) için kayırma değil sadece elemanın next'inin diğer elemanı göstermesi için birkaç assign gerçekleştirilir)

8. TEST CASES

Hepsinde test etme durumları aynı! Test etme şekilleri farklı! Yapılan sistem değişmedi, sistemin yapılma biçimi değişti. Bu nedenle HW1'deki test durumları hepsi için geçerlidir.

- Program başladığında ekrana gelen menüden doğru seçim yapılmama durumu:
  1 veye 2 girilmesi beklenir, başka bir şey girilirse tekrar seçim yapılması beklenir.
- Program başladığında ekrana gelen menüden 2 seçiminin yapılması durumu:
  2 seçilmesi demek programın kapatılması demektir.
- Program başladığında ekrana gelen menüden 1 seçiminin yapılması durumu:
  1 seçilmesi demek programa giriş yapılması demektir.
- Giriş yapılma işlemi başladığında kullanıcının tipi sorulur(staff-s or user-u)
  - Programa giriş yapılırken personel(staff) olarak giriş yapma durumu:
    Tip olarak s girilme durumudur. Sonrasında ID ve şifre girilmesi beklenir. Girilen ID ve şifre sistemde yoksa tekrar ID ve şifre girilmesi beklenir. Girilen ID ve şifre doğru ise personelin yapabileceklerinden oluşan bir menu ekrana gelir ve personelden seçim yapması beklenir. Menü de;
    - Kitap ekleme (2.AddBook)
    - Kullanıcı ekleme (3.AddUser)
    - Kitap çıkarma (4.RemoveBook)
    - Çıkış (5.Exit)
  - Menüdeki seçimler dışında bir sayı girilirse tekrar seçim yapılması beklenir
  - Kitap ekleme (2. AddBook) seçildiğinde kullanıcıdan kitap adı ve kitabın yazarının girilmesi beklenir. Kitap sistemde yoksa eklenir. Kitap sistemde varsa uyarı verilir. Ve tekrar menu ekrana getirilir.
  - Kullanıcı ekleme (3.AddUser) seçildiğinde kullanıcıdan user ID ve password girilmesi beklenir. User sistemde yoksa eklenir. User sistemde varsa uyarı verilir. Ve tekrar menu ekrana getirilir.
  - Kitap çıkarma (4. RemoveBook) seçildiğinde kullanıcıya kitap listesi verilir ve burden kitabı işaret eden sayılardan girmesi beklenir. Girilen sayı sistemdeki kitapları işaret etmiyorsa uyarı verilir ve tekrar seçim yapılması istenir. Girilen sayı sistemdeki kitaplardan birini işaret ediyorsa kitap sistemden çıkartılır. Ve tekrar menu ekrana getirilir.
  - Çıkış (5.Exit seçildiğinde) programın başına yani giriş yapmak ve çıkmak seçimlerinin olduğu yere dönülür. Seçim beklenir.

- Programa giriş yapılırken kullanıcı(user) olarak giriş yapma durumu:

Tip olarak u girilme durumudur. Sonrasında ID ve şifre girilmesi beklenir. Girilen ID ve şifre sistemde yoksa tekrar ID ve şifre girilmesi beklenir. Girilen ID ve şifre doğru ise kullanıcının yapabileceklerinden oluşan bir menu ekrana gelir ve kullanıcıdan seçim yapması beklenir. Menü de;

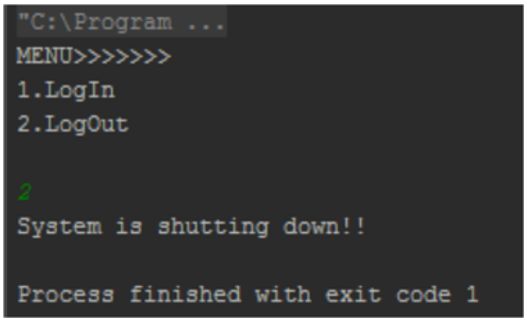
  - Kitap ödünç alma (2.BarrowBook)
  - Kitabı geri bırakma (4.ReturnBook)
  - Çıkış (4.Exit)

- Menüdeki seçimler dışında bir sayı girilirse tekrar seçim yapılması beklenir
- Kitap ekleme (2. BarrowBook) seçildiğinde kullanıcıya kitap listesi verilir ve burdan kitabı işaret eden sayılardan girmesi beklenir. Girilen sayı sistemdeki kitapları işaret etmiyorsa uyarı verilir ve tekrar seçim yapılması istenir. Girilen sayı sistemdeki kitaplardan birini işaret ediyorsa kitap kullanıcıya ödünç verilir ve sisteme ödünç almak için uygun değildir(notE) olarak kaydedilir. Ve tekrar menu ekrana getirilir.
- Kitap çıkarma (4. ReturnBook) seçildiğinde kullanıcıya kitap listesi verilir ve burdan kitabı işaret eden sayılardan girmesi beklenir. Girilen sayı sistemdeki kitapları işaret etmiyorsa ve kullanıcının ödünç aldığı bir kitap değilse uyarı verilir tekrar seçim yapılması istenir. Girilen sayı sistemdeki kitaplardan birini işaret ediyorsa ve kullanıcının ödünç aldığı kitap ise kitap sisteme alınması uygun(e) olarak kaydedilir. Ve tekrar menu ekrana getirilir.
- Çıkış (4.Exit seçildiğinde) programın başına yani giriş yapmak ve çıkmak seçimlerinin olduğu yere dönülür. Seçim beklenir.

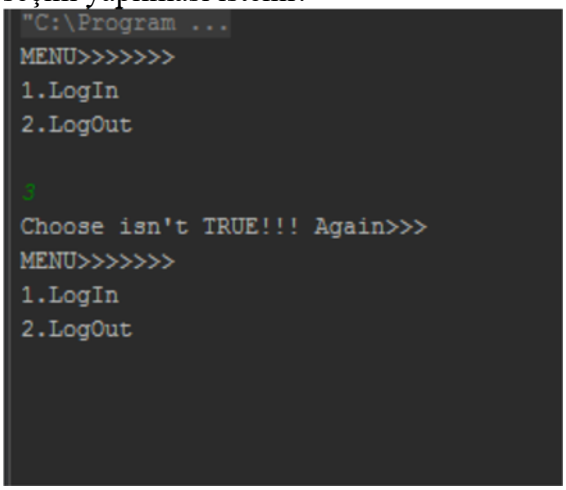
9. RUNNING AND RESULTS

1. LogOut durumu:

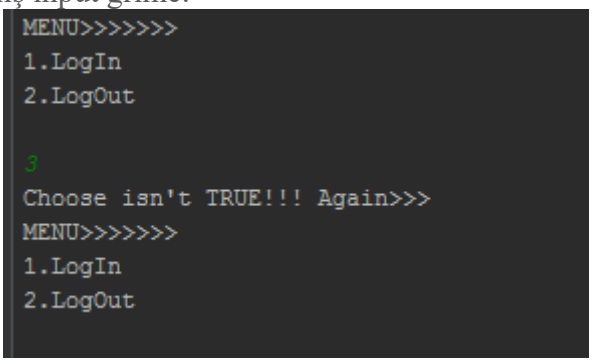
Programdan çıkılır.



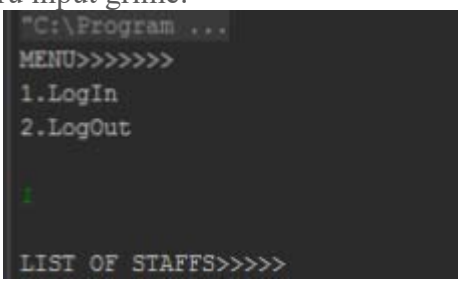
2. Yanlış seçim yapma durumu:
Tekrar seçim yapılması istenir.



3. LogIn durumu:
1. Yanlış input grime:



2. Doğru input grime:



1.Staff:

```
Enter user type(s or u):
s
Enter ID:
gozdedogan
Enter password:
gozde


MENU>>>>>
2.AddBook
3.AddUser
4.RemoveBook
5.Exit
```

1.AddBook seçildiğinde:

- Kitap adı ve yazarı girilmesi beklenir. Listede böyle bir kitap yoksa eklenir.

```
5.Exit
2

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN


Enter bookName:
Dava
Enter bookWriterName:
Franz Kafka


MENU>>>>>
2.AddBook
```

- Kitap listede varsa uyarı verilir ve menü tekrar ekrana getirilir.

```
5.Exit
2

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter bookName:
x-men
Enter bookWriterName:
Gozde
This book already exists in the library system! Again

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter bookName:
```

2.AddUser seçildiğinde:

- User için ID ve password girilmesi beklenir. Listede böyle bir user yoksa eklenir.

```
MENU>>>>>
2.AddBook
3.AddUser
4.RemoveBook
5.Exit
3
Enter user ID:
kerbasan
Enter parola:
260616


MENU>>>>>
2.AddBook
```

- User listede varsa uyarı verilir ve menu tekrar ekrana getirilir.

```
3.AddUser
4.RemoveBook
5.Exit
3
Enter user ID:
kadir
Enter parola:
erbasan
This user already exists in the library system! Again
Enter user ID:
```

3.RemoveBook seçildiğinde

- Girilen sayı listedeki kitaplardan birini gösteriyorsa kitap listeden çıkarılır.

```
3.ReturnBook
4.Exit

2

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter a number whose choose book>>
2


MENU>>>>>
2.BarrowBook
```

- Girilen sayı listedeki kitaplardan birini işaret etmiyorsa kullanıcıdan tekrar sayı girmesi beklenir.

```
3.ReturnBook
4.Exit

2

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter a number whose choose book>>
7
Your choose isn't true. Again Enter a number whose choose book>>
```

4.Exit seçilirse

Programdan çıkılır.

```
Enter user type(s or u):
s
Enter ID:
gozdedogan
Enter password:
gozde


MENU>>>>>
2.AddBook
3.AddUser
4.RemoveBook
5.Exit
5


MENU>>>>>>>
1.LogIn
2.LogOut
```

2. User:

```
Enter user type(s or u):
u
Enter ID:
elmas
Enter password:
parola


MENU>>>>>
2.BarrowBook
```

2.BarrowBook seçildiğinde

- Girilen kitap numarası listede varsa ve kitap başkası tarafından ödünç alınmadıysa kitap ödünç alındı olarak değiştirilir.

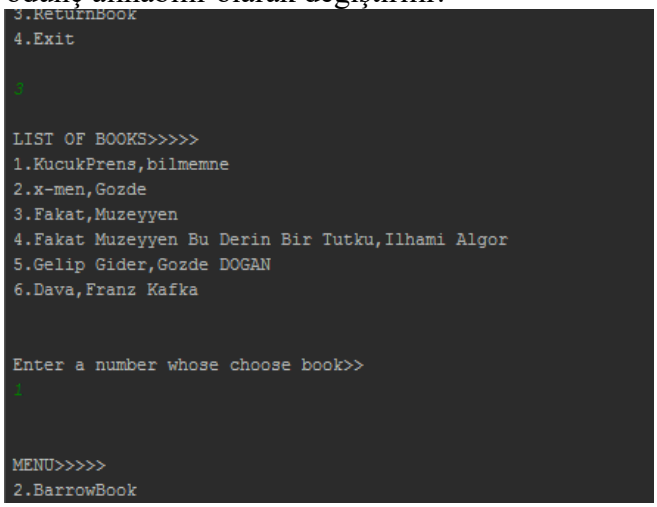
```
MENU>>>>>
2.BarrowBook
3.ReturnBook
4.Exit

2

LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter a number whose choose book>>
2


MENU>>>>>
2.BarrowBook
```

- Kitap başka user tarafından ödünç alındıysa bu user o kitabı ödünç alamaz.

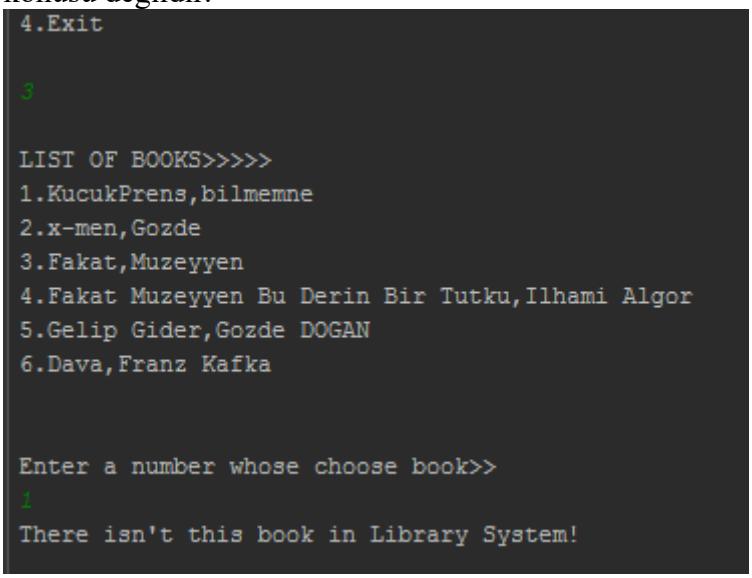
```
LIST OF BOOKS>>>>>
1.KucukPrens,bilmemne
2.x-men,Gozde
3.Fakat,Muzeyyen
4.Fakat Muzeyyen Bu Derin Bir Tutku,Ilhami Algor
5.Gelip Gider,Gozde DOGAN
6.Dava,Franz Kafka


Enter a number whose choose book>>
2
This book isn't eligible!


MENU>>>>>
2.BarrowBook
```

3.ReturnBook seçildiğinde

- Girilen kitap numarası listede varsa ve kitap ödünç alınmış bir kitap ise kitap ödünç alınabilir olarak değiştirilir.



- Girilen kitap numarası ödünç alınmamış bir kitap ise kitap için geri bırakılma söz konusu değildir.



4.Exit seçildiğinde
Programdan çıkılır.