Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 6 REPORT

AYDIN ÇALIKOĞLU
141044078

Course Assistant:
Nur Banu ALBAYRAK

1. Problem Solutions Approach

Q1 :

BinaryHeap Class;

BinaryHeap class'ını Comparable yapı alabilecek şekilde oluşturdum.

Binary Tree class 'ı içerisinden innerClass Node sınıfını güncelleyerek içerisine Parent Node ekledim. Bu Sayede ShiftUp işlemini logn zamanda yapabileceğim. Ancak Weight tutan bir değer ekleyerek last node a ulaşmak yerine, bir method geliştirdim. Bu method ile en alt da tutulan toplam node sayısna bakıyorum. Eğer bu Node sayısı treeLevel den büyük ise sağ küçük ise sol tarafa gideceğine karar veriyorum ve bu sayede logn zamanda son node üzerine de ulaşabiliyorum.

BinaryTree Alan contructor yapısını güncelleyerek. BinaryTree gelirse Iterator yardımı ile kendi yapımın içerisine ekledim. Böylece priority yapısnı sağladım.

Iterator Methodunu Level Order Iterator olarak override ettim. (Ekleme Yapısını Görebilmek için)

Ekleme işlemlerini için;

FindLastNode(root,size+1) isimli bir fonksiyon yazdım. O(logn) Zamanlı çalışıyor. Node sınıfına bu method sayesinde bir Node içerisinde öncelik değeri eklemek zorunda kalmadım.
Bu fonksiyon En alt node sayılarını kontrol ederek ve bos gideceği yöne karar veriyor. çocuğu ilk boş olan node geri döndürüyor.

AddLast() methodu ile bu bulduğum son node üzerine bir child ekliyorum. O(1) çalışıyor.

Son olarak da, Bu node için yazdığım shiftUp fonksiyonunu çağırıyorum. Bu fonksiyon O(logn) çalışıyor ve Node içerisindeki data exchange işlemleri ile data'yı gerekli bölgeye taşıyor.
Toplam Ekleme Süresi O(logn) = 2logn oluyor.

Silme işlemi için ;

FindLastNode(root,size) isimli aynı fonksiyon ile son nodu buluyorum. O(logn) çalışıyor.
Bulduğum node üzerindeki data'yı root'un datasına ekleyerek, bulunduğum node Parent node var ise parent node'dan bulduğum node siliyorum. Ardından ShiftDown işlemini çağırarak gerekli bölgeye taşınmasını sağlıyorum. Bu method O(logn) çalışıyor.
Silme işlemim çalışma süresi O(logn) = 2logn oluyor.

Passengers Class for Passengers Services ;

Comparable bir Passanger sınıfı oluşturdum, bu sınıfa identityNumber, totalGain, totalFlyCount ve age ekledim. (cinsiyet' de ekleyebilirdik)

Compare methodunu içersinde şu karşılaştırmaları yaptım ;

Kimlik Numaraları eşit olan yolculara eşitlik verdim. Böylece aynı kişi anlaşılabilecek şekilde Karşılaştırmalar yapılabilir duruma getirdim.

Kişileri öncelikle, Kişiler üzerinden kazandığımız toplam paraya göre değerlendirdim. Kazançlar Eşit ise uçuş sayışa göre değerlendirdim en az uçana öncelik verdirmemin nedeni daha az uçuş ile daha çok para kazandırmış, demek ki business uçuyor.

Uçuş sayıları da eşit ise yaşa göre öncelik verdim. Yaşlılara öncelik verdim.

Passengers Compare Summary

```
/**
ID Eşit ise Kişiler Eşit
Öncelik Sırası
    Firmanın Kişi Üzerinden Kazancı Yüksek Olana öncelik
    Kazanç Eşit ise
        Uçuş Sayısı az olana öncelik
        uçuş Sayısı Eşit ise
            yaşlı olanlara öncelik
*/
```

Q2 :

Huffman tree Class;

Freq.txt üzerinden okuduğum bilgiler ile HuffmanTree yapısını build edebilecek bir Fonksiyon oluşturdum. Bu fonksiyon her satır için huffman data yapısı oluşturuyor ve Bir arrayList içerisine ekliyor. Oluşan bu array ile kitabımızda ki buildTree fonksiyonunu kullanarak, HuffmanTree nin oluşmasını sağlıyorum.
Bu fonksiyonun çalışma süresi için T(N) için O(n^2) diyebiliriz.

Bu tree yi printCode fonksiyonu ile Oluşturarak çıktıları test ettim ve space charakteri için space freqNum şeklinde dosya içerisine yazmasını sağladım. Tree Build ederken aynı işlemi yapmasını sağladım.

Encode fonksiyonu için;
Recursive çalışan bir private encode fonksiyonu oluşturdum. Bu fonksiyon bir char için tree üzerinde gezerek, karaktere ulaşırken izlediği bit yolunu string olarak gönderiyor. Bu sayede bir karakter için kod alma işlemi oluşturdum.
T(N) = O(n^1)

Encode fonksiyonum içerisinde String yapısını karakterlere ayırarak, her karakter için private recursive encode fonksiyonunu çağırdım ve tüm string için encode işlemini tamamladım.
T(N) = O(n^2)

Frequency dosyasının oluşumu bir text file üzerinde ki tüm semboller için, hangi Sembolün kaç kez tekrarlandığını bularak, her sembolden en fazla 1 tane olacak şekilde Ve o sembol için kaç kez tekrarlandığını tutarak (huffdata'lar) oluşturulması gerekir.

Test işlemleri için Main üzerinde HuffmanTree.encode("blabla") şeklinde çağırarak çıktıları tekrar HuffmanTree.decode("çıktı") şeklinde test ettim ve blabla sonucunu aynı şekilde tekrar aldım. Bu işlemi tüm karakterler için denedim ve çıktı içerisinde karakter kayıpları olduğunu farkettim. Bu sorunun tree üzerinde gerçeklenmemiş karakterlerden dolayı olduğunu gözlemledim. Bu sorun için tree üzerinde olmayan elemanları aynı şekilde çıktısının verilmesi ve decode ederken 0 ve 1 haricindeki elemanların direk çıktısını sağlayarak yapının bozulmamasını çözümledim.

Q3 :

Class FamilyTree Level Order Iterator

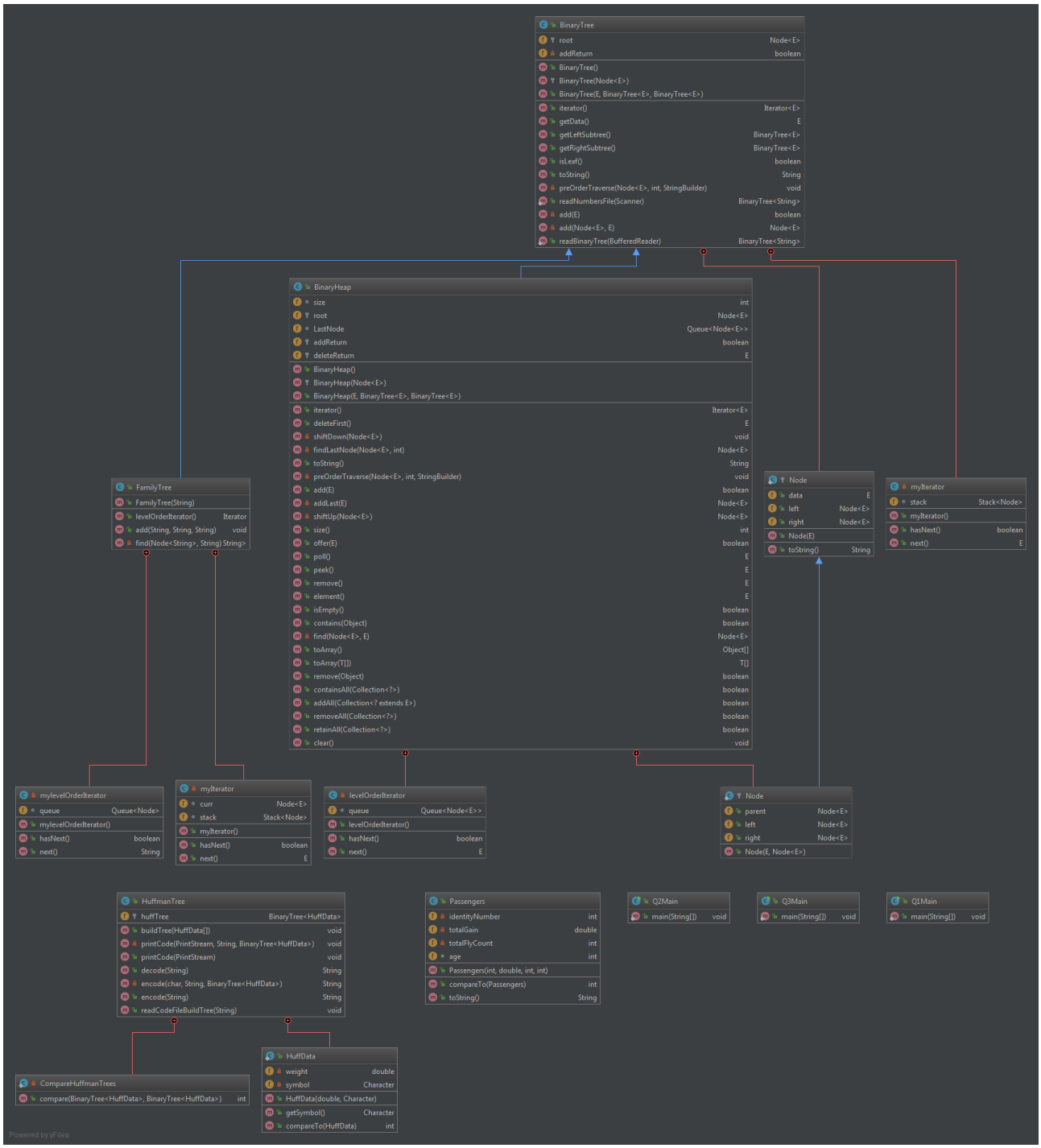
Level Order Iterator

Iterator'ün Level-order travers çalışabilmesi için bir Node tutan Queue yapısı kullandım.
Iterator'ünSınıfının Contructor kısmındaRoot node 'u Queue içineekledikten sonra
HasNext fonksiyonu için
Queuesınıfının size fonksiyonunu kullandım.
Next Fonksiyonumuz için
QueueCıkardığım Node elamanınınönce leftNodekısmınıQueueiçerisine ekledim.
Sonra
rightNode Queueiçerisine ekledim.
Çıkarmış olduğum Node içersindeki data return ettim.

Bu yöntem Sayesinde Sağ ve sol Node'laraulaştıktan sonra alt Node'lara ulaştım, böylece Tree yapısındaLevelOrderşekilde Travers edebildim.

Bu Yöntem sayesinde T(n) = O(1) çalışabildim.

2. Class Diagrams



3. Test Cases

Q1:

Ekleme ve Silme İşlemlerimin Sonuçları

```
BinaryHeap<Integer> binaryHeap = new BinaryHeap<>();
binaryHeap.add(new Integer(5));
binaryHeap.add(new Integer(2));
binaryHeap.add(new Integer(4));
binaryHeap.add(new Integer(3));
binaryHeap.add(new Integer(6));
binaryHeap.add(new Integer(7));
binaryHeap.add(new Integer(1));
```

```
System.out.println(binaryHeap.toString());
binaryHeap.remove();
System.out.println(binaryHeap.toString());
binaryHeap.poll();
System.out.println(binaryHeap.toString());
binaryHeap.remove();
System.out.println(binaryHeap.toString());
binaryHeap.remove();
```

```
1
  3
    5
      null
      null
    6
      null
      null
  2
    7
      null
      null
    4
      null
      null
2
  3
    5
      null
      null
    6
      null
      null
  4
    7
      null
      null
    4
      null
      null
3
  5
    7
      null
      null
    6
      null
      null
  4
    null
    null
5
  7
    null
    null
  6
    null
    null
```

HeapTree üzerinde Integer'lar üzerinden çalıştığını gözlemledik.
Passenger Sınıfımız ile gözlemleyelim.

Parameters: identityNumber, Kazanç, Uçuş Sayısı, Yaş

```
Passengers p1 = new Passengers(1,1000,5,25);
Passengers p2 = new Passengers(2,2000,5,25);
Passengers p3 = new Passengers(3,500,10,36);
Passengers p4 = new Passengers(3,500,10,36);
Passengers p5 = new Passengers(4,300,10,36);
Passengers p6 = new Passengers(5,300,5,36);
Passengers p7 = new Passengers(6,200,2,65);
Passengers p8 = new Passengers(7,200,2,36);
```

```
BinaryHeap<Passengers> binaryHeap1 = new BinaryHeap<>(p1,null,null);
BinaryHeap<Passengers> binaryHeap2 = new BinaryHeap<>(p2,null,null);
BinaryHeap<Passengers> binaryHeap3 = new BinaryHeap<>(p3,binaryHeap1,null);
BinaryHeap<Passengers> binaryHeap = new BnaryHeap<>(p4,binaryHeap2,binaryHeap3);
binaryHeap.add(p5);
binaryHeap.offer(p6);
binaryHeap.add(p7);
```

```
System.out.println(binaryHeap.toString());

binaryHeap.remove();
System.out.println(binaryHeap.toString());

binaryHeap.poll();
System.out.println(binaryHeap.toString());

binaryHeap.remove();
System.out.println(binaryHeap.toString());
```

```
2 2000.0 5 25
  3 500.0 10 36
    4 300.0 10 36
      null
      null
    5 300.0 5 36
      null
      null
  1 1000.0 5 25
    6 200.0 2 65
      null
      null
    7 200.0 2 36
      null
      null
```

```
1 1000.0 5 25
  3 500.0 10 36
    4 300.0 10 36
      null
      null
    5 300.0 5 36
      null
      null
  6 200.0 2 65
    7 200.0 2 36
      null
      null
    null
```

```
3 500.0 10 36
  5 300.0 5 36
    4 300.0 10 36
      null
      null
    7 200.0 2 36
      null
      null
  6 200.0 2 65
    null
    null
```

```
5 300.0 5 36
  4 300.0 10 36
    7 200.0 2 36
      null
      null
    null
  6 200.0 2 65
    null
    null
```

ID , Kazanç , Uçuş sayısı ve Yaş

2000 en çok kazanç sağladığımız Müşteri müşteriyi sıradan çıkardık.

1000 lira kazanç sağladığımızı çıkardık.

500 lira kazanç sağladığımızı çıkardık.

300 lira kazanç sağladığımız 2 kişi var.

Bunlardan ikisinin uçuş sayılarına baktık ve en az uçuş yapana öncelik verdik.

200 lira kazanç sağladığımız 2 kişi var.

Bu kişilerin uçuş sayıları da eşit

Bu kişileri yaşlarına göre kıyaslıyoruz.

```
4 300.0 10 36
  7 200.0 2 36
    null
    null
  6 200.0 2 65
    null
    null
```

```
6 200.0 2 65
  7 200.0 2 36
    null
    null
  null
```

ilk sırada yaşı büyük olan müşterimiz bulunmakta

Q2 :

HuffmanTree Class TEST;

```
PrintStream pS2=new PrintStream(new File("freq.txt"));
for (int i = 0; i < 128; i++) {

    if(i<10)
        pS2.println(i + " " + (new Double(Math.pow(i, 2))).intValue());
    else if(i>65)
    {
        char ch = (char) i;
        pS2.println(ch + " " + (new Double(Math.pow(i, 2))).intValue());
    }
}
pS2.println("space 1000002");
pS2.close();
```

```
hf.readCodeFileBuildTree("freq.txt");
```

Code blogunun benzerleri ile farklı şekilde dosyalar oluşturdum. Bu code bloğu tüm harfleri içeren bir çıktı dosyası üretiyor, bu çıktı dosyasının farklı Frequency değerleri ile oluşmasını sağladım ve printCode üzerinden oluşan ağaç yapılarını gözlemledim.

```
System.out.println((hf.encode("yi ga")));
System.out.println(hf.decode(hf.encode("yi ga")));
```

İçin aldığım sonuçlar;
0101111100000110110000010100010 11
yi ga

```
System.out.println((hf.encode("yi 123ga")));
System.out.println(hf.decode(hf.encode("yi 123ga")));
```

İçin aldığım sonuçlar;
01111000100011100000101010000000101011000001010001000000010111
yi 123ga

```
System.out.println((hf.encode("yi 123ga.*-")));
System.out.println(hf.decode(hf.encode("yi 123ga.*-")));
```

İçin aldığım sonuçlar
01111000100011100000101010000000101011000001010001000000010111
yi 123ga
.*- sembolleri tree içerisinde oluşturmadığımdan onlar olmadan bir sonuç çıkardı. Bunu çözmek için StringIndexOutOfBoundsException fırlatabilirdik.

Print Code Çıktıları İnceleyelim;
u 000000
v 000001
w 000010
T 0000110
U 0000111
x 000100
y 000101
z 000110
V 0001110
W 0001111
{ 001000
| 001001
} 001010
X 0010110
Y 0010111
~ 001100
□ 001101
Z 0011100
[0011101
\ 0011110
] 0011111
^ 0100000
_ 0100001
C 01000100
D 01000101
` 0100011
5 010010000000
0 0100100000010000
1 0100100000010001
2 010010000001001
3 01001000000101
4 0100100000011
8 01001000001
9 01001000010
6 010010000110
7 010010000111
B 010010001
E 01001001
a 0100101
b 0100110
c 0100111
F 01010000
G 01010001
d 0101001

e 0101010
f 0101011
H 01011000
I 01011001
g 0101101
h 0101110
i 0101111
J 01100000
K 01100001
j 0110001
k 0110010
l 0110011
L 01101000
M 01101001
m 0110101
n 0110110
o 0110111
N 01110000
O 01110001
p 0111001
q 0111010
P 01110110
Q 01110111
r 0111100
s 0111101
t 0111110
R 01111110
S 01111111
space 1

oluşturulan 1 ve 0 değerlerinin kullanım sayısına göre oluştuğunu gözlemliyoruz.

i^2 şeklinde oluşturduğumuzdan kullanım sayıları index değerlerine göre belirlendi.

Q3 :

Class FamilyTree

İnsert values;
Hasan
Ayşe, Hasan, ebu-Ayşe
Ali, Ayşe, ibn-Hasan
Sema, Hasan, ebu-Ayşe

TEST LEVEL ORDER ITERATOR ;

Test Code

```
        Iterator it3= ft.levelOrderIterator();
while (it3.hasNext())
{
    System.out.println(it3.next());
}
```

Test Output
Hasan
Ayşe
Ali
Sema

Level order bir sıralama ile çıktımızı ekrana bastırdık.