T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

Algoritma Analizi DERSİ

Final Soruları 2 ve 9

Heydar Binaliyev

110401066

Dersin Hocası

Ismail Kahraman

Soru-2

Heap yapısı N tane elemandan oluşan ikili ağaç şeklinde bir yapıdır.Bir kök düğümü vardır ve her düğümün en fazla iki çocuk düğümü vardır.Yığın yapısı bir grup eleman arasında önceliği en yüksek olan elemanı hızlıca bulmak için kullanılan veri yapısıdır.Maximum Heap veya Minimum Heap olarak oluştururlar.Maximum Heap'de bir grup elemandan en büyüğü yapının kökünü temsil etmektedir,ve daha küçük elemanlar ağacın aşağısına döğru yerleştirilmektedirler.Minumum Heap ise bunu tam tersidir.Bu veri yapısındaki elemanlar bellekte bir dizi ieçrisinde veya bağli liste ile tutulurlar.Bu veri yapısına eleman eklemenin karmaşıklığı en kötü durumda **O(logn)** dir.Çünkü bu yeni eklenen eleman tüm diğer elemanlardan daha büyükse (Max Heap için) bu ikili ağacın yeniden düzenlenmesi gerekecektir.Sileme işlemide ekleme işlemine benzerdir ve bunun karmaşıklığı **O(logn)** dir.Arama işleminin karmaşıklığı ise **O(n)** dir,çünkü bir arama yapılırken tüm elemanlar tek tek kontrol edilmektedir.

Kuyruk yapısında da elemanlar Heap veri yapısında olduğu gibi bir dizide tutulmaktadır.Bu veri yapısının en önemli işlemleri veri ekleme ve silmedir.Kuyruga yeni bir eleman eklendiyinde bu kuyrugun sonuna eklenir,bir eleman silindiyinde ise kuyrugun başındaki eleman silinir.Yani FIFO(First in First out) mantıgıyla çalışmaktadır.Ekleme ve silme işlemlerinin karmaşıklığı **O(1)** dir.Aramada ise **O(n).**

Bu iki veri yapısı arasındaki benzerliklerden biriside ekleme işlemindedir.Her ikisinde yeni bir eleman eklendiyinde dizinin sonuna eklenir.Fakat Heap de eklenen elemanın değerine göre ağaçda düzenleme yapılmaktadır.

Aşağıdaki program çıktısı 100,500,1000,5000,10000,50000 elemanlı heap ve kuyruk yapısında ekleme,silme ve arama işlemlerinin harcadığı zaman değerlerini göstermektedir.

```
-----------------Insert İşlemi---------------------
------100 boyutu için-------
Heap...
zaman 7 ms
Kuyruk...
zaman 3 ms
------500 boyutu için-------
Heap...
zaman 31 ms
Kuyruk...
zaman 1 ms
------1000 boyutu için-------
Heap...
zaman 3 ms
Kuyruk...
zaman 2 ms
------5000 boyutu için-------
Heap...
zaman 138 ms
Kuyruk...
zaman 20 ms
```

```
------10000 boyutu için-------
Heap...
zaman 655 ms
Kuyruk...
zaman 1 ms
------50000 boyutu için-------
Heap...
zaman 17052 ms
Kuyruk...
zaman 5 ms
-----------------Delete İşlemi----------------------
------100 boyutu için-------
Heap...
zaman 1 ms
Kuyruk...
zaman 0 ms
------500 boyutu için-------
Heap...
zaman 22 ms
Kuyruk...
zaman 0 ms
------1000 boyutu için-------
Heap...
zaman 3 ms
Kuyruk...
zaman 0 ms
------5000 boyutu için-------
Heap...
zaman 141 ms
Kuyruk...
zaman 1 ms
------10000 boyutu için-------
Heap...
zaman 543 ms
Kuyruk...
zaman 1 ms
------50000 boyutu için-------
Heap...
zaman 59244 ms
Kuyruk...
zaman 3 ms
-----------------Find İşlemi----------------------
------100 boyutu için-------
Heap...
zaman 1 ms
Kuyruk...
zaman 0 ms
------500 boyutu için-------
Heap...
zaman 7 ms
Kuyruk...
zaman 0 ms
------1000 boyutu için-------
Heap...
zaman 1 ms
Kuyruk...
zaman 0 ms
------5000 boyutu için-------
Heap...
zaman 111 ms
Kuyruk...
zaman 2 ms
```

```
------10000 boyutu için-------
Heap...
zaman 441 ms
Kuyruk...
zaman 1 ms
------50000 boyutu için-------
Heap...
zaman 36587 ms
Kuyruk...
zaman 2 ms
```

Kuyruk.java

```java
package soru2;

class Dugum{
	int data;
	public Dugum(int key){
		data=key;
	}
}

public class Kuyruk {
	int bas;
	int son;
	int N;
	Dugum[]dugumler;
	public Kuyruk(int N){
		bas=0;
		son=0;
		this.N=N;
		dugumler=new Dugum[N];
	}
	public boolean QBosmu(){
			if(bas!=son)
				return false;
			return true;
	}
	public boolean insert(int key){
		Dugum d=new Dugum(key);
		if(this.bas==this.son){
			this.dugumler[bas]=d;
			return true;
		}
		if(son==N){
			System.out.println("Kuyruk Dolu...");
			return false;
		}
		son++;
		this.dugumler[son]=d;
		return true;
	}
	public boolean delete(int key){
		if(QBosmu())
```

```
                return false;
            int index=0;
            for (Dugum dugum : dugumler) {
                if(dugum.data==key)
                    break;
                index++;
            }
            if(bas==index){
                bas++;
                return true;
            }else if(son==index){
                son--;
                return true;
            }else{
                for(int i=index;i<son;i++){
                    dugumler[i]=dugumler[i+1];
                }
                son--;
            }
            return true;
      }
      public int find(int key){
            int index=0;
            if(bas==son)
                return -1;
            for (Dugum dugum : dugumler) {
                if(dugum.data==key)
                    return index;
                index++;
            }
            return -1;
      }

}
```

Heap.java

```
package soru2;

class Node{
      int data;
      public Node(int key){
            this.data=key;
      }
      public int getKey(){
            return data;
      }
}
public class Heap{

      private int maxSize;
```

```
private   int elements=0;
private Node[] nodes;

public Heap(int size){
      this.maxSize=size;
      nodes=new Node[maxSize];
}

public boolean insert(int key){
      Node n=new Node(key);
      if(elements==0){
            nodes[0]=n;
            elements++;
            return true;
      }
      if(maxSize<elements){
            System.out.println("heap doldu");
            return false;
      }
      nodes[elements]=n;
      elements++;
      int index=elements-1;
      for(int i=index;i>0;i--){
            if(nodes[i-1].getKey()<n.getKey()){
                  Node temp=nodes[i-1];
                  nodes[i-1]=n;
                  nodes[i]=temp;
            }else
                  break;
      }
      return true;
}
public boolean delete(int key){
      if(elements<1)
            return false;
      int index;
      for(index=0;index<elements;index++){
            if(nodes[index].getKey()==key)
                  break;
            if(index==elements-1)
                  return false;
      }
      for(int i=index;i<elements-2;i++){
            if(nodes[i+1].getKey()<nodes[i+2].getKey()){
                  nodes[i]=nodes[i+2];
                  nodes[i+2]=nodes[i+1];
            }
      }
      elements--;
      return true;
}
```

```
	public int find(int key){
		int index;
		for(index=0;index<elements;index++)
			if(nodes[index].getKey()==key)
				return index;
		return -1;
	}

}
```

Karsilastirma.java

```
package soru2;

import java.util.Date;
import java.util.Random;

public class Karsilastirma {
	Heap maxHeap;
	Kuyruk kuyruk;
	Random r=new Random();
	Date once;
	Date sonra;
	long once_san,sonra_san;
	int boyutlar[]={100,500,1000,5000,10000,50000};

	public void insert(){
		System.out.println("----------------Insert İşlemi----------------------");
		for (int i : boyutlar) {
			System.out.println("------"+i+" boyutu için"+"-------");
			System.out.println("Heap...");
			once=new Date();
			once_san=once.getTime();
			maxHeap=new Heap(i);
			for(int j=0;j<i;j++){
				maxHeap.insert(r.nextInt(i));
			}
			sonra=new Date();
			sonra_san=sonra.getTime();
			System.out.println("zaman "+(sonra_san-once_san)+" ms");
			System.out.println("Kuyruk...");
			once=new Date();
			once_san=once.getTime();
			kuyruk=new Kuyruk(i);
			for(int j=0;j<i;j++){
				kuyruk.insert(r.nextInt(i));
			}
			sonra=new Date();
			sonra_san=sonra.getTime();
			System.out.println("zaman "+(sonra_san-once_san)+" ms");
		}
	}
```

```
public void delete(){
	System.out.println("-----------------Delete İşlemi----------------------");
	for (int i : boyutlar) {
		System.out.println("------"+i+" boyutu için"+"-------");
		System.out.println("Heap...");
		maxHeap=new Heap(i);

		for(int j=0;j<i;j++)
			maxHeap.insert(r.nextInt(i));

		once=new Date();
		once_san=once.getTime();

		for(int j=0;j<i;j++){
			maxHeap.delete(r.nextInt(i));
		}
		sonra=new Date();
		sonra_san=sonra.getTime();
		System.out.println("zaman "+(sonra_san-once_san)+" ms");
		System.out.println("Kuyruk...");
		kuyruk=new Kuyruk(i);
		for(int j=0;j<i;j++)
			kuyruk.insert(r.nextInt(i));
		once=new Date();
		once_san=once.getTime();

		for(int j=0;j<i;j++){
			kuyruk.delete(r.nextInt(i));
		}
		sonra=new Date();
		sonra_san=sonra.getTime();
		System.out.println("zaman "+(sonra_san-once_san)+" ms");
	}
}
public void find(){
	System.out.println("-----------------Find İşlemi----------------------");
	for (int i : boyutlar) {
		System.out.println("------"+i+" boyutu için"+"-------");
		System.out.println("Heap...");
		maxHeap=new Heap(i);
		for(int j=0;j<i;j++)
			maxHeap.insert(r.nextInt(i));
		once=new Date();
		once_san=once.getTime();
		for(int j=0;j<i;j++){
			maxHeap.find(r.nextInt(i));
		}
		sonra=new Date();
		sonra_san=sonra.getTime();
		System.out.println("zaman "+(sonra_san-once_san)+" ms");
		System.out.println("Kuyruk...");
		kuyruk=new Kuyruk(i);
```

```
                for(int j=0;j<i;j++){
                        kuyruk.insert(r.nextInt(i));
                }
                once=new Date();
                once_san=once.getTime();
                for(int j=0;j<i;j++){
                        kuyruk.find(r.nextInt(i));
                }
                sonra=new Date();
                sonra_san=sonra.getTime();
                System.out.println("zaman "+(sonra_san-once_san)+" ms");
        }
    }
    public static void main(String[] args) {
                new Karsilastirma().insert();
                new Karsilastirma().delete();
                new Karsilastirma().find();
    }
}
```

Soru-9

Bu problemi çözmek için elde bulunan ve çantaya yerleştirilebilecek olan elemanlar ağırlık->değer(key->value) şeklinde bir HashMap veri yapısı kullanılarak tanımlanmıştır.Bu elemanların birli,ikili,üçlü vs N'li kombinasyonu(N=>eleman sayısı) bulunarak toplamn ağırlıklarının 15'e eşit olup olmadığı kontrol edilerek,toplam ağırlığı 15 olan kombinasyonların arasından değeri en yüksek olan kombinasyon seçilmiştir.Bu programda kümeleme yapıldığı bu kümelerin kontrolü yapıldığı için zaman karmaşıklığı kombinasyon sayısına göre değişecektir.Kombinasyon sayısı ise 2^N dir(N=>eleman sayısı).Yani aşağıdaki kodun karmaşıklığı O(2^N) olacaktır.

CantaDoldur.java

```
package soru9;
import java.util.HashMap;
public class CantaDoldur {

        public int get_combinasyon_agirligi(int[]indises,Object[]agirliklar){
                int toplam=0;
                for(int i=0;i<indises.length;i++){
                        toplam+=(Integer)agirliklar[indises[i]];
                }
                return toplam;
        }
        public void kombinasyonu_yaz(HashMap<Integer, Integer>tablo ,Object[]agirliklar
                                        ,int[]indises){
                for(int i=0;i<indises.length;i++){
                        System.out.print((Integer)agirliklar[indises[i]]+" kg ");
                }
                System.out.println();
                System.out.println("degeri:"+new CantaDoldur().get_combinasyon_degeri
                                        (tablo, indises, agirliklar));
```

```
}
public int get_combinasyon_degeri (HashMap<Integer,Integer>tablo, int[]indises,
                                   Object[]agirliklar){
	int toplam=0;
	for(int i=0;i<indises.length;i++){
		toplam+=tablo.get((Integer)agirliklar[indises[i]]);
	}
	return toplam;
}
public void en_buyuk_deger_bul(HashMap<Integer, Integer>tablo){
	int en_buyuk_toplam_deger=0;
	int[]en_buyuk_indises = null;
	Object[] agirliklar=tablo.keySet().toArray();

	for(int kacli=1;kacli<=tablo.size();kacli++){

		CombinationGenerator x=new CombinationGenerator (agirliklar.length,
                                                                 kacli);
		int[]indises;
		while(x.hasMore()){
			//combinasyonun indexlerini al
			//0 2 5
			//0 2 4
			//0 2 3
			indises=x.getNext();

			int agirlik=new CantaDoldur().get_combinasyon_agirligi
                                                        (indises,agirliklar);
			int deger=new CantaDoldur().get_combinasyon_degeri(tablo,
                                                        indises, agirliklar);

			if(agirlik==15){
				new CantaDoldur().kombinasyonu_yaz(tablo, agirliklar,
                                                                indises);
				if(en_buyuk_toplam_deger<deger){
					en_buyuk_toplam_deger=deger;
					en_buyuk_indises=new int[indises.length];
					for(int i=0;i<indises.length;i++)
						en_buyuk_indises[i]=indises[i];
				}
			}
		}
	}
	System.out.println("--------------Seçilenler---------------");
	new CantaDoldur().kombinasyonu_yaz(tablo, agirliklar, en_buyuk_indises);
}
public static void main(String[] args) {
	//kg->değer
	HashMap<Integer, Integer>tablo=new HashMap<>();
	//tablo.put(kg,değer)
	tablo.put(1,2);
	tablo.put(2, 3);
	tablo.put(4, 4);
	tablo.put(5,6);
	tablo.put(8,9);
	tablo.put(7, 11);
	tablo.put(10, 12);
	tablo.put(12, 14);
	tablo.put(3, 8);
	tablo.put(6, 10);
	tablo.put(15, 16);
	tablo.put(13, 15);
```

```
            tablo.put(14, 14);
            new CantaDoldur().en_buyuk_deger_bul(tablo);

      }
}
```

Program Çıktısı:

```
15 kg
degeri:16
1 kg 14 kg
degeri:16
2 kg 13 kg
degeri:18
3 kg 12 kg
degeri:22
5 kg 10 kg
degeri:18
7 kg 8 kg
degeri:20
1 kg 2 kg 12 kg
degeri:19
1 kg 4 kg 10 kg
degeri:18
1 kg 6 kg 8 kg
degeri:21
2 kg 3 kg 10 kg
degeri:23
2 kg 5 kg 8 kg
degeri:18
2 kg 6 kg 7 kg
degeri:24
3 kg 4 kg 8 kg
degeri:21
3 kg 5 kg 7 kg
degeri:25
4 kg 5 kg 6 kg
degeri:20
1 kg 2 kg 4 kg 8 kg
degeri:18
1 kg 2 kg 5 kg 7 kg
degeri:22
1 kg 3 kg 4 kg 7 kg
degeri:25
1 kg 3 kg 5 kg 6 kg
degeri:26
2 kg 3 kg 4 kg 6 kg
degeri:25
1 kg 2 kg 3 kg 4 kg 5 kg
degeri:23
--------------Seçilenler---------------
1 kg 3 kg 5 kg 6 kg
degeri:26
```