Süre: 75 Dakika

Manisa Celal Bayar Üniversitesi Yazılım Mühendisliği Bölümü

YZM 2116 – Veri Yapıları Dersi

Arasınav Örnek Soruları

Bahar 2018

Adı ve Soyadı	YANIT ANAHTARI	Öğrenci Numarası	
Grubu		İmza	
Tarih		Not	/100

Soru#1 (20 puan): Aşağıdaki kod parçalarının hesaplama karmaşıklıklarını (koşma süreleri bakımından), *Big-O* hesaplama kuralları doğrultusunda değerlendirerek, en kötü durum senaryoları için *Big-O* değerlerini belirleyiniz. Ara işlemlerin yazılması zorunlu değildir.

Sözde Kod		Karmaşıklık Sınıfı (Big-O)

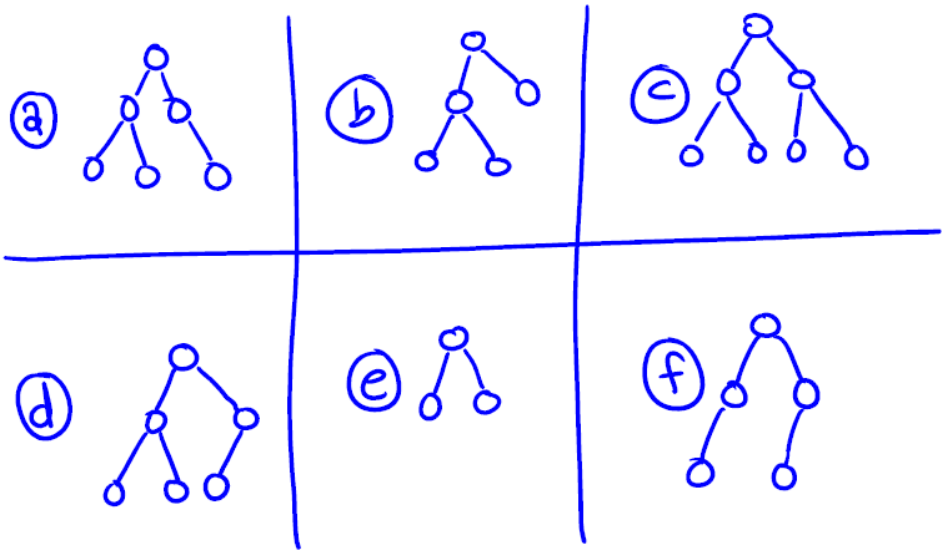
i.	`void deneme(int n) {`
	`  for (int i = 0; i < 1000; ++i) {`
	`  for (int j = 0; j < n; ++j) {`
	`  for (int k = 0; k < j; ++k)`
	`  System.out.println("k = " + k);`
	`  for (int m = 0; m < i; ++m)`
	`  System.out.println("m = " + m);`
	`  }`
	`  }`
	`}`
		$O(N^2)$

ii.	`void deneme(int n, int m) {`
	`  if (n < 5) return;`
	`  if (n < m)`
	`  deneme(n/2, m+1);`
	`  else`
	`  deneme(n/2, m);`
	`}`
		$O(logN)$

iii.	`void deneme(int n, int x, int y) {`
	`  for (int i = 0; i < n; ++i){`
	`  if (x < y) {`
	`  for (int j = 0; j < n * n; ++j)`
	`  System.out.println("j = " + j);`
	`  } else`
	`  System.out.println("i = " + i);`
	`  }`
	`}`
		$O(N^3)$

iv.	`void deneme(int n) {`
	`  for (int i = 0; i < n; ++i) {`
	`  j = 0;`
	`  while (j < n) {`
	`  System.out.println("j = " + j);`
	`  j++;`
	`  }`
	`  }`
	`}`
		$O(N^2)$

Soru#2 (10 puan): Aşağıda **a, b, c, d, e ve f** şeklinde isimlendirilmiş **altı adet ağaç** verilmiştir:

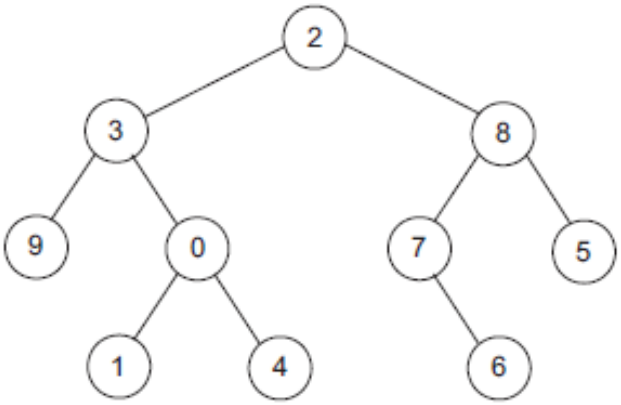


Buna göre, yukarıdaki "a, b, c, d, e ve f" **ağaçlarını, tam ikili ağaç** *(full binary tree),* **eksiksiz ikili ağaç** *(complete binary tree)* **ve katı ikili ağaç** *(strict binary tree)* **özellikleri taşımalarına göre gruplandırıp aşağıdaki tabloyu doldurunuz.** Not: Herhangi bir ağaç, birden fazla özelliği taşıyor olabilir.

İkili Ağaç Özelliği	İlgili Özelliği Örnekleyen/Taşıyan Ağaçlar
Tam İkili Ağaç	**c, e**
Eksiksiz İkili Ağaç	**b, c, d, e**
Katı İkili Ağaç	**b, c, e**

- **Tam İkili Ağaç: Her bir düğümün (i)net olarak iki çocuk düğüme sahip olduğu ve (ii)yaprak düğümlerin aynı seviyede olduğu ikili ağaçtır. Her düğüm eşit şekilde sağ ve sol alt-ağaçlara sahiptir.**
- **Eksiksiz İkili Ağaç: Son seviye dışındaki tüm seviyelerin tam (full) olduğu ikili ağaç türüdür. Düğümleri sol taraftan (düğüme göre) doldurulur. Yeni bir derinliğe soldan sağa doğru ekleme başlanır.**
- **Katı İkili Ağaç: Yaprak düğümler haricindeki tüm düğümler sıfır veya iki çocuğa sahip ise katı ikili ağaç olarak adlandırılır.**

Soru#3 (15 puan): "0, 1, 2, 3, 4, 5, 6, 7, 8, 9" **şeklinde numaralandırılmış on düğümden oluşan bir ikili ağaç, Inorder** *(Ortada kök)* **gezinildiği takdirde,** "9, 3, 1, 0, 4, 2, 7, 6, 8, 5" **ve Post-order** *(Sonra-kök)* **gezinildiği takdirde,** "9, 1, 4, 0, 3, 6, 7, 5, 8, 2" **sıralamalarını vermektedir. Buna göre, yukarıdaki özellikleri taşıyan ikili ağacı çiziniz.**



Preorder: Önce kök, sonra sol alt ağaç ve ardından sağ alt ağaç
Inorder: Önce sol alt ağaç, kök ve sağ alt ağaç
Postorder: Önce sol alt ağaç, sağ alt ağaç ve kök

Soru#4 (20 puan): Sondan başa ya da baştan sona okunduğunda aynı içeriği veren sözcük, kalıp ya da tümceler, **"*palindrom*"** olarak tanımlanmaktadır. Örneğin, boşluk ve noktalama işaretlerini göz ardı ettiğimizde, "*Race car*" bir palindromdur. Benzer şekilde, "*A man, a plan, a canal: Panama*" yine bir palindromdur. **Buna göre, girilen bir stringin palindrom olup olmadığını yığın***(stack)* **kullanarak belirleyiniz.** [Gerçekleştirimde, C#, Java, C vb. diller kullanılabilir. Temel yığın operasyonları kullanılarak çözüm, sözde kod ile algoritmik olarak da ifade edilebilir.]

- **Boşluk ve noktalama işaretleri dışında kalan "incelenmesi gereken karakter" sayısını *N* olarak belirle.**
- **Girdiler içerisinden ilk *N/2* karakteri yığına "push" operasyonu ile ekle.**
- **Eğer *N* değeri tek sayı ise (*N* ikiye tam bölünemiyorsa, ilgili karakteri) yığına dâhil etme.**
- **Sırasıyla yığından "pop" operasyonu ile çıkartılan her bir eleman ile girdiler arasındaki *N/2* karakteri karşılaştır.**
- **Tek olması nedeniyle yığına dâhil edilmeyen karakter dışında, eğer girdi seti ile yığından kaldırılan karakterler birbirleriyle eşleşmiyorsa, ilgili ifade "palindrom" değildir. Aksi takdirde, ilgili ifade palindromdur.**

```
import java.util.Stack;

public class PalindromeTest {

   public static void main(String[] args) {

      String input = "test";
      Stack<Character> stack = new Stack<Character>();

      for (int i = 0; i < input.length(); i++) {
         stack.push(input.charAt(i));
      }

      String reverseInput = "";

      while (!stack.isEmpty()) {
         reverseInput += stack.pop();
      }

      if (input.equals(reverseInput))
         System.out.println("Yes! that is a palindrome.");
      else
         System.out.println("No! that isn't a palindrome.");

   }
}
```

Soru#5 (10 puan): **"**89, 45, 68, 90, 29, 34, 17**" elemanlarını içeren bir diziyi, seçimli sıralama (***Selection Sort***) algoritması kullanarak sıralayınız.**

Seçimli Sıralama (Selection Sort)

- **Dizinin içerisindeki en küçük eleman bulunur.**
- **1. sıradaki elemanla yer değiştirilir.**
- **En küçük bulma işlemi dizinin ikinci elemanından başlanılarak tekrar edilir.**
- **Bulunan en küçük değer 2.sıradaki elemanla yer değiştirir.**
- **Bu işlem dizinin son elemanına kadar devam eder.**

89	45	68	90	29	34	**17**
17 |	45	68	90	**29**	34	89
17	29 |	68	90	45	**34**	89
17	29	34 |	90	**45**	68	89
17	29	34	45 |	90	**68**	89
17	29	34	45	68 |	90	**89**
17	29	34	45	68	89 |	90

Soru#6 (25 puan): Yığın veri yapısına ilişkin temel operasyonlar şu şekildedir:

- push(element): Yığına eleman ekleme.
- pop(): Yığının üstündeki elemanın çıkarılması.
- size(): Yığının boyutunun döndürülmesi.

Buna göre, yığın veri yapısına ilişkin yukarıda değinilen operasyonları, kuyruk veri yapısının temel operasyonlarını kullanarak gerçekleştiriniz. Her bir metot için, hesaplama karmaşıklığını belirleyiniz. [Gerçekleştirimde, C#, Java, C vb. diller kullanılabilir. Temel yığın operasyonları kullanılarak çözüm, sözde kod ile algoritmik olarak da ifade edilebilir.]

---

push(s, x)

1) x'i enque operasyonu kullanarak q1 kuyruğuna ekle.

pop(s)

1) **Kuyruğun son elemanı hariç geri kalan tüm elemanları q1 kuyruğundan q2 kuyruğuna enque ile ekle.**
2) **Deque(Q1) // Böylelikle, son eleman çıkarılmış oldu.**
3) **Q1 ve Q2 kuyruklarını birbirleriyle değiştir.**
4) **2. Adımda çıkarılan nesneyi döndür.**

```
class stack {
        Queue q, temp;

        Stack() {
                q = new Queue();
                temp = new Queue();
        }

        int size() {
                return (q.size());
        }

        void push(Object o) {
                q.enqueue (o);
        }

        Object pop() {
                int i, num_items;
                Object ret;
                num_items = q.size();

        for (i = 0; i < num_items - 1; i++)
                temp.enqueue(q.dequeue());
                ret = q.dequeue();
        for (i = 0; i < num_items-1; i++)
                q.enqueue(temp.dequeue());
                return ret;
        }
}
```

"Size" ve "push": *O(1)*
"Pop": *O(n)*