|  |  |
| --- | --- |
| **Adı Soyadı:**  **No:** | **VERİ YAPILARI**  **BİLGİSAYAR MÜHENDİSLİĞİ**  **11.01.2022** |
| **Sınav süresi 80 dakikadır.** Soruların tamamı cevaplanacaktır. Sınav süresi boyunca öğrenci kimliğinizi veya nüfus cüzdanınızı masanın üzerinde bulundurunuz. Cep telefonlarınızı kapatınız. Sınav sorumlularının talimatlarına uyunuz. Sınav başlangıcından itibaren ilk 15 dakikada sınavı terk etmeyiniz. | |

****

**SORULAR**

**BMÜ-221 Veri Yapıları Genel Sınav Soruları**

1. **(2p)a)** Aşağıda verilen postfix ifadenin değerini a=10, b=3 ve c=-2 için hesaplayınız?

|  |
| --- |
| aa+bc-\*= |

**b)** (**4p)** Aşağıdaki sayıları doğrusal sınama ve quadratik sınamaya göre Hash tablolarına yerleştiriniz?

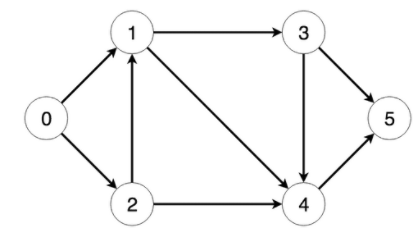
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a)Quadratik probing  12, 5, 19, 2, 23   |  |  | | --- | --- | | 0 | 23 | | 1 |  | | 2 | 19 | | 3 | 2 | | 4 |  | | 5 | 12 | | 6 | 5 | | b)Lineer probing  9, 18, 19, 27, 8, 17   |  |  | | --- | --- | | 0 | 18 | | 1 | 9 | | 2 | 19 | | 3 | 27 | | 4 | 17 | | 5 |  | | 6 |  | | 7 |  | | 8 | 8 | |

**c) (4p)** Elimizde **[** 38  5  4  1  9  -2**] şeklinde bir dizi bulunmaktadır. Dıştaki döngü 4 kez işletilene kadar Insertionsort ve Selectionsort’a göre dizinin adım adım değişimini yazınız?**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **i** | **Dizi** | | **[** 38  5  4  1  9  -2**]** | | **1** | [ 3 8 5 4 1 9 -2 ] | | **2** | [ 3 5 8 4 1 9 -2 ] | | **3** | [ 3 4 5 8 1 9 -2 ] | |  |  | | **4** | [ 1 3 4 5 8 9 -2 ] |   **Insertion sort(Eklemeli sıralama)** |
| |  |  | | --- | --- | | **i** | **Dizi** | | **[** 38  5  4  1  9  -2**]** | | **0** | [ -2 1 3 4 5 8 9 ] | | **1** | [ -2 1 3 4 5 8 9 ] | | **2** | [ -2 1 3 4 5 8 9 ] | | **3** | [ -2 1 3 4 5 8 9 ] |   **Selection sort(Seçmeli sıralama)** |

**d) (5p)** Aşağıdaki graf üzerinde BFS ve DFS yöntemine göre dolaşım esnasında yığıt/kuyruk’a eklenen ve çıkarılan elemanlar ekrana yazılmaktadır. Buna aşağıdaki grafta “0 “nolu düğümden dolaşılırsa ziyaret edilen düğümlerin sırasını veriniz. Örneğin 0 yığıta eklenmiş ise ekrana 0E çıkarılmış ise 0C yazılacaktır.

**NOT:** Gidilebilecek iki düğüm var ise daha küçük nolu düğüm ziyaret edilecektir.



|  |  |
| --- | --- |
| DFS | BFS |
| 0E  1E  3E  4E  5E  5C  4C  3C  1C  2E  2C  0C | 0E  1E  2E  0C  3E  4E  1C  2C  5E  3C  4C  5C |

1. **(10p)** Elinizde tam sayıları tutan iki adet sıralı bağlı liste olduğu varsayılmaktadır. Bu iki bağlı listeyi sıralı birleştirip listeyi döndüren bir public Liste birlestir(Liste L1, Liste L2){ } metodunu yazınız?

L1: 1🡪3🡪4🡪6🡪Null

L2:0🡪2🡪5🡪7🡪Null

L3:0🡪1🡪2🡪3🡪4🡪5🡪6🡪7🡪Null

Cozum : public void Liste\_birlestir( liste1 l1,liste2 l2 ) {

sayi gecici1 =l1.ilk ;

sayi gecici2 =l2.ilk ;

while (gecici1 != null && gecici2 != null) {

if (gecici1.no < gecici2.no) {

sona\_ekle(gecici1.no);

gecici1 = gecici1.sonraki;

} else{

sona\_ekle(gecici2.no);

gecici2 = gecici2.sonraki;

}

}

while (gecici1 != null) {

sona\_ekle(gecici1.no);

gecici1 = gecici1.sonraki;

}

while (gecici2 != null) {

sona\_ekle(gecici2.no);

gecici2 = gecici2.sonraki;

}

1. (**10p)** Kendisine parametre olarak bir dizi alan ve dizinin max-heap olup olmadığını döndüren boolean maxheapmi(int[] d){} metodunu yazınız?

public static boolean maxheapmi(int[] d){

boolean p= true ;

for (int i = 0; i < d.length/2-1; i++) {

if(p==true){

if(d[i]>d[(i\*2)+1]&&d[i]>d[(i\*2)+2]){

p=true;

}else {

p=false;

}

}

}

return p;}

min-heap olup olmadığını döndüren boolean maxheapmi(int[] d){} metodunu ise

public static boolean minheapmi(int[] d){

boolean p= true ;

for (int i = 0; i < d.length/2-1; i++) {

if(p==true){

if(d[i]<d[(i\*2)+1]&&d[i]<d[(i\*2)+2]){

p=true;

}else {p=false;}

}

}

return p;}

1. **(15p)** Elimizde yönsüz ve ağırlıksız bir grafı oluşturmak için sadece her düğümün derecesini tutan bir dizi bulunmaktadır. Bu diziyi kullanarak grafı temsil eden komşuluk matrisini oluşturunuz? int[][] kmatris(int [] derece) {} metodu yazılmalıdır.

Örneğin derece[]={2,2,1,1} ise komşuluk matrisi aşağıdaki gibi oluşur.

(0) (1) (2) (3)

(0) 0 1 1 0

(1) 1 0 0 1

(2) 1 0 0 0

(3) 0 1 0 0

Cozum:

public static int[][] komsuluk\_matrisini (int[] degree) {

int n = degree.length;

int[][] yeni = new int[n][n];

// düğümler arasında komşuluk olup olmadığını kontrol eden

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (degree[i] > 0 && degree[j] > 0) {

// i ve j düğümleri arasında bir komşuluk

yeni[i][j] = 1;

yeni[j][i] = 1;

// dereceleri azaltın

degree[i]--;

degree[j]--;

}

}

}

return yeni;

}

|  |
| --- |
| **a)(5p)**Aşağıda verilen graf için 0 düğümünden başlayarak Dijkstra’s algoritması ile en kısa yolu çiziniz ve toplam yol ağırlığını bulunuz. |
| Toplam Ağırlık: |
| **b) (5p)** Aşağıda verilen graf için Minimum Kapsayan Ağacı çiziniz ve çizdiğiniz ağacın toplam ağırlığını yazınız. |
| Toplam Ağırlık: |

1. **(10p)** 1 ile 100 arasından rastgele üretilen 20 adet sayıyı bir bağlı listeye ekleyen ve ekleme yaparken Rastgele üretilen sayı eğer bağlı listede var ise bu sayı bağlı listeye eklemeyerek yeni bir sayı üretilmesini sağlayan programı yazınız?

Cozum :

// Rastgele üretilen sayı listede varmi bir metod

public boolean varmi(int number){

sayi gecici=ilk;

while(gecici!=null){

if(gecici.no!=number ){

return false;

}gecici=gecici.sonraki;

}

return true;

}

// Rastgele üretilen sayı listeye ekleme

public void Liste\_olustur(){

int number = random.nextInt(100) + 1;

sona\_ekle(number);

int sayac=1;

for (int i = 1; sayac < 20; i++) {

number = random.nextInt(100) + 1;

if(!varmi(number)){

sona\_ekle(number);

sayac++;}

}

}

1. **(15p)** Bir bağlı yığıt veri yapısında bulunan elemanlardan verilen bir parametrenin değerine eşit olan tüm elemanları silerek yığıtın yeni halini döndüren hepsini\_sil() isimli bir metot yazınız. Örneğin { 9, 1, 9, 2, 3, 9, 4, 9, 9} elemanlarını içeren bir yığıt için 9 elemanı parametre olarak verilerek hepsini\_sil(9) metodu çağrıldığında {1,2,3,4} elemanlarını içeren bir yığıt geri döndürülecektir. Not: Bu metodu yazarken başka bir veri yapısı tanımlanmayacaktır(başka bir yığıt, bağlı liste veya kuyruk vb.).

**Başarılar Dilerim.**

**Doç. Dr. İlhan AYDIN**

**Dr. Öğr. Üyesi Gülşah KARADUMAN**