

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2020–2021 FALL SEMESTER**

**PROJECT-3 REPORT**

**(Search Tree, Heap, Hash Table, Bike Rental System)**

**DELIVERY DATE**

02/02/2021

**PREPARED BY**

05190000043, Ahmet Hakan Demirel

//İÇİNDEKİLERDE DEĞİŞİKLİK YAPAMADIM NOT OLARAK BELİRTMEK İSTEDİM.

İçindekiler

[1.a Durak nesnelerini oluşturma ve ağaca ekleme 2](#_Toc61463807)

[1.b Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi 2](#_Toc61463808)

[1.b.1 Kaynak Kod 2](#_Toc61463809)

[1.b.2 Ekran görüntüleri 2](#_Toc61463810)

[1.c Verilen müşteri ID’si için bilgi listeleme 2](#_Toc61463811)

[1.c.1 Kaynak Kod 2](#_Toc61463812)

[1.c.2 Ekran görüntüleri 2](#_Toc61463813)

[1.d Kiralama İşlemi 2](#_Toc61463814)

[2.a Hash Tablosuna Ekleme 2](#_Toc61463815)

[2.b Hash Tablosu Güncelleme 2](#_Toc61463816)

[3.a Heap Veri Yapısı Tasarlama 2](#_Toc61463817)

[3.a.1 Ön Çalışma 2](#_Toc61463818)

[3.a.2 Kaynak Kod 2](#_Toc61463819)

[3.b Max Heap düğüm yerleştirme 2](#_Toc61463820)

[3.c Heap bilgi çekme 2](#_Toc61463821)

[3.c.1 Kaynak Kod 2](#_Toc61463822)

[3.c.2 Ekran görüntüleri 3](#_Toc61463823)

[4.a Simple sorting algoritması 3](#_Toc61463824)

[4.b Advanced sorting algoritması 3](#_Toc61463825)

[4.c Sıralama algoritmalarının karşılaştırılması 3](#_Toc61463826)

[4.d Görselleştirme araçları 3](#_Toc61463827)

[Özdeğerlendirme Tablosu 4](#_Toc61463828)

ARAMA AĞAÇLARI, YIĞINLAR VE HASH TABLOSU: BİSİKLET KİRALAMA SİSTEMİ

Visual Studio 2019’da C# ile yazdım bütün kodu.

## 1.a Durak nesnelerini oluşturma ve ağaca ekleme

string[] duraklar = { "İnciraltı, 28, 2, 10", "Sahilevleri, 8, 1, 11", "Doğal Yaşam Parkı, 17, 1, 6", "Bostanlı İskele, 7, 0, 5", "Hasan Ağa Parkı, 5, 1, 8", "Karşıyaka Evlendirme Dairesi, 8, 1, 3", "Bayraklı, 10, 0, 5", "Buz Pisti, 11, 0, 9", "Bornova Metro, 5, 0, 7" };

DurakAgaci durakAgaci = new DurakAgaci();

int musteriSayisi = 20;

Random random = new Random();

for (int i = 0; i < duraklar.Length; i++)

{

string[] list = duraklar[i].Split(',');

DurakNesnesi durakNesnesi = new DurakNesnesi(list);

int durakMusteriSayisi = 0;

if (musteriSayisi > 0)

{

if (musteriSayisi > 10)

{

int duraktakiMusteri = random.Next(0, 11);

musteriSayisi -= duraktakiMusteri;

for (int j = 0; j < duraktakiMusteri; j++)

{

Musteri musteri = new Musteri();

durakNesnesi.list.Add(musteri);

}

}

else

{

int duraktakiMusteri = random.Next(0, musteriSayisi);

musteriSayisi -= duraktakiMusteri;

for (int j = 0; j < duraktakiMusteri; j++)

{

Musteri musteri = new Musteri();

durakNesnesi.list.Add(musteri);

}

}

}

musteriSayisi -= durakMusteriSayisi;

durakAgaci.insert(durakNesnesi);

## 1.b Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi

### 1.b.1 Kaynak Kod

public void preOrder(DurakNode localRoot)

{

if (localRoot != null)

{

localRoot.displayNode();

preOrder(localRoot.leftChild);

preOrder(localRoot.rightChild);

}

}

public int depth(DurakNode root)

{

if (root == null)

{

return 0;

}

else

{

int lDepth = depth(root.leftChild);

int rDepth = depth(root.rightChild);

if (lDepth > rDepth)

return (lDepth + 1);

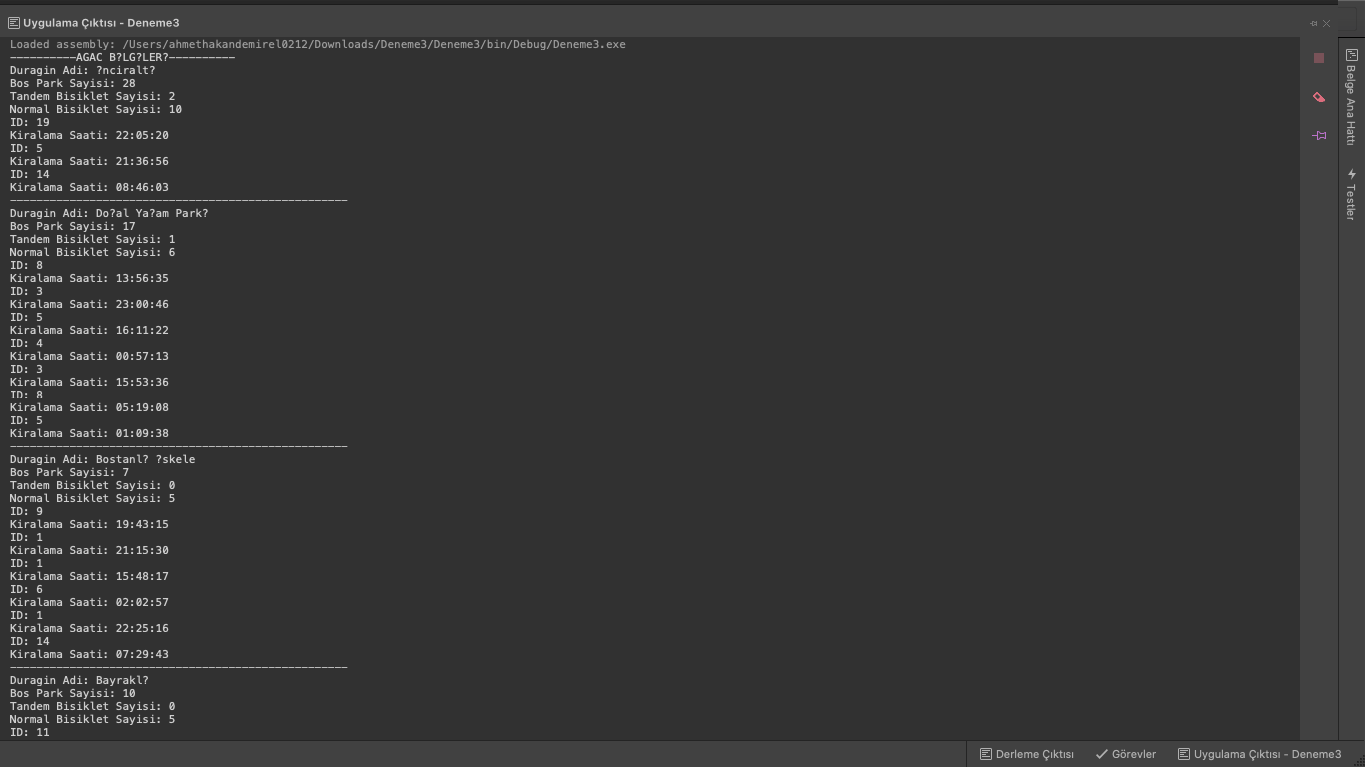
else

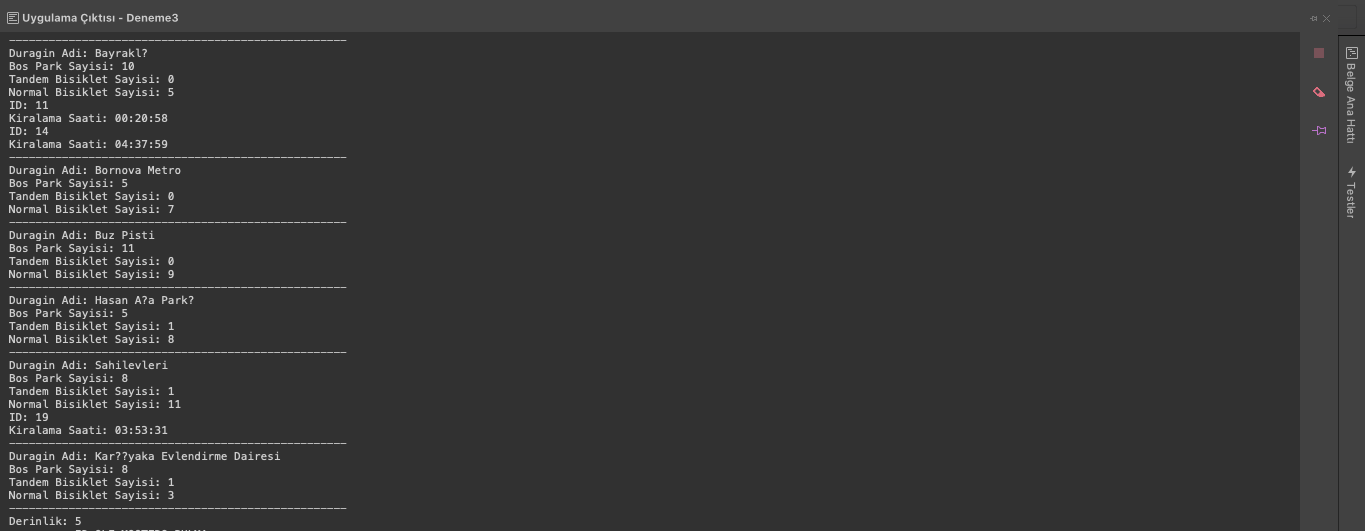
return (rDepth + 1);

}

}

### 1.b.2 Ekran görüntüleri





## 1.c Verilen müşteri ID’si için bilgi listeleme

### 1.c.1 Kaynak Kod

public DurakNesnesi preOrderID(DurakNode durakNode)

{

if (durakNode != null)

{

DurakNesnesi durakNesnesi = IDileBulma(durakNode);

if (durakNesnesi != null)

{

return durakNesnesi;

}

preOrderID(durakNode.leftChild);

preOrderID(durakNode.rightChild);

return null;

}

return null;

}

public DurakNesnesi IDileBulma(DurakNode root)

{

Console.WriteLine("İstediğiniz ID'yi giriniz(1-20): ");

string girilenID = Console.ReadLine();

DurakNode current = root;

foreach (Musteri musteri in current.data.list)

{

if (girilenID == musteri.ID)

{

return current.data;

}

}

return null;

}

### 1.c.2 Ekran görüntüleri

//1-c maddesi için üretilen konsol/ekran görüntüsünü buraya ekleyiniz

## 1.d Kiralama İşlemi

//1-c maddesi için yazmış olduğunuz kodları ve açıklamaları buraya ekleyiniz

## 2.a Hash Tablosuna Ekleme

public static Hashtable hashtableOlusturucu(string[] arr)

{

Hashtable hashtable = new Hashtable();

for (int i = 0; i < arr.Length; i++)

{

string[] list = arr[i].Split(',');

string bilgiler = "Boş Park Sayısı: " + list[1] + "," + "Tandem Bisiklet Sayısı :" + list[2] + "," + "Normal Bisiklet Sayısı" + list[3];

hashtable[list[0]] = bilgiler;

}

return hashtable;

}

## 2.b Hash Tablosu Güncelleme

public static Hashtable hashtableGuncelleme(string[] arr)

{

Hashtable hashtable = new Hashtable();

for (int i = 0; i < arr.Length; i++)

{

string[] list = arr[i].Split(',');

int BP = Convert.ToInt32(list[1]);

int NB = Convert.ToInt32(list[3]);

if (BP > 5)

{

NB += 5;

}

list[3] = NB.ToString();

string bilgiler = "Boş Park Sayısı: " + list[1] + "," + "Tandem Bisiklet Sayısı :" + list[2] + "," + "Normal Bisiklet Sayısı" + list[3];

hashtable[list[0]] = bilgiler;

}

return hashtable;

}

## 3.a Heap Veri Yapısı Tasarlama

### 3.a.1 Ön Çalışma

//3-a maddesi için kurşunkalemle A4 kağıda yazılmış açıklamalı kodun görüntüsünü ekleyiniz.

### 3.a.2 Kaynak Kod

//3-a maddesi için yazmış olduğunuz kodları ve açıklamaları buraya ekleyiniz

## 3.b Max Heap düğüm yerleştirme

//3-b maddesi için yazmış olduğunuz kodları ve açıklamaları buraya ekleyiniz

## 3.c Heap bilgi çekme

### 3.c.1 Kaynak Kod

//3-c maddesi için yazmış olduğunuz kodları ve açıklamaları buraya ekleyiniz

### 3.c.2 Ekran görüntüleri

//3-c maddesi için üretilen konsol/ekran görüntüsünü buraya ekleyiniz

## 4.a Simple sorting algoritması

class SelectionSort

{

private long[] arr;

private int nElems;

public SelectionSort(int max)

{

arr = new long[max];

nElems = 0;

}

public void inset(long value)

{

arr[nElems] = value;

nElems++;

}

public void display()

{

for (int i = 0; i < nElems; i++)

{

Console.Write(arr[i] + " ");

}

}

public void selectionsort()

{

int a, b, c;

for (a = 0; a < nElems-1 ; a++)

{

c = a;

for (b = a + 1; b < nElems; b++)

if (arr[b] < arr[c])

{

c = b;

}

swap(a, c);

}

}

public void swap(int a, int b)

{

long temp = arr[a];

arr[a] = arr[b];

arr[b] = temp;

}

}

## 4.b Advanced sorting algoritması

class ShellSort

{

private long[] arr;

private int nElems;

public ShellSort(int max)

{

arr = new long[max];

nElems = 0;

}

public void insert(long value)

{

arr[nElems] = value;

nElems++;

}

public void display()

{

for (int i = 0; i < nElems; i++)

{

Console.Write(arr[i] + " ");

}

}

public void shellsort()

{

int inner, outer;

long temp;

int h = 1;

while (h <= nElems / 3)

{

h = h \* 3 + 1;

}

while (h > 3)

{

for (outer = h; outer < nElems; outer++)

{

temp = arr[outer];

inner = outer;

while (inner > h - 1 && arr[inner - h] >= temp)

{

arr[inner] = arr[inner - h];

inner -= h;

}

arr[inner] = temp;

}

h = (h - 1) / 3;

}

}

}

## 4.c Sıralama algoritmalarının karşılaştırılması

Sıralama algoritmaları uygulanacak veri yapısına göre değişiklik gösterir. Örneğin, Quicksort çok hızlı bir algoritmadır ancak uygulanması nispeten karmaşıktır, bu nedenle küçük veri grupları için pratik olmayabilir. Kabarcık sıralama ise yavaş bir algoritmadır fakat uygulanması çok kolaydır. Bu sebeple sıralama algoritmalarının seçimi kullanılacağı yere göre ve beklentiye göre değişir.

## 4.d Görselleştirme araçları

Görselleştirmenin etkisi bence olumlu oldu. Çünkü hem nasıl çalıştıklarını görüyoruz hem de hızlarını daha iyi anlıyoruz. Görsel etkileşim araçları videolardan daha etkili. Çünkü var olan bir uygulama üstünde denemeler yapabiliyoruz.

# Özdeğerlendirme Tablosu

**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Durak nesnelerini oluşturma ve ağaca ekleme** | **10** | **10** | **Istenilen her şeyi eksiksiz yaptım.** |
| **1 b) Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi** | **10** | **10** | **Derinliği doğru hesaplayıp bilgileri yazdırdım** |
| **1 c) Verilen müşteri ID’si için bilgi listeleme** | **10** | **10** | **Istenilen ID ile bilgi listelemeyi yaptım.** |
| **1 d)** **Kiralama İşlemi** | **10** | **0** | **Kodu yazamadım.** |
| **2 a) Hash Tablosuna Ekleme** | **10** | **10** | **Istenildiği gibi Hashtable yaptım.** |
| **2 b) Hash Tablosu Güncelleme** | **5** | **5** | **Boş park sayısı beşten fazla olan duraklara 5 tane normal bisiklet eklettim.** |
| **3 a) Heap Veri Yapısı Tasarlama** | **5** | **0** | **Yapamadım** |
| **3 b) Max Heap düğüm yerleştirme** | **5** | **0** | **Yapamadım** |
| **3 c) Heap bilgi çekme** | **5** | **0** | **Yapamadım** |
| **4 a) Simple sorting algoritması** | **4** | **4** | **Selection sort yaptım** |
| **4 b) Advanced sorting algoritması** | **4** | **4** | **Shell sort yaptım** |
| **4 c) Sıralama algoritmalarının karşılaştırılması** | **4** | **4** | **Karşılaştırdım** |
| **4 d) Görselleştirme araçları** | **8** | **8** | **Ikisini de kullanıp değerlendirdim** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **Tabloyu dürüstçe doldurdum.** |
| **Toplam** | **100** | **75** | **Eksiklerim var.** |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı (ve nelerin yapılmadığı / yapılamadığı) yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**