

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2021–2022 FALL SEMESTER**

**PROJECT-3 REPORT**

**(Search Tree, Hash Table, Heap, Food Delivery Company)**

**DELIVERY DATE**

15/01/2022

**PREPARED BY**

05200000742 , Mehmet ÖKTEN

05180000063 , Burak KIZILAY

İçindekiler

[1.a YemekSınıfı nesnesinin ve Yemek Sipariş Ağacının oluşturulması 3](#_Toc92533878)

[1.b Ağaç derinliği ve ağaçtaki bilgilerin ekrana listelenmesi 3](#_Toc92533879)

[1.b.1 Kaynak Kod 3](#_Toc92533880)

[1.b.2 Ekran görüntüleri 3](#_Toc92533881)

[1.c 150 TL üstündeki siparişlerin bilgilerini listeleme 3](#_Toc92533882)

[1.c.1 Kaynak Kod 3](#_Toc92533883)

[1.c.2 Ekran görüntüleri 3](#_Toc92533884)

[1.d Bir yiyecek/içeceğin tüm ağaçta kaç adet sipariş verildiğini döndürme ve fiyat güncelleme 3](#_Toc92533885)

[1.d.1 Kaynak Kod 3](#_Toc92533886)

[1.d.2 Ekran görüntüleri 3](#_Toc92533887)

[2.a Hash Tablosuna Ekleme 3](#_Toc92533888)

[2.b Hash Tablosu Güncelleme 3](#_Toc92533889)

[3.a Heap Veri Yapısı Tasarlama 3](#_Toc92533890)

[3.b Max Heap düğüm yerleştirme 3](#_Toc92533891)

[3.c Heap bilgi çekme 4](#_Toc92533892)

[3.c.1 Kaynak Kod 4](#_Toc92533893)

[3.c.2 Ekran görüntüleri 4](#_Toc92533894)

[4.a Simple sorting veya Advanced sorting algoritması 4](#_Toc92533895)

[4.b Sıralama algoritması ile karşılaştırılması 4](#_Toc92533896)

[4.c Görselleştirme araçları 4](#_Toc92533897)

[Özdeğerlendirme Tablosu 5](#_Toc92533898)

ARAMA AĞAÇLARI, YIĞINLAR VE HASH TABLOSU: YEMEK DAĞITIM ŞİRKETİ

We used VISUAL STUDIO 2019 and C# language for this project.

## 1.a YemekSınıfı nesnesinin ve Yemek Sipariş Ağacının oluşturulması

class YemekSiparişAğacı

{

public Mahalle kök;

private int derinlik = 0;

public YemekSiparişAğacı()

{

kök = null;

}

public void ekleme(String mahalleAdı)

{

Mahalle eklenecekMahalle = new Mahalle(mahalleAdı);

if (kök == null)

{

kök = eklenecekMahalle;

}

else

{

Mahalle şuAnki = kök;

Mahalle biÜstü;

while (true)

{

biÜstü = şuAnki;

if (string.Compare(mahalleAdı, şuAnki.adı) == 1)

{

şuAnki = şuAnki.rightChild;

if (şuAnki == null)

{

biÜstü.rightChild = eklenecekMahalle;

return;

}

}

else

{

şuAnki = şuAnki.leftChild;

if (şuAnki == null)

{

biÜstü.leftChild = eklenecekMahalle;

return;

}

}

}

}

}

public void inOrderYazdırma()

{

inOrderYazdır(kök);

}

private void inOrderYazdır(Mahalle mahalle)

{

if (mahalle != null)

{

inOrderYazdır(mahalle.leftChild);

mahalle.Yazdır();

inOrderYazdır(mahalle.rightChild);

}

}

public void DerinliğiGüncelle(Mahalle mahalle, int derinliği)

{

if (derinliği > derinlik && mahalle != null)

{

derinlik = derinliği;

}

if (mahalle != null)

{

DerinliğiGüncelle(mahalle.leftChild, derinliği + 1);

DerinliğiGüncelle(mahalle.rightChild, derinliği + 1);

}

}

public int GetDerinlik()

{

DerinliğiGüncelle(kök, 0);

return derinlik;

}

public Mahalle AdıVerilenMahalleyiDöndür(Mahalle mahalle, string bulunacakMahalleninAdı)

{

if (mahalle != null && string.Compare(mahalle.adı, bulunacakMahalleninAdı) == 0)

{

return mahalle;

}

else

{

if (mahalle != null)

{

AdıVerilenMahalleyiDöndür(mahalle.leftChild, bulunacakMahalleninAdı);

AdıVerilenMahalleyiDöndür(mahalle.rightChild, bulunacakMahalleninAdı);

}

}

return null;

}

public void yüzElliLirayıGeçeniYazdır(string mahalleAdı)

{

Console.WriteLine("\n" + mahalleAdı + " mahallesinde 150 lirayı geçen siparişler aşağıda listelenecektir\n\n");

for (int i = 0; i < AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi.Count; i++)

{

if (AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi[i].yüzElliLirayıGeçiyorMu())

{

AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi[i].Yazdır();

}

}

}

public int KaçAdetSöylendiyseDöndürVeYüzdeOnİndirimYap(string ürünAdı)

{

return kök.siparişlerListesi[0].siparişbilgileri[0].KaçAdetSöylendiSonrasındaYüzdeOnİndirimYap(ürünAdı);

}

## }

public class TekBirUrun

{

public string adı;

public int adedi;

public int adetUcretininSırası;

private static string[] yemeklerVeİçecekler = { "Menemen", "Mantı", "Kebap", "Ayran", "Tavuk Döner", "Et Döner", "Ekmek Arası Ekmek Döner", "Sandviç", "Coca Cola", "Fanta", "Beyti", "Soğuk Çay" };

public static double[] fiyatlar = { 10, 11, 20, 5, 8, 12, 6, 8, 7, 7, 20, 6 };

public static int[] kaçAdetOldu = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

public TekBirUrun()

{

Random rastgele = new Random();

int sayi = rastgele.Next(1, 9);

adedi = sayi;

sayi = rastgele.Next(0, 12);

adı = yemeklerVeİçecekler[sayi];

kaçAdetOldu[sayi] = kaçAdetOldu[sayi] + adedi;

adetUcretininSırası = sayi;

}

public string ToString()

{

return "Adı: " + adı + " Adedi: " + adedi + " Toplam fiyatı: " + Fiyatı(); ;

}

public double Fiyatı()

{

return adedi \* fiyatlar[adetUcretininSırası];

}

private void yuzde10indirim(int i)

{

fiyatlar[i] = (9 \* fiyatlar[i]) / 10;

}

public int KaçAdetSöylendiSonrasındaYüzdeOnİndirimYap(string yemekYadaİçecek)

{

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

{

if (yemeklerVeİçecekler[i] == yemekYadaİçecek)

{

yuzde10indirim(i);

return kaçAdetOldu[i];

}

}

return 0;

}

}

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

### 1.b.1 Kaynak Kod

class Program

{

static void Main(string[] args)

{

string[] mahalleAdları = { "Evka 3", "Özkanlar", "Atatürk", "Erzene", "Kazımdirik" };

YemekSiparişAğacı yemekSiparişAğacı = new YemekSiparişAğacı();

yemekSiparişAğacı.ekleme(mahalleAdları[0]);

yemekSiparişAğacı.ekleme(mahalleAdları[1]);

yemekSiparişAğacı.ekleme(mahalleAdları[2]);

yemekSiparişAğacı.ekleme(mahalleAdları[3]);

yemekSiparişAğacı.ekleme(mahalleAdları[4]);

yemekSiparişAğacı.inOrderYazdırma();

Console.WriteLine("Ağacımızın derinliği: " + yemekSiparişAğacı.GetDerinlik());

yemekSiparişAğacı.yüzElliLirayıGeçeniYazdır("Evka 3");

Console.WriteLine("\n Menemen kaç adet söylendi: " + yemekSiparişAğacı.KaçAdetSöylendiyseDöndürVeYüzdeOnİndirimYap("Menemen"));

Console.WriteLine(" indirimden sonraki fiyatlar: \n");

yemekSiparişAğacı.inOrderYazdırma();

}

### }

class YemekSiparişAğacı

{

public Mahalle kök;

private int derinlik = 0;

public YemekSiparişAğacı()

{

kök = null;

}

public void DerinliğiGüncelle(Mahalle mahalle, int derinliği)

{

if (derinliği > derinlik && mahalle != null)

{

derinlik = derinliği;

}

if (mahalle != null)

{

DerinliğiGüncelle(mahalle.leftChild, derinliği + 1);

DerinliğiGüncelle(mahalle.rightChild, derinliği + 1);

}

}

public int GetDerinlik()

{

DerinliğiGüncelle(kök, 0);

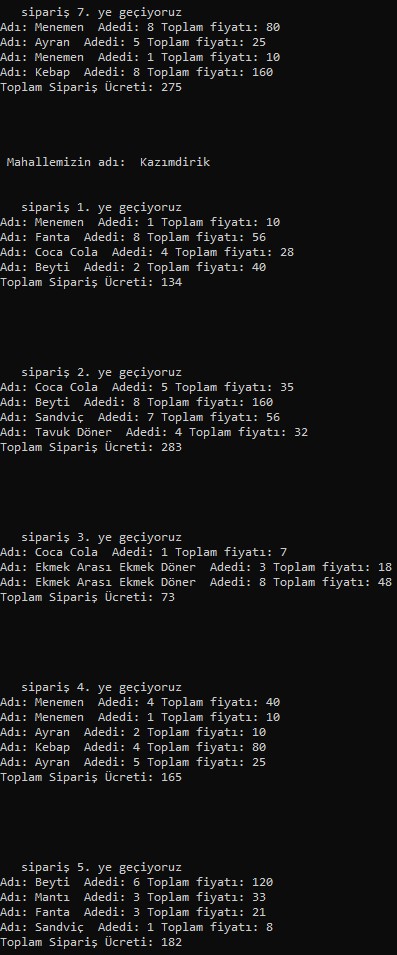
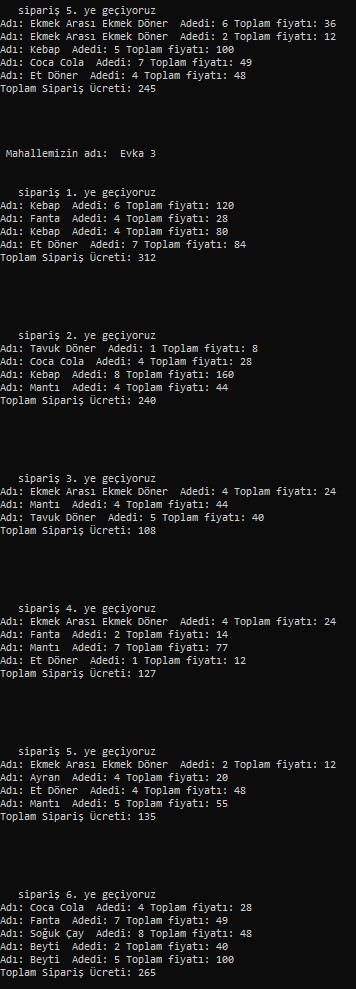
return derinlik;

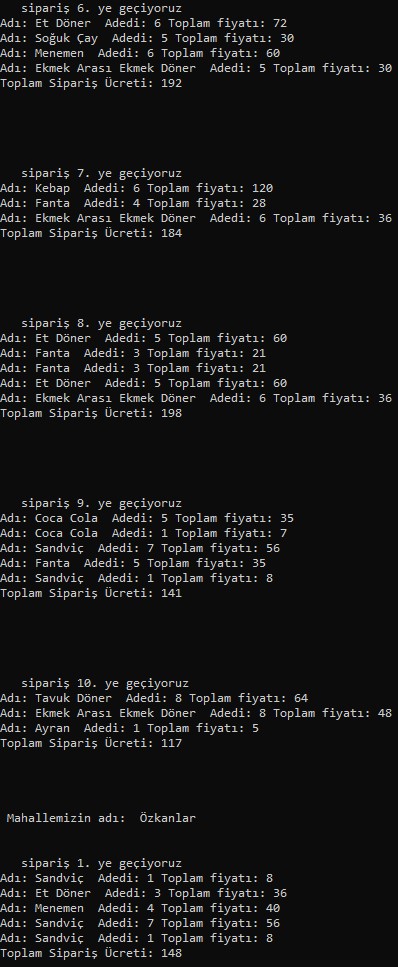
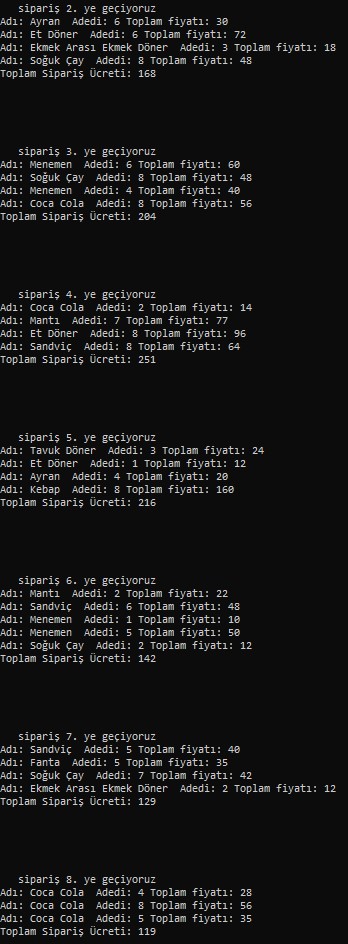
}

}

### 1.b.2 Ekran görüntüleri

## 



## 1.c 150 TL üstündeki siparişlerin bilgilerini listeleme

### 1.c.1 Kaynak Kod

yemekSiparişAğacı.yüzElliLirayıGeçeniYazdır("Evka 3");

public void yüzElliLirayıGeçeniYazdır(string mahalleAdı)

{

Console.WriteLine("\n" + mahalleAdı + " mahallesinde 150 lirayı geçen siparişler aşağıda listelenecektir\n\n");

for (int i = 0; i < AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi.Count; i++)

{

if (AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi[i].yüzElliLirayıGeçiyorMu())

{

AdıVerilenMahalleyiDöndür(kök, mahalleAdı).siparişlerListesi[i].Yazdır();

}

}

### }

### 1.c.2 Ekran görüntüleri

## 

## 1.d Bir yiyecek/içeceğin tüm ağaçta kaç adet sipariş verildiğini döndürme ve fiyat güncelleme

### 1.d.1 Kaynak Kod

Console.WriteLine("\n Menemen kaç adet söylendi: " + yemekSiparişAğacı.KaçAdetSöylendiyseDöndürVeYüzdeOnİndirimYap("Menemen"));

Console.WriteLine(" indirimden sonraki fiyatlar: \n");

class YemekSiparişAğacı

{

public int KaçAdetSöylendiyseDöndürVeYüzdeOnİndirimYap(string ürünAdı)

{

return kök.siparişlerListesi[0].siparişbilgileri[0].KaçAdetSöylendiSonrasındaYüzdeOnİndirimYap(ürünAdı);

}

}

public class TekBirUrun

{

public string adı;

public int adedi;

public int adetUcretininSırası;

private static string[] yemeklerVeİçecekler = { "Menemen", "Mantı", "Kebap", "Ayran", "Tavuk Döner", "Et Döner", "Ekmek Arası Ekmek Döner", "Sandviç", "Coca Cola", "Fanta", "Beyti", "Soğuk Çay" };

public static double[] fiyatlar = { 10, 11, 20, 5, 8, 12, 6, 8, 7, 7, 20, 6 };

public static int[] kaçAdetOldu = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

public TekBirUrun()

{

Random rastgele = new Random();

int sayi = rastgele.Next(1, 9);

adedi = sayi;

sayi = rastgele.Next(0, 12);

adı = yemeklerVeİçecekler[sayi];

kaçAdetOldu[sayi] = kaçAdetOldu[sayi] + adedi;

adetUcretininSırası = sayi;

}

public string ToString()

{

return "Adı: " + adı + " Adedi: " + adedi + " Toplam fiyatı: " + Fiyatı(); ;

}

public double Fiyatı()

{

return adedi \* fiyatlar[adetUcretininSırası];

}

private void yuzde10indirim(int i)

{

fiyatlar[i] = (9 \* fiyatlar[i]) / 10;

}

public int KaçAdetSöylendiSonrasındaYüzdeOnİndirimYap(string yemekYadaİçecek)

{

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

{

if (yemeklerVeİçecekler[i] == yemekYadaİçecek)

{

yuzde10indirim(i);

return kaçAdetOldu[i];

}

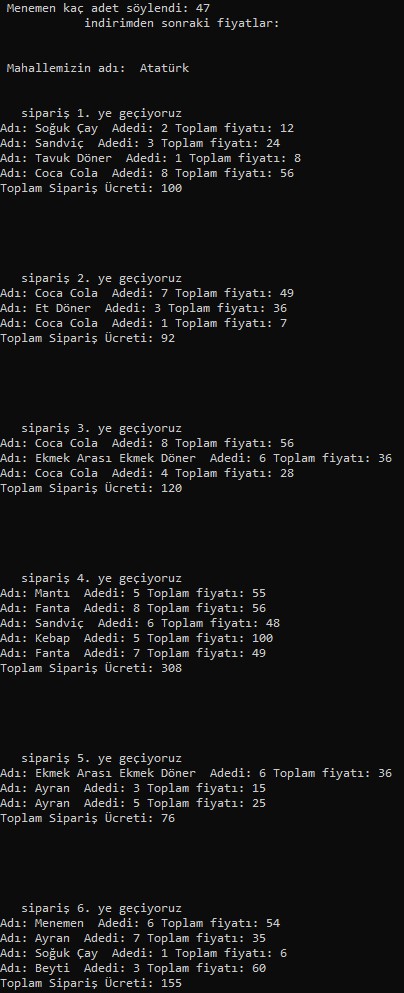
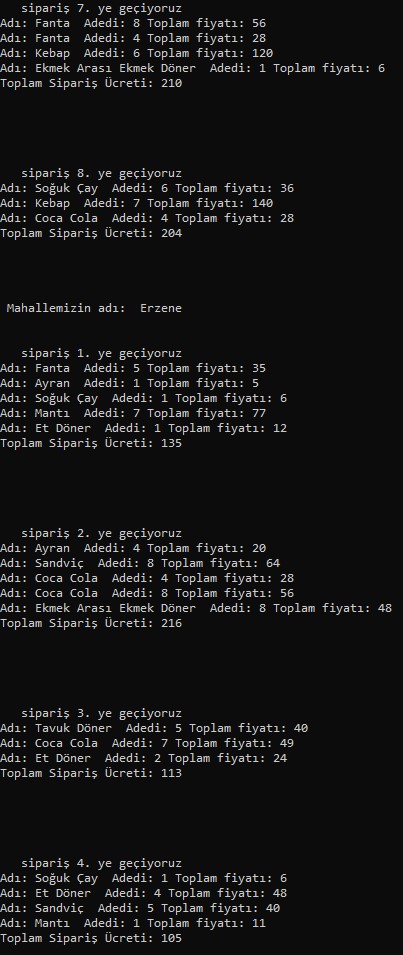
}

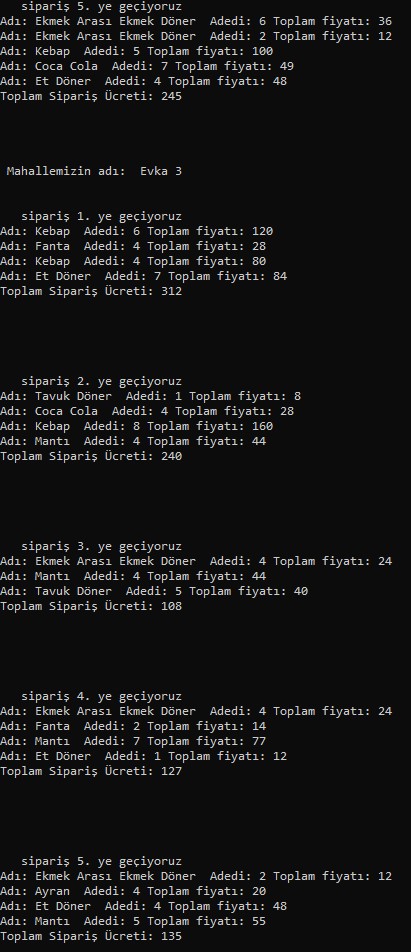
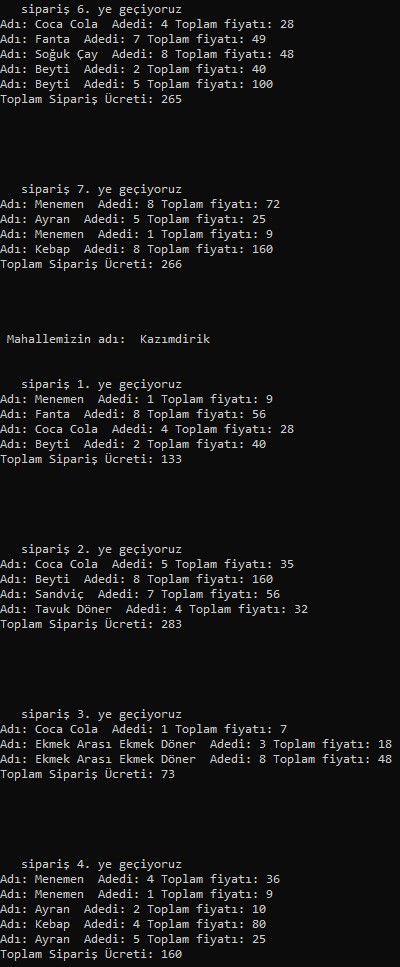
return 0;

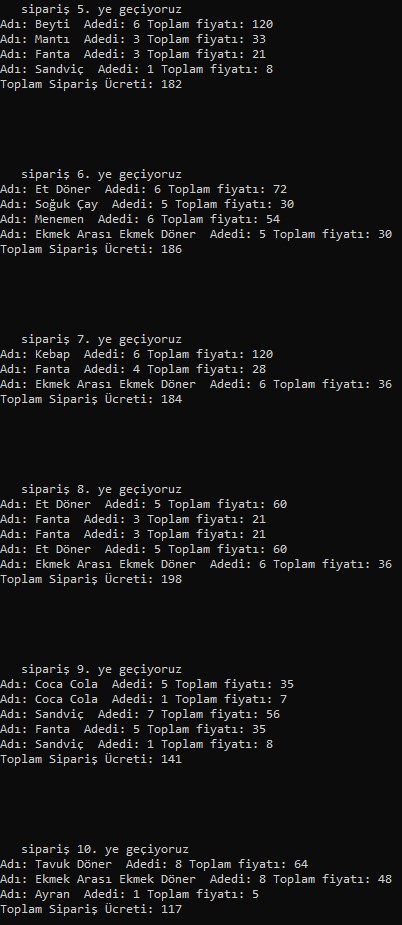
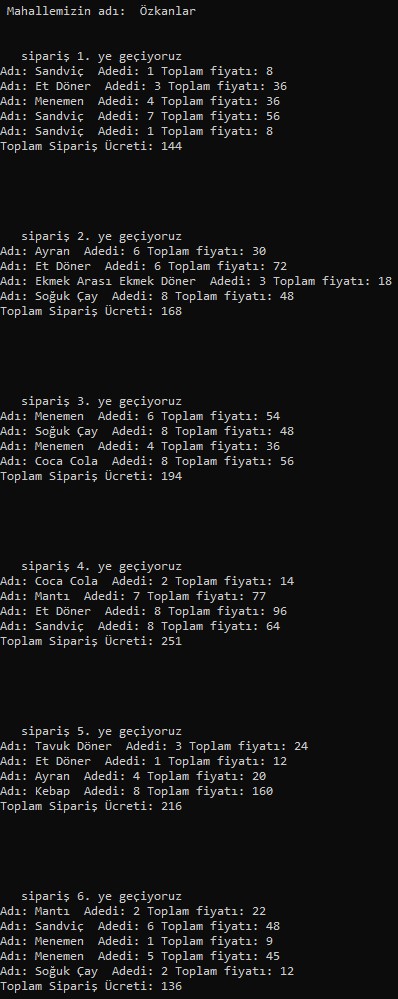
}

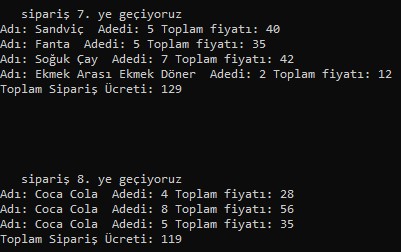
### }

### 1.d.2 Ekran görüntüleri



## 

## 2.a Hash Tablosuna Ekleme

class Program

{

static void Main(string[] args)

{

Hashtable MahallelerVeNufusları = new Hashtable();

string[] mahalleAdları = { "MERİÇ", "KOŞUKAVAK", "TUNA", "BİRLİK", "SERİNTEPE", "ÇAMKULE", "MERKEZ", "YILDIRIM BEYAZIT", "ÜMİT", "ÇINAR" };

MahallelerVeNufusları.Add(mahalleAdları[0], 8394);

MahallelerVeNufusları.Add(mahalleAdları[1], 7916);

MahallelerVeNufusları.Add(mahalleAdları[2], 7607);

MahallelerVeNufusları.Add(mahalleAdları[3], 6949);

MahallelerVeNufusları.Add(mahalleAdları[4], 6905);

MahallelerVeNufusları.Add(mahalleAdları[5], 6709);

MahallelerVeNufusları.Add(mahalleAdları[6], 6341);

MahallelerVeNufusları.Add(mahalleAdları[7], 6192);

MahallelerVeNufusları.Add(mahalleAdları[8], 4593);

MahallelerVeNufusları.Add(mahalleAdları[9], 4287);

Console.WriteLine(" Önce olduğu gibi yazdırıyoruz: ");

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

{

Console.WriteLine("Mahallenin Adı: " + mahalleAdları[i] + "\n" +

"Mahallenin Nüfusu: " + MahallelerVeNufusları[mahalleAdları[i]] + "\n");

}

## 

## 2.b Hash Tablosu Güncelleme

string verilenBaşHarf = "M"; //Şimdi verilen harfle başlayan keylerin valuesine 1 eklenecek.

for (int i = 0; i < 10; i++)

{

string şuAnkiMahalle = mahalleAdları[i];

if (şuAnkiMahalle.StartsWith(verilenBaşHarf))

{

int a = (int)MahallelerVeNufusları[mahalleAdları[i]];

a = a + 1;

MahallelerVeNufusları[mahalleAdları[i]] = a;

}

} //Eklendi, şimdi yazdırılacak.

Console.WriteLine(" Sonra ise " + verilenBaşHarf + " baş harfine sahiplerin nüfusunu 1 arttırıp yazdırıyoruz: ");

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

{

Console.WriteLine("Mahallenin Adı: " + (string)mahalleAdları[i] + "\n" +

"Mahallenin Nüfusu: " + MahallelerVeNufusları[mahalleAdları[i]] + "\n\n");

}

## 

## 3.a Heap Veri Yapısı Tasarlama

class Program

{

static void Main(string[] args)

{

Mahalle mahalle1 = new Mahalle("ERZENE", 35135);

Mahalle mahalle2 = new Mahalle("KAZIMDİRİK", 33934);

Mahalle mahalle3 = new Mahalle("YEŞİLOVA", 33934);

Mahalle mahalle4 = new Mahalle("ATATÜRK", 28912);

Mahalle mahalle5 = new Mahalle("İNÖNÜ", 25778);

Mahalle mahalle6 = new Mahalle("MEVLANA", 25492);

Mahalle mahalle7 = new Mahalle("DOĞANLAR", 21461);

Mahalle mahalle8 = new Mahalle("EVKA 3", 20445);

Mahalle mahalle9 = new Mahalle("RAFET PAŞA", 19476);

Mahalle mahalle10 = new Mahalle("KIZILAY", 15795);

Heap maxHeapımız = Heap.GetHeap();

maxHeapımız.Ekle(mahalle1);

maxHeapımız.Ekle(mahalle2);

maxHeapımız.Ekle(mahalle3);

maxHeapımız.Ekle(mahalle4);

maxHeapımız.Ekle(mahalle5);

maxHeapımız.Ekle(mahalle6);

maxHeapımız.Ekle(mahalle7);

maxHeapımız.Ekle(mahalle8);

maxHeapımız.Ekle(mahalle9);

maxHeapımız.Ekle(mahalle10);

}

## }

## 3.b Max Heap düğüm yerleştirme

public class Heap //Max heap yazdık direkt olarak.

{

List<Mahalle> listem = new List<Mahalle>();

static Heap a;

private Heap() //Burada tek nesne olmasını istediğimiz için böyle bir yol izledik. Tek bir heap bu ödevde yetiyor.

{

}

public static Heap GetHeap()

{

if (a == null)

{

a = new Heap();

return a;

}

else

{

return a;

}

}

public void Ekle(Mahalle mahalle)

{

if (listem.Count == 0)

{

listem.Add(new Mahalle("abc", 123));//0. elemanı hiç kullanmayacağız. İlk elemanı boş bırakmak kolaylık sağlıyor.

}

listem.Add(mahalle);

Console.WriteLine("eklendi");

YeriniDüzelt(listem.Count);

}

public void Sil()

{

int sıra = 1;

listem[sıra] = listem[listem.Count];

listem.RemoveAt(listem.Count);

Console.WriteLine("max heap kuralına göre kökteki eleman silindi");

YeriniDüzelt(sıra);

}

public void Yazdır()

{

for (int i = 1; i < listem.Count; i++)

{

Console.WriteLine(i + ". eleman=> " + listem[i].ToString());

}

}

public void EnYüksek3()

{

for (int i = 1; i < 4; i++)

{

Console.WriteLine(i + ". eleman=> " + listem[i].ToString());

}

}

public void YeriniDüzelt(int sıra)

{

if (listem.Count > 2) //İlk elemanı boş eleman olarak biz ekleyeceğiz. Listenin 1 den başlaması kolaylık sağlıyor.

{

int atasi;

if (sıra / 2 == 1)

{

atasi = (listem.Count - 1) / 2;

}

else

{

atasi = (listem.Count) / 2;

}

if (listem.Count < atasi && listem[sıra].nüfusu > listem[atasi].nüfusu)

{

Mahalle geçici = listem[atasi];

listem[atasi] = listem[sıra];

listem[sıra] = geçici;

YeriniDüzelt(atasi);

}

if (listem.Count > sıra \* 2)

{

if (listem[sıra \* 2].nüfusu > listem[sıra].nüfusu)

{

Mahalle geçici = listem[sıra \* 2];

listem[sıra \* 2] = listem[sıra];

listem[sıra] = geçici;

YeriniDüzelt(sıra \* 2);

}

}

}

}

## }

## 3.c Heap bilgi çekme

### 3.c.1 Kaynak Kod

Console.WriteLine(" Şimdi eklediğimiz tüm mahalleleri sırasıyla yazdıracağız");

maxHeapımız.Yazdır();

Console.WriteLine("\n Şimdi ise ödevde istenildiği üzere en yüksek nüfuslu 3 mahalleyi çekiyoruz");

maxHeapımız.EnYüksek3();

public void EnYüksek3()

{

for (int i = 1; i < 4; i++)

{

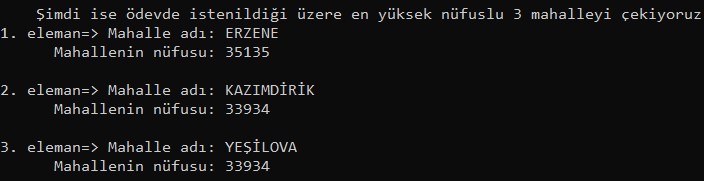
Console.WriteLine(i + ". eleman=> " + listem[i].ToString());

}

}

### 3.c.2 Ekran görüntüleri

## 



## 4.a Simple sorting veya Advanced sorting algoritması

using System;

using System.Collections.Generic;

namespace data\_proje\_3\_part\_4

{

class Program

{

static void Main(string[] args)

{

List<int> sayilar = new List<int>();

sayilar.Add(17);

sayilar.Add(15);

sayilar.Add(20);

sayilar.Add(10);

sayilar.Add(6);

sayilar.Add(9);

sayilar.Add(25);

BubbleSort(sayilar);

for (int i = 0; i < sayilar.Count; i++)

{

Console.WriteLine(sayilar[i]);

}

}

public static void BubbleSort(List<int> liste)

{ //Küçükten büyüğe sıralayan bubblesort, en büyüğü sola taşıyor. Her while a girdiğinde listenin en sonuna kesin olarak en büyüğü taşıyor bu yüzden.

if (liste.Count > 1) // Sıralanması gereken kısmı 1 azaltarak hız kazandırdık ama yine de bu yazmasının kolaylığı haricinde; insertion sort veya quick sort olsun, bunlara karşı bir avantaj sağlamıyor.

{ //Mesela quick sortun yazması zordur ama yüksek elemanlı (+30 elemanlı) veri kümelerinde bubble sorta göre baya zaman kazandırır, ya da insertion sort az karşılaştırma yapar.

bool islemYapıldıMı = true;// Zaman karmaşıklığı ilk turda n-1 karşılaştırma sonrasında n-2, n-3 ... tarzı gittiği için n^2 kare gelir.

int sıralanmasıGerekenKısım = liste.Count;

while (islemYapıldıMı)

{

islemYapıldıMı = false;

int arka;

int şuAnKi;

int tutucu;

for (int i = 1; i < sıralanmasıGerekenKısım; i++)

{

arka = liste[i - 1];

şuAnKi = liste[i];

if (arka > şuAnKi)

{

liste[i] = arka;

liste[i - 1] = şuAnKi;

islemYapıldıMı = true;

}

}

sıralanmasıGerekenKısım = sıralanmasıGerekenKısım - 1;

}

}

}

}

}

## 4.b Sıralama algoritması ile karşılaştırılması

## QuickSort çok hızlı bir algoritmadır ancak uygulanması nispeten karmaşıktır, bu nedenle küçük veri grupları için pratik olmayabilir. BubbleSort ise yavaş bir algoritmadır fakat uygulanması çok kolaydır.

## Sıralama algoritmalarının çalışma hızları, sıralanacak verinin büyüklüğü, kısmen sıralı olması, tersten sıralı olması veya tümüyle karışık yapıda olmasına bağlı olarak farklılık göstermektedir. Seçim yapılırken verinin bu anlamda değerlendirilmesi ve uygun algoritmanın tercihi en iyi sonuç için faydalı olacaktır.

## 4.c Görselleştirme araçları

Görselleştirme araçlarının en büyük faydası konuyu kavrama konusunda büyük bir etkisi oluyor.

Genelde yaptığımız şeyler abstract olarak kaldığından dolayı ortaya bir şey koyamıyoruz hissi oluşabiliyor ya da ne yaptığımızı tam kavrayamayabiliyoruz. Ama şöyle görsel metaryelleri izledikçe ufkumuzun daha da açıldığını hissediyoruz. Aynı zamanda konuyu daha doğru bir mantıkta anlamamızı sağlıyor.

Örneğin bu sorting algoritmalarında konuyu kavramamızda gerçekten çok büyük yardımcı olduğunu söyleyebiliriz. Hem de diğer sorting algoritmalarında hangisinin hangi konuda daha avantajlı olduğunu görmemizde daha net bir bilgiye ulaşmamızı sağlamış oluyor.

# 

# Özdeğerlendirme Tablosu

**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 3 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| **1 a) Ağaç (Yemek Sipariş Ağacı Oluşturma)** | **10** | **10** | **Yapıldı** |
| **1 b) Derinlik Bulma, Ağacı Listeleme** | **10** | **10** | **Yapıldı** |
| **1 c) Arama ve Listeleme** | **10** | **10** | **Yapıldı** |
| **1 d) Ağaçta bir yemek türüne indirim yapma** | **10** | **10** | **Yapıldı** |
| **2) Hash Tablosu** | **15** | **15** | **Yapıldı** |
| **3) Yığın Ağacı (Heap)** | **15** | **15** | **Yapıldı** |
| **4) Sıralama Algoritmaları** | **20** | **20** | **Yapıldı** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **Yapıldı** |
| **Toplam** | **100** | **100** | **Projeyi tam bir şekilde gerçekleştirdiğimizi düşünüyoruz** |

**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.**