

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2021–2022 FALL SEMESTER**

**PROJECT-3 REPORT**

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

**DELIVERY DATE**

15/01/2022

**PREPARED BY**

05200000091, Mehmet Akif Çelebi

05100000046, Ramazan Özdemir

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

Visual Studio 2019 / C#

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

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Proje3

{

// Bu nesne oluşturulduğu sırada(new yapıldığında yani) yemek, yemeğin fiyatı, adeti ve tutarı hesaplanmış bir şekilde hazır oluşuyor.

internal class Siparis

{

public string yemek\_adi;

int yemek\_adedi;

int yemek\_fiyati;

public int odenecek\_tutar;

ArrayList siparis= new ArrayList();

public static Random random = new Random();

public static ArrayList rastegele\_yemek()

{

int yemek\_index = random.Next(0, 30); // yemek seçimi için rastgele index üretiyor

int yemek\_count = random.Next(1, 9); // rastgele yemek adedi üretiyor

string[] yemek\_listesi = { "Lahmacun", "Çiğ\_Köfte", "Dolma", "Kebap", "Ciğer" ,

"Çorba","Kelle Paça","Pizza","Poğaça","Makarna","Simit","Kizartma","Türlü","Pilav",

"Tavuk sote","Omlet","Haşlanmiş yumurta","Ezo gelin çorba","Tavuk döner","Et döner","Mücver","Kokoreç",

"Börek","Karniyarik","Salata","Tarhana çorbası","Sarma","Taze fasulye","Kuru fasulye","Kabak yemegi"};

int[] yemek\_fiyat = { 15, 10, 15, 17, 22, 15, 23, 15, 8, 19, 5, 20, 15, 18, 21, 20, 15, 20, 15, 17, 15, 18, 15, 20, 15, 17, 20, 18, 20, 15 };

ArrayList cikti = new ArrayList(); // değerleri constructorda atamak için oluşturduğum dizi

cikti.Add(yemek\_listesi[yemek\_index]);

cikti.Add(yemek\_count);

cikti.Add(yemek\_fiyat[yemek\_index]);

cikti.Add(yemek\_fiyat[yemek\_index] \* yemek\_count);

return cikti;

}

public Siparis()

{

this.siparis = rastegele\_yemek();

this.yemek\_adi = (string)this.siparis[0];

this.yemek\_adedi = (int)this.siparis[1];

this.yemek\_fiyati=(int)this.siparis[2];

this.odenecek\_tutar=(int)this.siparis[3];

}

public override string ToString()

{

return this.yemek\_adi + "; |Fiyat:" + this.yemek\_fiyati + "|" + "Adet:" + this.yemek\_adedi+" Tutar-->"+odenecek\_tutar;

}

}

internal class Siparisler

{

public List<Siparis> siparisleri = new List<Siparis>(); // Sipariş nesnelerini tutacak olan liste

public ArrayList tum\_siparisler = new ArrayList(); // siparişler nesnesini tutacak olan dizi

public int toplam\_tutar; // siparişin toplam tutarını tutacak olan değişken

}

internal class Mahalle

{

public string Mahalle\_adi { get; set; } // mahallenin ismini barındıracak

public Siparisler mahalle\_siparisler = new Siparisler(); // her bir mahallenin teslimatlarını barındıracak

public ArrayList adres = new ArrayList(); // mahalleleri tutan arraylist

public Mahalle(string Mahalle\_adi\_, Siparisler siparisler\_)

{

this.Mahalle\_adi = Mahalle\_adi\_;

this.mahalle\_siparisler = siparisler\_;

}

public Mahalle()

{

}

public override string ToString()

{

return Mahalle\_adi;

}

}

internal class TreeNode

{

public Mahalle data;

public TreeNode leftChild;

public TreeNode rightChild;

public TreeNode(Mahalle data\_)

{

this.data = data\_;

}

public TreeNode()

{

}

* TreeNode kısmının console kısmını almadım çünkü sadece ağaca yerleştirme işlemlerini isteniyor console kısmı b açıklamasında

internal class Tree

{

public static int derinlik = 0;

private TreeNode root;

public Tree() { root = null; }

public TreeNode getRoot()

{ return root; }

// Ağacın preOrder Dolasılması

public void preOrder(TreeNode localRoot)

{

if (localRoot != null)

{

localRoot.displayNode();

preOrder(localRoot.leftChild);

preOrder(localRoot.rightChild);

}

}

// Agacın inOrder Dolasılması

public void inOrder(TreeNode localRoot)

{

if (localRoot != null)

{

inOrder(localRoot.leftChild);

localRoot.displayNode();

inOrder(localRoot.rightChild);

}

}

// Agacın postOrder Dolasılması

public void postOrder(TreeNode localRoot)

{

if (localRoot != null)

{

postOrder(localRoot.leftChild);

postOrder(localRoot.rightChild);

localRoot.displayNode();

}

}

// istenilen mahallenin siparişlerinin turarının 150 den büyük olanın bilgilerini yazdıran metod

public void yuzelliden\_buyukler(TreeNode localRoot,string mahalle\_adi)

{

if(localRoot != null)

{

localRoot.displayNode(mahalle\_adi);

yuzelliden\_buyukler(localRoot.leftChild,mahalle\_adi);

yuzelliden\_buyukler(localRoot.rightChild,mahalle\_adi);

}

}

public void insert(Mahalle newdata)

{

TreeNode newNode = new TreeNode(newdata);

newNode.data.Mahalle\_adi = newdata.Mahalle\_adi;

if (root == null)

{

root = newNode;

}

else

{

TreeNode current = root;

TreeNode parent;

while (true)

{

parent = current;

int sonuc = String.Compare(current.data.Mahalle\_adi, newdata.Mahalle\_adi);

if (sonuc == 1)

{

current = current.leftChild;

if (current == null)

{

parent.leftChild = newNode;

return;

}

}

else

{

current = current.rightChild;

if (current == null)

{

parent.rightChild = newNode;

derinlik++;

return;

}

}

} // end while

} // end else not root

} // end insert()

}

{

static Random random\_ = new Random();

// Bu fonksiyon ile her bir mahalleye siparişleri ve mahalle isimlerini atıyorum

public static Mahalle siparisleri\_atama(string[] mahalleler)

{

Mahalle mahalle\_list = new Mahalle(); // Tüm mahalle nesnelerini toplayabileceğim bir nesne ürettim

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

{

Siparisler mahalleler\_siparis\_listesi = new Siparisler(); // mahallenin siparişlernini tutabilecek bir nesne

mahalleler\_siparis\_listesi = siparisleri\_olusturma();

Mahalle mahalle = new Mahalle(mahalleler[i], mahalleler\_siparis\_listesi);

// her bir mahalle için oluşan nesneyi mahalle listesine ekliyorum

mahalle\_list.adres.Add(mahalle);

}

return mahalle\_list;

}

// Bu fonksiyon ile her bir Sipariş listesine siparişi oluşturup atıyorum

static Siparisler siparisleri\_olusturma()

{

// mahalle nesnesine atacağım nesneyi oluşturdum

Siparisler mahalle\_siparisleri = new Siparisler();

// verilen aralıklarda random sayılar ürettim

int sipari\_sayisi = random\_.Next(3, 6);

int siparis\_bilgileri\_sayisi = random\_.Next(5, 11);

for (int j = 0; j < siparis\_bilgileri\_sayisi; ++j)

{

Siparisler tek\_siparis = new Siparisler();

for (int i = 0; i < sipari\_sayisi; ++i)

{

// istenile sayıda sipariş oluşturuluyor

Siparis siparis = new Siparis();

// her bir siparişin tutarını topluyorum

tek\_siparis.toplam\_tutar+= siparis.odenecek\_tutar;

tek\_siparis.siparisleri.Add(siparis);

}

mahalle\_siparisleri.tum\_siparisler.Add(tek\_siparis);

}

return mahalle\_siparisleri;

}

static void agac\_islemleri(Mahalle mahalleler)

{

Tree agac = new Tree();

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

{

// burada mahalle nesnelerini ağaca yerleştiriyorum

agac.insert((Mahalle)mahalleler.adres[i]);

};

}

static void Main(string[] args)

{

// verilen değerleri diziye attım

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

Mahalle mahalleler= siparisleri\_atama(MahalleAdı);

agac\_islemleri(mahalleler);

Console.ReadKey();

}

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

### 1.b.1 Kaynak Kod

//1-b maddesi için yazmış olduğunuz kod parçasını ve açıklamaları buraya ekleyiniz

* Program classında agaç işlemlerinin console kısmı

static void agac\_islemleri(Mahalle mahalleler)

{

Tree agac = new Tree();

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

{

// burada mahalle nesnelerini ağaca yerleştiriyorum

agac.insert((Mahalle)mahalleler.adres[i]);

}

Console.WriteLine("1.Soru b şıkkı:\n-----------------");

Console.Write("Agacın preOrder Dolasılması :\n-------------------------------------\n");

agac.preOrder(agac.getRoot());

Console.WriteLine("Ağacın derinliği:" + Tree.derinlik);

Console.WriteLine();

Console.WriteLine("1.Soru c şıkkı:\n-----------------");

Console.WriteLine("İstenilen mahallenin 150TL den yüksek siparişlerinin bilgileri:");

Console.WriteLine("-----------------------------------------------------------------");

// burada istenilen mahalleyi Erzene olarak aldım

agac.yuzelliden\_buyukler(agac.getRoot(),"Erzene");

}

TreeNode kısmında console kısmı

internal class TreeNode

{

public Mahalle data;

public TreeNode leftChild;

public TreeNode rightChild;

public TreeNode(Mahalle data\_)

{

this.data = data\_;

}

public TreeNode()

{

}

// kod tekrarına düşmemek için siparis\_cekme diye bir fonksiyon oluşturdum

public void siparis\_cekme(int j)

{

Console.WriteLine("Siparis:" + (j + 1));

Console.WriteLine("-------------");

// sipariş adetinin içindeki siparişler kadar dönecek döngü

for (int k = 0; k < ((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).siparisleri.Count; k++)

{

Console.WriteLine(((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).siparisleri[k] + "|");

}

Console.WriteLine("Mahallenin toplam tutarı:" + ((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).toplam\_tutar);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

public void displayNode()

{

Console.WriteLine("Mahalle:" + data.Mahalle\_adi);

Console.WriteLine("-------------");

Console.WriteLine("Siparişleri:");

Console.WriteLine("-------------");

// mahalledeki siparis adeti kadar dönecek döngü

for (int j = 0; j < data.mahalle\_siparisler.tum\_siparisler.Count; j++)

{

siparis\_cekme(j);

}

}

}

- Derinlik için tree sınıfında static bir derinlik tanımladım ve bunu sadece sol taraf taraf arttığı sırada arttırdım

internal class Tree

{

public static int derinlik = 0; - tanımladığım kısım

public void insert(Mahalle newdata)

{

TreeNode newNode = new TreeNode(newdata);

newNode.data.Mahalle\_adi = newdata.Mahalle\_adi;

if (root == null)

{

root = newNode;

}

else

{

TreeNode current = root;

TreeNode parent;

while (true)

{

parent = current;

int sonuc = String.Compare(current.data.Mahalle\_adi, newdata.Mahalle\_adi);

if (sonuc == 1)

{

current = current.leftChild;

if (current == null)

{

parent.leftChild = newNode;

return;

}

}

else

{

current = current.rightChild;

if (current == null)

{

parent.rightChild = newNode;

derinlik++; ----------------- bu kısımda arttırıyorum sadece

return;

}

}

} // end while

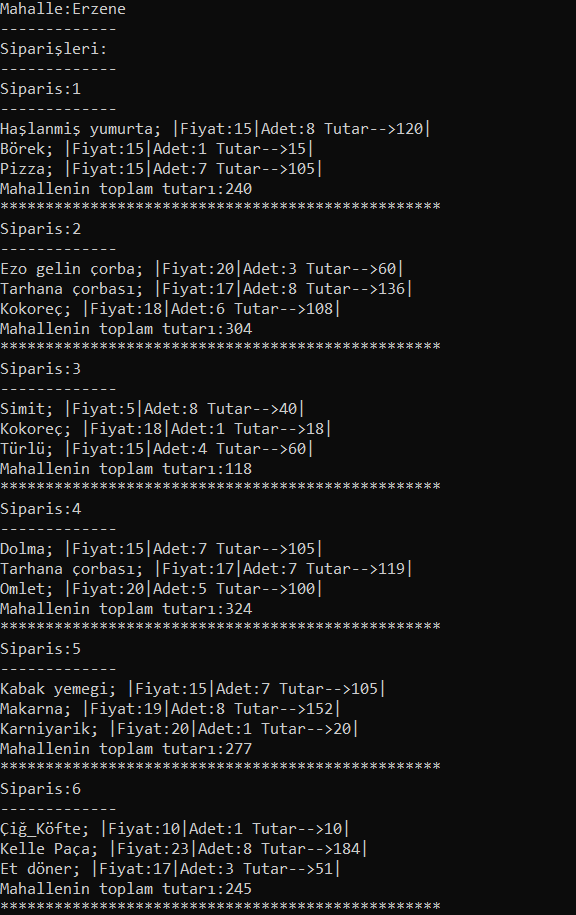
} // end else not root

} // end insert()

### 1.b.2 Ekran görüntüleri

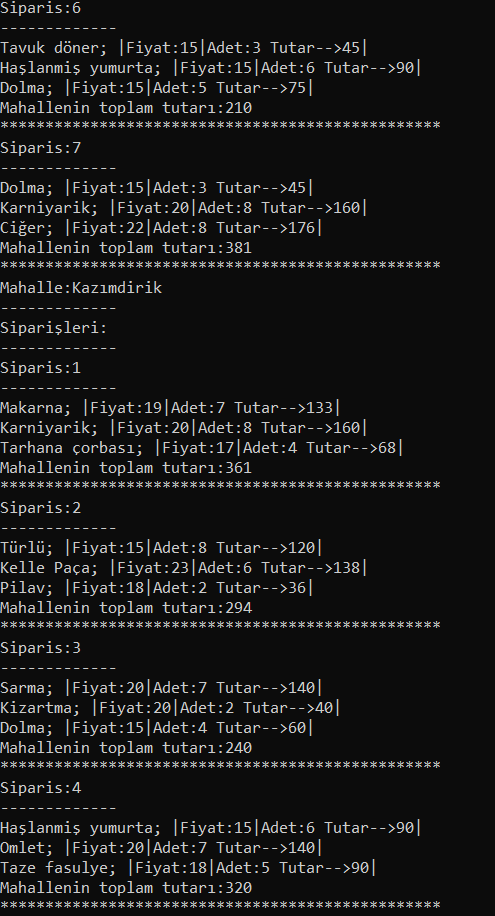
## 

## 



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin, açık hava, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.c 150 TL üstündeki siparişlerin bilgilerini listeleme

### 1.c.1 Kaynak Kod

-Tree sınıfında yuzelliden\_buyukler diye bir metod oluşturduk

public void yuzelliden\_buyukler(TreeNode localRoot,string mahalle\_adi)

{

if(localRoot != null)

{

localRoot.displayNode(mahalle\_adi);

yuzelliden\_buyukler(localRoot.leftChild,mahalle\_adi);

yuzelliden\_buyukler(localRoot.rightChild,mahalle\_adi);

}

}

- TreeNode kısmında displayNode u parametreli olarak bir daha yazdık

public void siparis\_cekme(int j)

{

Console.WriteLine("Siparis:" + (j + 1));

Console.WriteLine("-------------");

// sipariş adetinin içindeki siparişler kadar dönecek döngü

for (int k = 0; k < ((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).siparisleri.Count; k++)

{

Console.WriteLine(((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).siparisleri[k] + "|");

}

Console.WriteLine("Mahallenin toplam tutarı:" + ((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).toplam\_tutar);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//istenilen mahalleye göre çıktı döndüren displayNode

public void displayNode(string istenilen\_data)

{

int ilk = 1;

for (int j = 0; j < data.mahalle\_siparisler.tum\_siparisler.Count; j++)

{

if (istenilen\_data == data.Mahalle\_adi && ((Siparisler)data.mahalle\_siparisler.tum\_siparisler[j]).toplam\_tutar>150)

{

if (ilk==1)

{

Console.WriteLine("Mahalle:" + data.Mahalle\_adi);

Console.WriteLine("-------------");

Console.WriteLine("Siparişleri:");

Console.WriteLine("-------------");

ilk += 1;

}

siparis\_cekme(j);

}

}

}

- Program sınıdında ağaç işlemleri metodunda çağırdım kısım

Console.WriteLine("1.Soru c şıkkı:\n-----------------");

Console.WriteLine("İstenilen mahallenin 150TL den yüksek siparişlerinin bilgileri:");

Console.WriteLine("-----------------------------------------------------------------");

// burada istenilen mahalleyi Erzene olarak aldık

agac.yuzelliden\_buyukler(agac.getRoot(),"Erzene");

### 1.c.2 Ekran görüntüleri

## metin içeren bir resim Açıklama otomatik olarak oluşturuldu

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

### 1.d.1 Kaynak Kod

### 1.d.2 Ekran görüntüleri

## 2.a Hash Tablosuna Ekleme

using System;

using System.Collections;

using System.Collections.Generic;

namespace HashTableBornova

{

class Program

{

public static void Main(string[] args)

{ // Creates and initializes a new Hashtable.

var BornovaMah = new Hashtable();

BornovaMah.Add("ERZENE ", 35135);

BornovaMah.Add("KAZIMDİRİK", 33934);

BornovaMah.Add("YEŞİLOVA ", 31008);

BornovaMah.Add("ATATÜRK", 28912);

BornovaMah.Add("İNÖNÜ", 25778);

BornovaMah.Add("MEVLANA ", 25492);

BornovaMah.Add("DOĞANLAR", 21461);

BornovaMah.Add("RAFET PAŞA", 19476);

BornovaMah.Add("NALDÖKEN ", 9892);

BornovaMah.Add("ZAFER", 9876);

//INCREASING THE POPULATION

char firstLetter;

Console.Write("Enter the first letter of county which you want to increase it's population: ");

firstLetter = Console.ReadLine()[0];

// Displays the Hashtable.

Console.WriteLine("The Hashtable contains the following:");

IDictionaryEnumerator enumerator = BornovaMah.GetEnumerator();

while (enumerator.MoveNext())

{

string key = enumerator.Key.ToString();

int value = enumerator.Value.GetHashCode();

Console.WriteLine((key, value));

}

List<string> list = new List<string>();

foreach (object item in BornovaMah.Keys)

{

string countyName = (string)item;

if (countyName[0].CompareTo(firstLetter) == 0)

{

list.Add(countyName);

}

}

foreach (string item in list)

{

BornovaMah[item] = (int)BornovaMah[item] + 1;

## }2.b Hash Tablosu Güncelleme

char firstLetter;

Console.Write("Enter the first letter of county which you want to increase it's population: ");

firstLetter = Console.ReadLine()[0];

// Displays the Hashtable.

Console.WriteLine("The Hashtable contains the following:");

IDictionaryEnumerator enumerator = BornovaMah.GetEnumerator();

while (enumerator.MoveNext())

{

string key = enumerator.Key.ToString();

int value = enumerator.Value.GetHashCode();

Console.WriteLine((key, value));

}

List<string> list = new List<string>();

foreach (object item in BornovaMah.Keys)

{

string countyName = (string)item;

if (countyName[0].CompareTo(firstLetter) == 0)

{

list.Add(countyName);

}

}

foreach (string item in list)

{

BornovaMah[item] = (int)BornovaMah[item] + 1;

}

Console.WriteLine("After increasing the population");

foreach (DictionaryEntry entry in BornovaMah)

{

// The new population

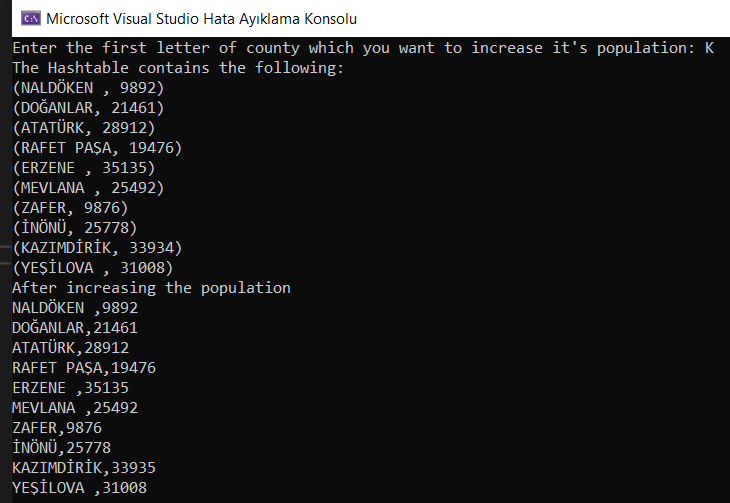
Console.WriteLine("{0},{1}", entry.Key, entry.Value);

}

}

}

}



## 3.a Heap Veri Yapısı Tasarlama

using System;

using System.Collections;

using System.Collections.Generic;

using System.Collections.ObjectModel;

namespace HeapBornova

{

internal class Hood

{

public string Name;

public int Number;

}

public class example

{

static void heapSort(Tuple<int, string>[] arr)

{

int n = arr.Length;

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{

(arr[0], arr[i]) = (arr[i], arr[0]);

heapify(arr, i, 0);

}

}

static void heapify(Tuple<int, string>[] arr, int n, int i)

{

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && arr[left].Item1 > arr[largest].Item1)

largest = left;

if (right < n && arr[right].Item1 > arr[largest].Item1)

largest = right;

if (largest != i)

{

(arr[i], arr[largest]) = (arr[largest], arr[i]);

heapify(arr, n, largest);

}

## }3.b Max Heap düğüm yerleştirme

namespace HeapBornova

{

internal class Hood

{

public string Name;

public int Number;

}

public class example

{

static void heapSort(Tuple<int, string>[] arr)

{

int n = arr.Length;

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{

(arr[0], arr[i]) = (arr[i], arr[0]);

heapify(arr, i, 0);

}

}

static void heapify(Tuple<int, string>[] arr, int n, int i)

{

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && arr[left].Item1 > arr[largest].Item1)

largest = left;

if (right < n && arr[right].Item1 > arr[largest].Item1)

largest = right;

if (largest != i)

{

(arr[i], arr[largest]) = (arr[largest], arr[i]);

heapify(arr, n, largest);

}

}

public static void Main()

{

Tuple<int, string>[] arr = {

Tuple.Create(35135, "ERZENE "),

Tuple.Create(33934, "KAZIMDİRİK"),

Tuple.Create(31008, "YEŞİLOVA"),

Tuple.Create(28912, "ATATÜRK"),

Tuple.Create(25778, "İNÖNÜ"),

Tuple.Create(25492, "MEVLANA "),

Tuple.Create(432, "LAKA "),

Tuple.Create(21461, "DOĞANLAR"),

Tuple.Create(19476, "RAFETPAŞA"),

Tuple.Create(13464, "GAZİOSMANPAŞA")

};

Console.WriteLine("Initial array is: ");

foreach (var pair in arr)

{

Console.Write(pair.Item2 + " ");

## }3.c Heap bilgi çekme

### 3.c.1 Kaynak Kod

foreach (var pair in arr)

{

Console.Write(pair.Item2 + " ");

}

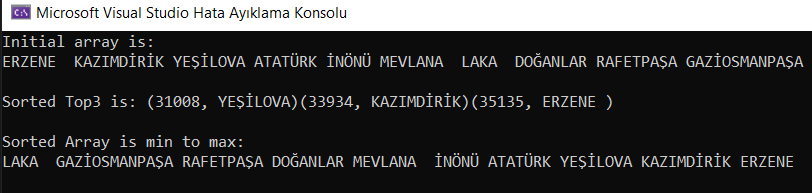
Console.WriteLine();

Console.WriteLine();

heapSort(arr);

Console.WriteLine("Sorted Top3 is: " + arr[7] + arr[8] + arr[9]);iz

### 3.c.2 Ekran görüntüleri

## 4.a Simple sorting veya Advanced sorting algoritması

Simple Sorting’den Bubble Sort kullanıldı. Bubble ve Selection Sort öğrenilip karşılaştırıldı

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BubleSort

{

class Program

{

static void Main(string[] args)

{

int gecici;

Console.Write("Kaç Adet Sayı Girilecek : ");

int k = Convert.ToInt32(Console.ReadLine());

int[] dizi = new int[k];

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

{

Console.Write("Dizinin {0}. elemanını girin : ", i + 1);

dizi[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("---------------------------------------");

Console.WriteLine("Oluşturulan Dizi");

DiziYazdir(dizi);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bubble Sort Sıralama\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for (int i = 0; i <= dizi.Length - 1; i++)

{

for (int j = 1; j <= dizi.Length - 1; j++)

{

if (dizi[j - 1] > dizi[j])

{

gecici = dizi[j - 1];

dizi[j - 1] = dizi[j];

dizi[j] = gecici;

}

}

DiziYazdir(dizi);

}

Console.WriteLine("Dizinin Sıralı Hali");

DiziYazdir(dizi);

Console.ReadKey();

}

public static void DiziYazdir(int[] dizi)

{

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

{

Console.Write("{0} ", dizi[i]);

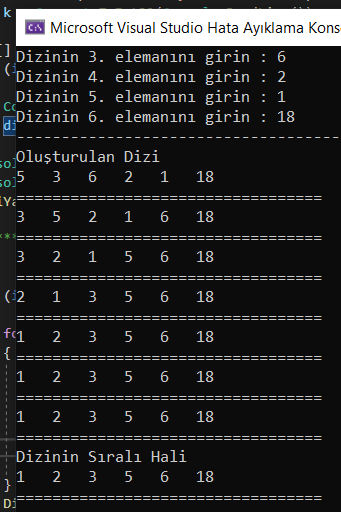
}

Console.WriteLine();

Console.WriteLine("==================================");

}

}



## }4.b Sıralama algoritması ile karşılaştırılması

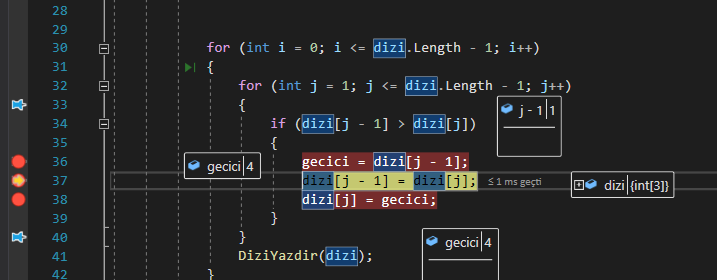
## Bubble Sort ve Selection Sort'u kitabımızdan okuduk ve karşılaştırdık. Bubble Sort'un best case zaman karışıklığı O(n) iken bu durum Selection Sort'ta (n^2) (Selection Sort'ta durumdan bağımsız olarak (n^2) iken bu durum Bubble Sort’ta değişkenlik göstermektedir.) Bizim kullandığımız Bubble Sort’ta elemanlarımız  [6, 5, 4, 3, 2, 1] olsun. Bu da 15 karşılaştırma ve değişim işlemi demek. Bunu da şu şekilde hesaplayabiliriz: Altı elemam çarpı beş karşılaştırma ve değişim işlemi; ikiye bölünür, çünkü tüm yinelemelerde ortalama olarak öğelerin yarısı karşılaştırılır ve değiştirilir.

## 6 × 5 × ½   =   30 × ½   =   15

6'yı n ile değiştirirsek, şunu elde ederiz: n × (n – 1) × ½

Çarpıldığında, bu bize şunu verir: ½ (n² – n)

Bu terimde n'nin en yüksek kuvveti n²'dir; Öyleyse: Bubble Sort’un Worst Case zaman karmaşıklığı: O(n²)’dir.



Debug aracılığıyla değişimlerin gözlemlenmesi

## 4.c Görselleştirme araçları

İlgili video sıralama algoritmalarının hangi farklı mantıklarla çalıştığını anlamamda yardımcı oldu. Görselleştirme adına farklı kaynaklardan da faydalandık.

<https://www.happycoders.eu/algorithms/bubble-sort/>

<https://www.kodlamamerkezi.com/algoritma/c-ile-kabarcik-siralama-bubble-sort-algoritmasi/>

Adreslerindeki görselleştirmeler Bubble Sort algoritmasını daha iyi kavramamızı sağladı. Aynı şekilde zaman karmaşıklığı da ilk adreste oldukça sade anlatılmış. Genel olarak Veri yapılarını öğrenirken görsel ve işitsel yönden öğrenme süresince de desteklenmeli. Konu üzerine hazırlanan (aynı zamanda kodunu nu da gördüğümüz) animasyon, video ve görsellerin işi daha da basitleştirdiğini düşünmekteyiz.

# Özdeğerlendirme Tablosu

**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 3 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| **1 a) Ağaç (Yemek Sipariş Ağacı Oluşturma)** | **10** | **10** | **Siparişler dizi/liste olarak ağaçlara isim bazlı olarak yerleştirildi** |
| **1 b) Derinlik Bulma, Ağacı Listeleme** | **10** | **10** | **Kökten sonra sola her eklendiğinde derinliğin arttırarak derinlik bulundu** |
| **1 c) Arama ve Listeleme** | **10** | **10** | **Istenilen koşul şatları altında istenilen sonuçlar elde edildi** |
| **1 d) Ağaçta bir yemek türüne indirim yapma** | **10** | **0** |  |
| **2) Hash Tablosu** | **15** | **15** | **Hash tablosu siteden alınan bilgilere göre .Add kullanılarak dolduruldu** |
| **3) Yığın Ağacı (Heap)** | **15** | **15** | **Heap algoritması öğrenildi. Mahalle verileri ve nüfusları Heap’e yerleştirilip sıralandı.** |
| **4) Sıralama Algoritmaları** | **20** | **20** | **Sıralama Algoritmaları kitaptan okundu öğrenildi. İnternet üzerinden iyi kaynaklardan faydanıldı ve uygulandı.** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **Tablo dürüst ve açık bir biçimde dolduruldu** |
| **Toplam** | **100** | **90** |  |

**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.**