Veri

Yapıları Proje 3B Raporu

Saifullah NASRULLAH 05120000826

1)a) Ürünler dizisine 3 tane daha ürün bilgileri eklenmesi ve daha sonra ürünler nesneleri oluşturulup ürün adına göre ikili ağaca yerleştirilmesi istenmektedir.

1)b)dizideki string’leri sahalarına ayrıştırarak ürün adına göre kategorilerine yerleştirilmesi istenmektedir.

1)c)kategori ağaçlarının derinliği, eleman sayısı ,düğümlerim derinlik ortalamasını ve toplam kar miktar istenmektedir.

**Kaynak kod….**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B

{

class Program

{

static void Main(string[] args)

{

string[] ürünler ={"Dizüstü bilgisayar,Bilgisayar,Dell,XPS 13 9333 intel core i5,10,3799,5927",

"Buzdolabı,Beyaz eşya,Regal,CoolRGL 3000,45,699,929",

"LandCruiser,Araba,Toyota,Prado,5,20000,250000",

"iphone,Telefon,Apple,iphone 5s 16 GB,17,1150,2500",

"Piyano,Müzik Alet,Kawai,GX-5,15,12000,87000"};

Tree ağaç = new Tree();

Console.WriteLine("Ürünlerin İlk Hali:" +

"\n-------------------\n");

for (int i = 0; i < 5; i++)

{

ÜrünNesnesi ürünNesnesi = new ÜrünNesnesi();

string[] properties = ürünler[i].Split(',');//Dizideki her indeksini ,e göre ayırıyor

ürünNesnesi.ürünAdı = properties[0];

ürünNesnesi.kategori = properties[1];

ürünNesnesi.marka = properties[2];

ürünNesnesi.model = properties[3];

ürünNesnesi.miktar = Convert.ToInt32(properties[4]); //to int

ürünNesnesi.maliyet = Convert.ToDouble(properties[5]); //to double

ürünNesnesi.satışFiyatı = Convert.ToDouble(properties[6]);

ürünNesnesi.yazdir();

ağaç.insert(ürünNesnesi);

}

Console.WriteLine("\nÜrünlerin isimlerine Göre Sıralanması: "+

"\n--------------------------------------------\n");

ağaç.inOrder(ağaç.getRoot());

Console.WriteLine("\nİstatistikler:" +

"\n--------------");

Console.WriteLine("\tAğacın Derinliği: " + ağaç.FindDepthOfTree(ağaç.getRoot())

+ "\n\tAğacın Eleman Sayısı: " + ağaç.düğümSayısı

+ "\n\tAğacın Düğümlerinin Derinlik Ortalaması: " + (double)(ağaç.sumOfDepth(ağaç.getRoot()) / ağaç.düğümSayısı)

+ "\n\tToplam Kar Miktarı: " + ağaç.karToplamı);

Console.ReadKey();

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B

{

class ÜrünNesnesi

{

public string ürünAdı, marka, model, kategori;

public int miktar;

public double maliyet, satışFiyatı;

public void yazdir()

{

Console.WriteLine(ürünAdı

+ "\n\tKategorisi: " + kategori

+ "\n\tMarkası: " +marka

+ "\n\tModeli: " + model

+ "\n\tMiktarı: " + miktar

+ "\n\tMaliyeti: " + maliyet

+ "\n\tSatış Fiyatı: " + satışFiyatı);

}

}

// Düğüm Sınıfı

class TreeNode

{

public ÜrünNesnesi ürün = null;//başlangıç için null alıyor..

public TreeNode left;

public TreeNode right;

public void NodeGörüntüle()

{

ürün.yazdir();

}

}

// Ağaç Sınıfı

class Tree

{

private TreeNode root;//ağacın en üstteki düğümü yani ROOT düğümü

public int düğümSayısı = 0;

public int karToplamı = 0;

//public ArrayList kar = new ArrayList();

public Tree()

{

root = null;//ilkleme

}

public TreeNode getRoot()

{

return root;//en üst düğüm döndürülüyor..

}

// Agacın inOrder Dolasılması

public void inOrder(TreeNode localRoot)

{

if (localRoot != null)

{

inOrder(localRoot.left);//sol çocuğunda çocuklarına bakılıyor. aynısı onun çocukları için de gerçekleşecek.(Recursive)

localRoot.NodeGörüntüle();// O anki düğümün bilgileri yazdırılıyor..

inOrder(localRoot.right);

}

}

//Ağacın derinliğini hesaplama

public int FindDepthOfTree(TreeNode n)

{

if (n == null) return 0;

return (Math.Max(FindDepthOfTree(n.left), FindDepthOfTree(n.right)) + 1);

}

// derinlik toplamları hesaplama

public int sumOfDepth(TreeNode n)

{

int leftDepth = 0, rightDepth = 0, sumDepth = 0;

if (n == null)// null ise çocuğu da yoktur demek ki..

return (0);

leftDepth = sumOfDepth(n.left) + 1;//Sol ya da sağ çocuk kendi derinliğini bulurken kendini saymadığı için 1 ekledik..

rightDepth = sumOfDepth(n.right) + 1;

sumDepth = leftDepth + rightDepth;

return (sumDepth++);

}

public void insert(ÜrünNesnesi ürün)

{

TreeNode newNode = new TreeNode();

newNode.ürün = ürün;

karToplamı += Convert.ToInt32(ürün.miktar \* (ürün.satışFiyatı - ürün.maliyet));

düğümSayısı++;

if (root == null)

root = newNode;

else

{

TreeNode current = root;

TreeNode parent;

while (true)

{

parent = current;

if (ürün.ürünAdı.CompareTo(current.ürün.ürünAdı) == -1)//compareTo fonksiyonu ilk veri daha küçükse -1, daha büyükse 1, eşitse 0 döndürür..

{

current = current.left;

if (current == null)

{

parent.left = newNode;

return;

}

}

else

{

current = current.right;

if (current == null)

{

parent.right = newNode;

return;

}

}

}

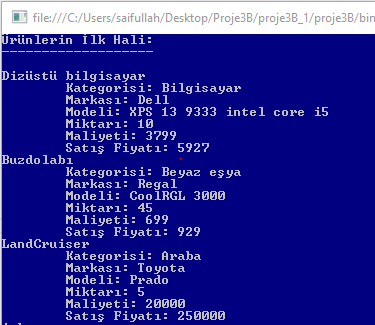
}

}

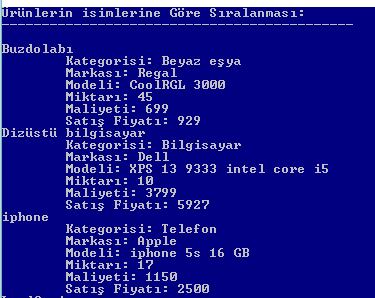
}

}

**Ekran Çıktısı**..





2) a) Birinci soruda belirtilen ürün nesnelerini ad+marka+model bileşimine göre Hash tabe’e eklenmesi istenmektedir.

2) b)Tüm ürünlerin fiyatine %5 indirimle hash tablosunda güncellenmesi istenmektedir..

**Kaynak Kod..**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace proje3B\_2

{

class Program

{

static void Main(string[] args)

{

string[] ürünler ={"Dizüstü bilgisayar,Dell,XPS 13 9333 intel core i5,10,3799,5927",

"Buzdolabı,Regal,CoolRGL 3000,45,699,929",

"LandCruiser,Toyota,Prado,5,20000,250000",

"iphone,Apple,iphone 5s 16 GB,17,1150,2500",

"Piyano,Kawai,GX-5,15,12000,87000"};

Hashtable hstb = new Hashtable();

ÜrünNesnesi obje = new ÜrünNesnesi();

Console.WriteLine("Ürünleri hashtable,e eklenmesi:\n");

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

{

string[] parçalama = ürünler[i].Split(',');

obje.adı = parçalama[0];

obje.marka = parçalama[1];

obje.model = parçalama[2];

obje.miktar = Convert.ToInt32(parçalama[3]);

obje.maliyet = Convert.ToDouble(parçalama[4]);

obje.satişFiyat = Convert.ToDouble(parçalama[5]);

hstb.Add(parçalama[0], obje);

Console.WriteLine( obje.adı+ " markasi "+ obje.marka+ " modeli "+ obje.model+ " miktari " +obje.miktar

+" maliyeti " +obje.maliyet+ " satış fiyatı " +obje.satişFiyat+"\n");

}

Console.WriteLine("Ürünlerin %5 indirimlerimle güncel hali :\n");

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

foreach (DictionaryEntry indis in hstb)

{

ÜrünNesnesi newproduct = new ÜrünNesnesi();

newproduct = (ÜrünNesnesi)indis.Value;

double yeniFiyat = obje.fiyatGüncelle(newproduct);

Console.WriteLine(newproduct.ürünAdı + " Markası: " + newproduct.marka + " Modeli: " + newproduct.model

+ " Miktarı: " + newproduct.miktar + " Maliyeti: " + newproduct.maliyet + " Satış Fiyatı: " + newproduct.satişFiyat

+ " İndirimli Fiyatı: " + yeniFiyat);

}

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_2

{

class ÜrünNesnesi

{

public string adı, marka, model;

public int miktar;

public double maliyet, satişFiyat;

public double fiyatGüncelle(ÜrünNesnesi ürün)

{

return ürün.satişFiyat \* 0.95;

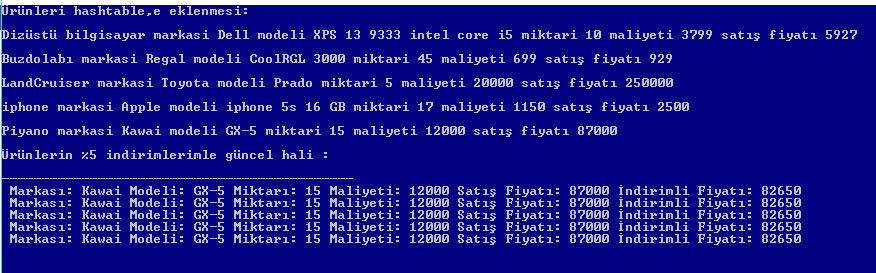
}

public string ürünAdı { get; set; }

}

}

**Ekran Çıktısı…**



3)a)Sadece ürünlerin maliyetleri bir yığına heap’e yerleştirilmesi isteniyor.

3)b)maliyetlerine göre en ucuz 2 ve en pahalı 2 ürünün heap’ten çıkarılması ve listenlemesi isteniyor.

**Kaynak Kod..**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_3

{

class Node

{

private int iData;

public Node(int key)

{

iData = key;

}

public int getKey()

{

return iData;

}

public void setKey(int id)

{

iData = id;

}

}

class Heap

{

private Node[] heapArray;

private int maxSize;

private int currentSize;

public int removedPrice;

public Heap(int mx)

{

maxSize = mx;

currentSize = 0;

heapArray = new Node[maxSize];

}

public bool isEmpty()

{

return currentSize == 0;

}

public bool insert(int key)

{

if (currentSize == maxSize)

return false;

Node newNode = new Node(key);

heapArray[currentSize] = newNode;

trickleUp(currentSize++);

return true;

}

public void trickleUp(int index)

{

int parent = (index - 1) / 2;

Node bottom = heapArray[index];

while (index > 0 &&

heapArray[parent].getKey() < bottom.getKey())

{

heapArray[index] = heapArray[parent];

index = parent;

parent = (parent - 1) / 2;

}

heapArray[index] = bottom;

}

public Node remove()

{

Node root = heapArray[maxSize-1];

removedPrice = heapArray[--currentSize].getKey();

heapArray[maxSize-1] = heapArray[currentSize];

trickleDown(maxSize-1);

return root;

}

public void trickleDown(int index)

{

int largerChild;

Node top = heapArray[index];

while (index < currentSize / 2)

{

int leftChild = 2 \* index + 1;

int rightChild = leftChild + 1;

if (rightChild < currentSize &&

heapArray[leftChild].getKey() <

heapArray[rightChild].getKey())

largerChild = rightChild;

else

largerChild = leftChild;

if (top.getKey() >= heapArray[largerChild].getKey())

break;

heapArray[index] = heapArray[largerChild];

index = largerChild;

}

heapArray[index] = top;

}

public bool change(int index, int newValue)

{

if (index < 0 || index >= currentSize)

return false;

int oldValue = heapArray[index].getKey();

heapArray[index].setKey(newValue);

if (oldValue < newValue)

trickleUp(index);

else

trickleDown(index);

return true;

}

public void displayHeap()

{

Console.Write("heapArray: ");

for (int m = 0; m < currentSize; m++)

if (heapArray[m] != null)

Console.Write(heapArray[m].getKey() + " ");

else

Console.Write("-- ");

Console.WriteLine();

int nBlanks = 32;

int itemsPerRow = 1;

int column = 0;

int j = 0;

String dots = "...............................";

Console.WriteLine(dots + dots);

while (currentSize > 0)

{

if (column == 0)

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

Console.Write(' ');

Console.Write(heapArray[j].getKey());

if (++j == currentSize)

break;

if (++column == itemsPerRow)

{

nBlanks /= 2;

itemsPerRow \*= 2;

column = 0;

Console.WriteLine();

}

else

for (int k = 0; k < nBlanks \* 2 - 2; k++)

Console.Write(' ');

}

Console.WriteLine("\n" + dots + dots);

}

} // Source: http://homepage.cs.uiowa.edu/~sriram/21/spring07/code/heap.java

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace proje3B\_3

{

class Program

{

static void Main(string[] args)

{

string[] ürünler ={"Dizüstü bilgisayar,Dell,XPS 13 9333 intel core i5,10,3799,5927",

"Buzdolabı,Regal,CoolRGL 3000,45,699,929",

"LandCruiser,Toyota,Prado,5,20000,250000",

"iphone,Apple,iphone 5s 16 GB,17,1150,2500",

"Piyano,Kawai,GX-5,15,12000,87000"};

Heap heap = new Heap(5);

ArrayList arrayList = new ArrayList();

for (int i = 0; i < 5; i++)

{

String[] parçaÜrün = ürünler[i].Split(',');

heap.insert(Convert.ToInt32(parçaÜrün[4]));

}

Console.WriteLine("Yerleştirilen Maliyat:\n");

heap.displayHeap();

for (int i = 0; i < 2; i++)

{

heap.remove();

arrayList.Add(heap.removedPrice);

}

Console.Write("\nMaliyeti En Düşük iki Ürün: ");

foreach (int i in arrayList)

Console.Write(i + " ");

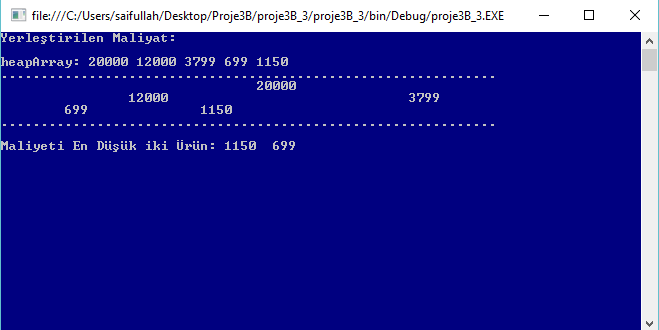
Console.ReadKey();

}

}

}

**Ekran Çıktısı..**



4)a)(Simple sorting)Sıralama algoritmaları öğrenip daha sonra kağıda elle yazarak ve kodlayarak çalıştırılması isteniyor.

**Kaynak Kod..**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_4

{

class SimpleSort

{

public Int32[] dizi;

public int boyut;

public SimpleSort(int Boyut)

{

dizi = new Int32[Boyut];

boyut = 0;

}

public void insert(int eklenen)

{

dizi[boyut] = eklenen;

boyut++;

}

public void display()

{

Console.WriteLine("Dizideki Elemanlar: \n"

+ "-------------------");

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

Console.WriteLine(dizi[i] + " ");

}

public void selectionSort()

{

int indeks = 0;

for (int i = 0; i < boyut-1; i++)

{

indeks = i;

for (int j = i+1; j < boyut; j++)

{

if (dizi[j] < dizi[indeks])

indeks = j;

}

if(indeks != i)

taşıma(i, indeks);

}

}

public void taşıma(int a, int b)

{

Int32 geçici = dizi[a];

dizi[a] = dizi[b];

dizi[b] = geçici;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_4

{

class Program

{

static void Main(string[] args)

{

Random rnd = new Random();

int boyut = 5;

SimpleSort arr = new SimpleSort(boyut);

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

{

int eleman = rnd.Next(1000);

arr.insert(eleman);

}

arr.display();

Console.WriteLine("\nDizideki elemanları sıralama..\n");

arr.selectionSort();

arr.display();

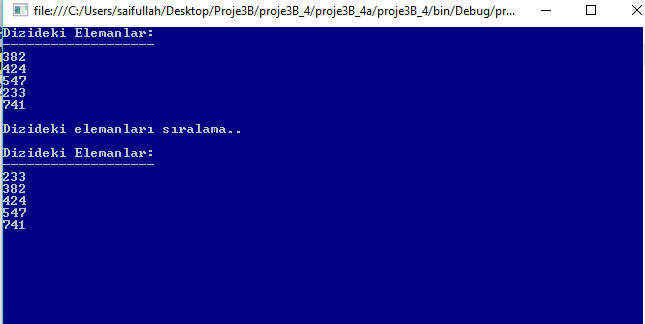
Console.ReadKey();

}

}

}

**Ekran Çıktısı..**



4)b)(Advanced sorting) Sıralama algoritmasını öğrenip kodlayarak çalıştırılması isteniyor.

4)c)her iki yöntemin zaman karmaşıklarını hesaplayarak bir paragraf açıklama yazılacak.

4)d)ilgili adresteki videonun izlenmesi..

**Kaynak Kod..**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_4b

{

class ArrayShellSort

{

public Int32[] dizi;

public int boyut;

public ArrayShellSort(int Boyut)

{

dizi = new Int32[Boyut];

boyut = 0;

}

public void insert(int eklenen)

{

dizi[boyut] = eklenen;

boyut++;

}

public void display()

{

Console.WriteLine("Dizideki Elemanlar: \n"

+ "-------------------");

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

Console.WriteLine(dizi[i] + " ");

}

public void shellSort()

{

int inner, outer;

int temp;

int h = 1;

while (h <= boyut / 3)

h = h \* 3 + 1;

while (h > 0)

{

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

{

temp = dizi[outer];

inner = outer;

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

{

dizi[inner] = dizi[inner - h];

inner -= h;

}

dizi[inner] = temp;

}

h = (h - 1) / 3;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace proje3B\_4b

{

class Program

{

static void Main(string[] args)

{

Random rnd = new Random();

int boyut = 5;

ArrayShellSort arr = new ArrayShellSort(boyut);

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

{

int eleman = rnd.Next(1500);

arr.insert(eleman);

}

arr.display();

Console.WriteLine("\nDizideki elemanlar sıralanıyor..\n");

arr.shellSort();

arr.display();

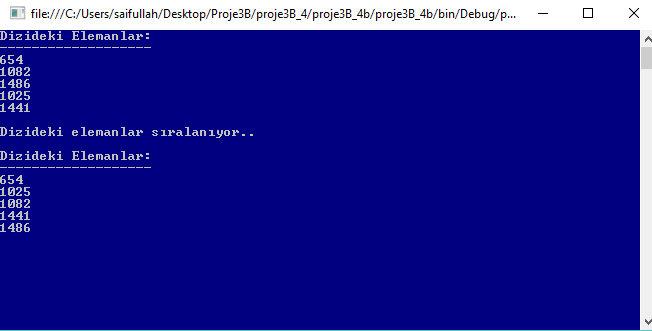
Console.ReadKey();

}

}

}

**Ekran Çıktısı.**



**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 3 Maddeleri (her maddede kodlar ve ekran görüntüleri de yer alacaktır)** | **Not** | **Tahmini Not** | **Açıklama** |
| **1 a) Ağaç** | **10** | **10** | **Doğur çalışıyor** |
| **1 b) Kategori Ağacı** | **10** | **10** | **Doğru çalışıyor** |
| **1 c) Değerlerin hesaplanması, listelemeler** | **10** | **9** | **Doğru hespalmasından emein değilim** |
| **2) Hash Tablosu** | **20** | **20** | **Doğru ekleniyor** |
| **3) Yığın Ağacı (Heap)** | **20** | **20** | **Doğru çalışıyor** |
| **4) Sıralama Algoritmaları** | **20** | **17** | **A ve B şıklar yapıldı C şıkında karşılaştırma olmadı.** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **Yapıldı** |