**EGE UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2013-2014 FALL SEMESTER**

**PROJECT 3 : İkili Arama Ağacı, Hash Tablosu,Yığın Ağacı,Sıralama Algoritmaları**

**Date : 12.12.2013**

**Elenur BİLGİN : 05110000040**

**PROGRAMLARIN KODLARI ve EKRAN GÖRÜNTÜLERİ**

1. **PROGRAM**

class Program

{

static void Main(string[] args)

{

string[] ülke\_adları = { "Japonya", "ABD", "Fransa", "Çin", "Hindistan", "İtalya", "Rusya", "Almanya", "Türkiye", "İspanya"};

int[] sıra = { 10, 3, 21, 1, 2, 23, 9, 16, 18, 28}; //en kalabalık ülkeler arasında dünyadaki sırasını belirtir

Tree agac = new Tree();

for (int i = 0; i <ülke\_adları.Length; i++) //10 tane ülke nesnesi oluşturulup ağaca yerleştirilirler...

{

Ülke nesne = new Ülke();

nesne.ülke\_adı = ülke\_adları[i];

nesne.ülke\_sırası = sıra[i];

nesne.nüfus = 0;

nesne.sayı = i;

agac.insert(nesne);

}

agac.bilgileri\_yazdır(); //Agacın derinliğini, elemansayısını, düğümlerin derinlik ortalamasını bulan fonfsiyon çağırılır...

Console.ReadLine();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Ülke // Ülke sınıfı oluşturulur...

{

public string ülke\_adı;

public int ülke\_sırası;

public int nüfus;

public int sayı; // 1-b için pratik yöntem olarak sayı değişkenini kullandım...

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class TreeNode // Hazır TreeNode sınıfını kullandım...

{

public Ülke data; // Data ülke tipinde yapıldı...

public TreeNode leftChild;

public TreeNode rightChild;

public void displayNode()

{

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

Console.WriteLine("{0,-15}{1,-10}{2,-10}",data.ülke\_adı,data.ülke\_sırası, data.nüfus);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Tree // Hazır Tree sınıfı kullanıldı...

{

public TreeNode root;

public int d=-1,toplam=0;

public int sayac = 0, düzey;

public Tree() { root = null; } // Root yani köke null atadık...

public void insert(Ülke newdata) // Ağaca eleman ekleyen hazırfonksiyon...

{

TreeNode newNode = new TreeNode();

newNode.data = newdata;

if (root == null)

root = newNode;

else

{

TreeNode current = root;

TreeNode parent;

while (true)

{

parent = current;

if (newdata.ülke\_sırası < current.data.ülke\_sırası) // Burada ağaca ülke sırasına göre ekleme yaptım...

{

current = current.leftChild;

if (current == null)

{

parent.leftChild = newNode;

return;

}

}

else

{

current = current.rightChild;

if (current == null)

{

parent.rightChild = newNode;

return;

}

}

} // end while

}// end else not root

}// end insert

public void derinlik\_ortalama\_elemansay\_bulma(TreeNode root ,int[] nüfus)

{

if ( root!= null)

{

d++;

derinlik\_ortalama\_elemansay\_bulma(root.leftChild,nüfus);

sayac++; /// eleman sayısını arttırır..

root.data.nüfus=nüfus[root.data.sayı]; // Başda nüfusu 0 olan ülkelerin nüfusları güncellenmiştir...

toplam += d; // her elemanın düzeyi toplanmıştır...

if (düzey < d)

{

düzey = d; /// derinliği bulur...

}

derinlik\_ortalama\_elemansay\_bulma(root.rightChild,nüfus);

d--;

}

}

public void bilgileri\_yazdır() // Yukarıdaki fonksiyonu kullanabilmek için bu fonksiyonu çağıran ağaç sınıfı içerisinde bir metot yapıldı...

{

Console.WriteLine("Ükelerin nüfusları güncellenmeden önce :");

yazdır(root);

int[] nüfuslar = { 127350000, 316418000, 65707000, 1359250000, 1232320000, 59704082, 143400000, 80493000, 75627384, 47059533 };

derinlik\_ortalama\_elemansay\_bulma(root,nüfuslar);

Console.WriteLine("Ülkelerin nüfusları güncellendikden sonra :");

yazdır(root);

Console.WriteLine("Bu ağacın derinliği = {0}",düzey); // Ekrana ağacın düzeyini yazdırır...

Console.WriteLine("Bu ağacın eleman sayısı= {0}", sayac); // Ekrana ağacın eleman sayısını yazdırır...

Console.WriteLine("Bu ağacın ortalama derinliği=" + (double)toplam / sayac); // Ekrana düğümlerin derinlik ortalamasını yazdırır...

}

public void yazdır(TreeNode root)

{

if (root != null)

{

root.displayNode();

yazdır(root.leftChild);

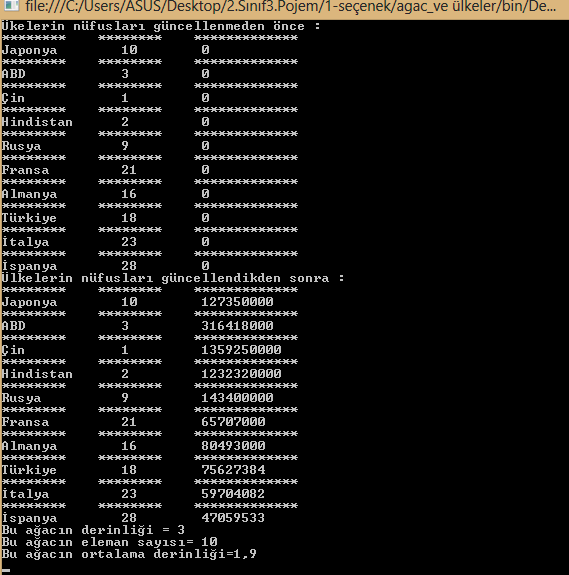
yazdır(root.rightChild);

}

}

}

1.Programın ekran çıktısı



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **PROGRAM**

class Program

{

static void Main(string[] args)

{

string[] ülke\_adları = { "Japonya", "ABD", "Fransa", "Çin", "Hindistan", "İtalya", "Rusya", "Almanya", "Türkiye", "İspanya" };

int[] sıra = { 10, 3, 21, 1, 2, 23, 9, 16, 18, 28 }; //en kalabalık ülkeler arasında dünyadaki sırasını belirtir

int[] nüfuslar = { 127350000,316418000, 65707000, 1359250000, 1232320000, 59704082, 143400000, 80493000, 75627384,47059533};

Hashtable tablo = new Hashtable(); // Hashtablo tipinde tablo oluşturulur...

for (int i = 0; i < 10; i++) //10 tane nesne oluşturulur ve bu nesneler tablo ya eklenir...

{

Ülke ülkeler = new Ülke();

ülkeler.ülke\_adı = ülke\_adları[i];

ülkeler.ülke\_sırası = sıra[i];

ülkeler.nüfus = nüfuslar[i];

tablo.Add(ülkeler.ülke\_adı,ülkeler);

}

Console.WriteLine("Üzerinde işlem yapılmadan önceki ülkelerin bilgileri...");

yazdır(tablo);

Console.WriteLine("Ülkelerin nüfusları %10 arttırılmış ülkelerin bilgileri... ");

hashtable\_hesapla(tablo);

yazdır(tablo);

Console.ReadLine();

}

static public void hashtable\_hesapla(Hashtable tablo) // Ülkelerin nüfuslarını %10 oranında arttıran fonksiyon...

{

foreach (DictionaryEntry entry in tablo)

{

((Ülke)entry.Value).nüfus =(int) Math.Floor((double)((Ülke)entry.Value).nüfus \* 11 / 10);

}

}

static public void yazdır(Hashtable tablo ) //Tablodan çekerek nesneleri yazdıran fonksiyon...

{

Ülke ülkeler = new Ülke();

foreach (DictionaryEntry entry in tablo)

{

ülkeler = (Ülke)entry.Value; // Value tipinin ülke olduğunu beli ettim çünkü diger türlü tek birşey gibi algılıyor...

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

Console.WriteLine("{0,-15}{1,-10}{2,-10}", entry.Key, ülkeler.ülke\_sırası, ülkeler.nüfus);

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Ülke // Ülke sınıfı oluşturulur...

{

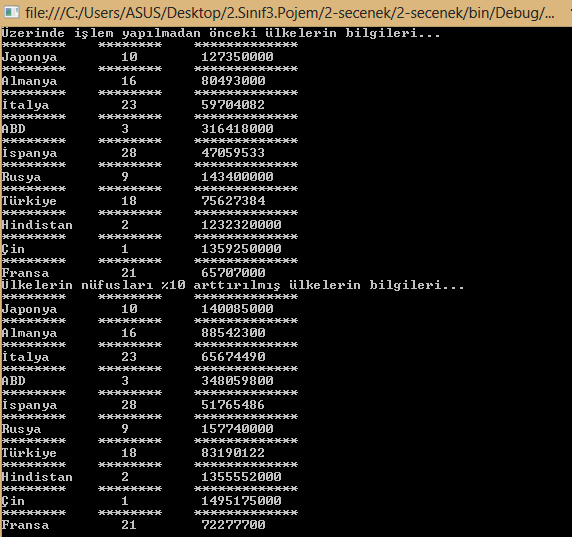
public string ülke\_adı;

public int ülke\_sırası;

public int nüfus;

}

2.Programın ekran çıktısı



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **PROGRAM**

class Program

{

static void Main(string[] args)

{

int[] nüfuslar = { 127350000, 316418000, 65707000, 1359250000, 1232320000, 59704082, 143400000, 80493000, 75627384, 47059533 };

string[] ülke\_adları = { "Japonya", "ABD", "Fransa", "Çin", "Hindistan", "İtalya", "Rusya", "Almanya", "Türkiye", "İspanya" };

int[] sıra = { 10, 3, 21, 1, 2, 23, 9, 16, 18, 28 }; //en kalabalık ülkeler arasında dünyadaki sırasını belirtir

Heap heap = new Heap(nüfuslar.Length);

Ülke ülk = new Ülke();

//int değer;

for (int i = 0; i < ülke\_adları.Length; i++) //10 tane ülke nesnesi oluşturulup heap'e yerleştirilirler...

{

Ülke nesne = new Ülke();

nesne.ülke\_adı = ülke\_adları[i];

nesne.ülke\_sırası = sıra[i];

nesne.nüfus =nüfuslar[i];

heap.insert(nesne);

}

Console.WriteLine("Dünyada ki en kalabalık on ülke içerisinde yer alanlar :");

if (!heap.isEmpty()) // heap'in boş olup olmadıgını kontrol ediyor...

{

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

{

ülk = heap.remove().getKey();

if (ülk.ülke\_sırası <= 10) // Dünyada en kalabalık 10 ülke içerisinde olanları heap'den çekip yazdırdım...

{

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

Console.WriteLine("{0,-15}{1,-10}{2,-10}",ülk.ülke\_adı,ülk.ülke\_sırası,ülk.nüfus);

}

}

}

Console.ReadLine();

}

}

class Ülke // Ülke sınıfı oluşturulur...

{

public string ülke\_adı;

public int ülke\_sırası;

public int nüfus;

}

class Node

{

private Ülke iData;// data item (key)

// -------------------------------------------------------------

public Node(Ülke key) // constructor

{ iData = key; }

// -------------------------------------------------------------

public Ülke getKey()

{ return iData; }

// -------------------------------------------------------------

public void setKey(Ülke id)

{ iData = id; }

// -------------------------------------------------------------

}

class Heap

{

private Node[] heapArray;

private int maxSize; // size of array

private int currentSize; // number of nodes in array

public Heap(int mx) // constructor

{

maxSize = mx;

currentSize = 0;

heapArray = new Node[maxSize];

}

public bool isEmpty()

{

return currentSize == 0;

}

// -------------------------------------------------------------

public bool insert(Ülke 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().nüfus < bottom.getKey().nüfus)

{

heapArray[index] = heapArray[parent]; // move it down

index = parent;

parent = (parent - 1) / 2;

} // end while

heapArray[index] = bottom;

} // end trickleUp()

public Node remove() // delete item with max key

{ // (assumes non-empty list)

Node root = heapArray[0];

heapArray[0] = heapArray[--currentSize];

trickleDown(0);

return root;

}

public void trickleDown(int index)

{

int largerChild;

Node top = heapArray[index]; // save root

while (index < currentSize / 2) // while node has at

{ // least one child,

int leftChild = 2 \* index + 1;

int rightChild = leftChild + 1;

if (rightChild < currentSize && heapArray[leftChild].getKey().nüfus < heapArray[rightChild].getKey().nüfus)

largerChild = rightChild;

else

largerChild = leftChild;

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

break;

heapArray[index] = heapArray[largerChild];

index = largerChild;

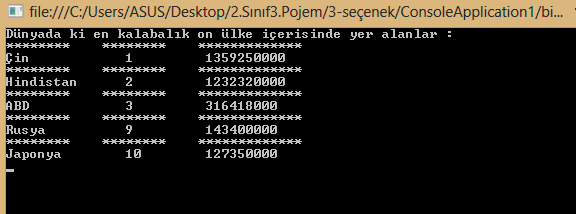
} // end while

heapArray[index] = top; // root to index

}

}

3.Programın ekran çıktısı



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**4-a) PROGRAM**

class Program

{

static void Main(string[] args)

{

}

}

class ArraySel

{

private long[] a; // ref to array a

private int nElems;

public ArraySel(int max) // constructor

{

a = new long[max]; // create the array

nElems = 0;

}

//--------------------------------------------------------------

public void insert(long value) // put element into array

{

a[nElems] = value; // insert it

nElems++; // increment size

}

//--------------------------------------------------------------

public void display() // displays array contents

{

for (int j = 0; j < nElems; j++) // for each element,

Console.Write(a[j] + " "); // display it

Console.WriteLine("");

}

//--------------------------------------------------------------

public void selectionSort()

{

int cikan, giren, min;

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

{

min = cikan;

for (giren = cikan + 1; giren < nElems; giren++)

if (a[giren] < a[min])

min = giren;

swap(cikan, min);

}

}

//--------------------------------------------------------------

private void swap(int one, int two)

{

long temp = a[one];

a[one] = a[two];

a[two] = temp;

}

//--------------------------------------------------------------

}

4-a).Programın ekran çıktısı



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**4-b) PROGRAM**

class Program

{

static void Main(String[] args)

{

}

}

class ArrayIns

{

private long[] array;

private int index;

public ArrayIns(int max)

{

array = new long[max];

index = 0;

}

public void insert(long value)

{

array[index] = value;

index++;

}

public void yazdir()

{

Console.WriteLine("A=");

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

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

Console.WriteLine("");

}

public void quickSort()

{

recQuickSort(0, index - 1);

}

public void recQuickSort(int sol, int sag)

{

if (sag - sol <= 0)

return;

else

{

long on = array[sag];

int parcalama;

parcalama = bolme(sol, sag, (int)on);

recQuickSort(sol, parcalama - 1);

recQuickSort(parcalama + 1, sag);

}

}

public int bolme(int sol, int sag, int on)

{

int solPtr = sol - 1;

int sagPtr = sag;

while (true)

{

while (array[++solPtr] < on)

;

while (sagPtr > 0 && array[--sagPtr] > on)

;

if (solPtr >= sagPtr)

break;

else swap(solPtr, sagPtr);

}

swap(solPtr, sag);

return solPtr;

}

public void swap(int deger1, int deger2)

{

long gecici = array[deger1];

array[deger1] = array[deger2];

array[deger2] = gecici;

}

}

4-b).Programın ekran çıktısı



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**4-c) PROGRAM**

**YORUM:** 4-a bölümünde selection sort sıralama algoritmasını kullandım. Bu algoritmanın zaman karmaşıklığı O(n2) ‘ dir ve örnek olarak verdiğim sayılarda n=10 ise zaman karmaşıklığı 100 olarak bulunur. 4-b bölümünde ise quick sort sıralama algoritmasını kullandım. Bu algoritmanın zaman karmaşıklığı O(n log2n) ‘ dir ve en kötü durumda O(n2)’ dir. n=10 alırsak en kötü durumda zaman karmaşıklığı 100 olacakdır. Bu yüzden quick sort zaman karmaşıklığı selection sort’a göre bazı durumlarda daha iyidir.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\* NOT:** Projenin 1-b kısmında pratik yöntem olarak int tipinde bir sayı değişkenini Ülke sınıfında tuttum ve böylece her nesneyi oluşturduğumda sayı değişkenini de nesne ile arttırmış oldum. Ve daha sora nüfus dizisinden sayı değerine göre çekilip nüfus güncellendi.