



Recursive Fonksiyonlar ve Recursive Fonksiyonların Analizi

08.10.2006

Her şeyden önce bu benim nerdeyse ciddi ciddi oturup yazdığım ilk yazı. Bu yüzden, yok İngilizce yok Türkçe, arada kaynayacak anlamadığınız bir şey olursa artık yapacak bir şey yok.

Her şeyden önce herhangi bir programlama dilindeki fonksiyonların normal lise matematiğindeki fonksiyonlardan hiç bir farkı yoktur. Fonksiyonun tanımını bir kutu olarak düşünürsek bu kutudan çıkacak değerler ya da o kutuda inputlara uygulanacak metotlar kutunun içine yani fonksiyona bakar. Bir iterative fonksiyonu normal $f(x) = 8x$ gibi bir fonksiyon olarak tanımlayabiliriz.

Bir kısa örnek görelim:

```
public int ItFunction(int x)
{
    return x*8;
}
```

Burda normal bir algoritma analizi yaparsak görüldüğü gibi $\Theta(1)$ dir.

Lisedeyken de karşımıza genelde matematik sınavlarında çıkan soru tarzı ise

$$f(x) = f(x - 1) + 1 :$$

$f(2) = 1$ ise $f(10) = ?$ gibi sorular karşımıza çıkmıştır.

İşte bu tip fonksiyonların bilgisayar programlamasında adı ise recursive fonksiyonlardır yani bir fonksiyon kendini çağırıyorsa recursive olur (Recurssion for dummies gibi oldu ama ne yapalım herkes anlasın)

Recursive fonksiyonların kodlarını vermeden önce biraz complexity sinden bahsedelim yani algoritma analizi yapalım. Her yerde olan ve merge sort üzerinden anlatılan recursive fonksiyonları ben Fibonacci sayıları üzerinden anlatmayı tercih edeceğim.

Fibonacci sayılar bir sayı kendinden önce gelen iki sayının toplamıdır diye tanımlanır

O zaman $Fib(x) = Fib(x - 1) + Fib(x - 2)$ şeklinde yazabiliriz

Recursive fonksiyonların analizini yapmak 3 tane yöntem vardır Bunlar substitution , recurssion tree ve master methodlardır. En kolay ve gözde hemen canlanacak olan recursion tree olduğu için hemen bu fonksiyonun recursion tree sini yapalım.


```

using System;
namespace IntroSample4
{
    class Class3
    {
        static void Main(string[] args)
        {
            BinaryTree t = new BinaryTree();

            for(int i=0;i<20;i++)
            {
                t.Add(i );
            }

            t.Show();
            Console.WriteLine("The height of the Tree is {0}",t.GetHeight());
        }
    }

    class Node
    {
        private Node leftNode;
        private Node rightNode;
        private int data;

        public Node(int data)
        {
            this.data = data;
            this.rightNode = null;
            this.leftNode = null;
        }

        public int Data
        {
            get {return this.data;}
        }

        public Node LeftNode
        {
            get {return this.leftNode;}
            set {this.leftNode = value;}
        }

        public Node RightNode
        {
            get {return this.rightNode;}
            set {this.rightNode = value;}
        }
    }

    class BinaryTree
    {

```

```

private Node root;

public BinaryTree()
{
    root = null;
}

public void Add(int x)
{
    Node newNode = new Node(x);

    //check root
    if(root == null)
    {
        root = newNode;
        return;
    }

    Node temp = root;

    while(true)
    {
        if(newNode.Data > temp.Data)
        {
            if(temp.RightNode != null)
            {
                temp = temp.RightNode;
                continue;
            }
            else
            {
                temp.RightNode = newNode;
                return;
            }
        }
        else
        {
            if(temp.LeftNode != null)
            {
                temp = temp.LeftNode;
                continue;
            }
            else
            {
                temp.LeftNode = newNode;
                return;
            }
        }
    }
}

private void show(Node n)
{
    if(n != null)
    {

```

```

        this.show(n.LeftNode);
        this.show(n.RightNode);
        Console.WriteLine(n.Data);
    }
}

public void Show()
{
    this.show(root);
}

private int getHeight(Node n)
{
    //int height = 0;
    int rHeight = 0;
    int lHeight = 0;

    if(n.LeftNode != null)
        lHeight = 1 + getHeight(n.LeftNode);

    if(n.RightNode != null)
        rHeight = 1 + getHeight(n.RightNode);

    return (rHeight > lHeight) ? rHeight : lHeight;
}

public int GetHeight()
{
    if(root == null)
        return 0;
    else
        return this.getHeight(root);
}
}
}

```

Kodların hepsi bana aittir ama sadece yukarıdaki resimleri bir üniversitenin slaytlarından koydum onu paintte yapamayacağım için flash gibi şeyleri de bilmediğim için hemen bir referans koydum neyse ki.

Referanslar:

<http://www.cse.unr.edu/~monica/Courses/CS477-677/Lectures/lecture3.ppt> ppt

Introduction to Algorithms Thomas H. Cormen , Charles E.Leiserson, Ronald L. Rivest, Clifford Stein
MIT Press