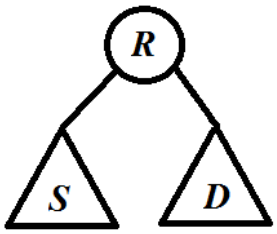
Parcurgerea arborilor binari şi aplicaţii

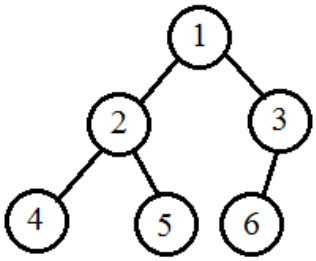
Un ***arbore binar*** este un arbore în care fiecare nod are gradul cel mult 2, adică fiecare nod are cel mult 2 fii. Arborii binari au şi o definiţie recursivă : - un arbore binar este fie vid, fie format dintr-o ***rădăcină R*** şi doi subarbori, numiţi ***subarbore stâng S*** şi respectiv ***subarbore drept D***.



Se face întotdeauna o distincţie clară între cei doi subarbori. Dacă subarborele stâng S este nevid, rădăcina lui se numeşte fiul stâng al rădăcinii R. Analog, dacă subarborele drept D este nevid, rădăcina lui este fiul drept al rădăcinii R.

Parcurgerile arborilor binari sunt cele mai frecvente operaţii utilizate pe arbori. *Parcurgerea* unui arbore înseamnă vizitarea fiecărui nod al arborelui o singură dată, cu scopul prelucrării informaţiei memorate în acel nod. Dintre cele mai utilizate parcurgeri sunt parcurgerile în adâncime (*algoritmul* **DFS**) şi pe niveluri (*algoritmul* **BFS**).

A. Parcurgeri în adâncime sunt parcurgerile în preordine, inordine şi postordine. În toate cele trei tipuri de parcurgere în adâncime se vizitează prima dată subarborele stâng şi apoi subarborele drept, iar diferenţa constă în momentul în care se vizitează rădăcina. Funcţiile de parcurgere pot fi scrise recursiv sau iterativ, folosind o stivă.



n	1	2	3	4	5	6
S	2	4	6	0	0	0
D	3	5	0	0	0	0

Parcurgerea în preordine RSD – se vizitează mai întâi rădăcina ***R***, apoi se parcurge în preordine subarborele stâng ***S*** şi subarborele drept ***D***. *Pentru arborele din figură, şirul parcurgerii **RSD** este:* **1 2 4 5 3 6**.

Parcurgerea în inordine SRD – se parcurge mai întâi în inordine suarborele stâng ***S***, apoi rădăcina ***R***, apoi se parcurge în inordine subarborele drept ***D***. *Pentru arborele din figură, şirul parcurgerii **SRD** este:* **4 2 5 1 6 3**.

Parcurgerea în postordine SDR – se parcurg în postordine subarborele stâng ***S*** şi subarborele drept ***D***, iar apoi se vizitează rădăcina ***R***. *Pentru arborele din figură, şirul parcurgerii **RSD** este:* **4 5 2 6 3 1**.

B. Parcurgerea pe niveluri – se vizitează rădăcina, apoi toţi fii nodului rădăcină, de la stânga spre dreapta şi se continuă în acest mod pe toate nivelurile. *Pentru arborele din figură, şirul parcurgerii pe niveluri este:* **1 2 3 4 5 6**.

Parcurgerile arborilor binari

1. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Afişaţi pe ecran şirurile parcurgerilor RSD, SRD şi SDR ale arborelui citit.

arbore.in	*rezultate*
8 1	**RSD:** 1 2 4 5 6 3 7 8
2 4 7 0 0 0 0 0	**SRD:** 4 2 5 6 1 7 3 8
3 5 8 0 6 0 0 0	**SDR:** 4 6 5 2 7 8 3 1

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101];
void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}
void RSD(int p)
{
    if(p>0)

    {   cout<<p<<' ';
        RSD(St[p]);
        RSD(Dr[p]);
    }
}
void SRD(int p)
{
    if(p>0)
    {
        SRD(St[p]);
        cout<<p<<' ';
        SRD(Dr[p]);
    }

}
void SDR(int p)
{
    if(p>0)
    {SDR(St[p]);
    SDR(Dr[p]);
    cout<<p<<' ';}

}
int main()
{
    citire();
    cout<<"RSD: ";
    RSD(R);
    cout<<endl;
    cout<<"SRD: ";
    SRD(R);
    cout<<endl;
    cout<<"SDR: ";
    SDR(R);
}
```

Vectorul de taţi

2. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Afişaţi pe ecran vectorul de taţi al arborelui citit.

arbore.in	*rezultate*
8 1 2 4 7 0 0 0 0 0 3 5 8 0 6 0 0 0	8 1 0 1 1 2 2 5 3 3

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101],T[101];
void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

void tati(int x,int p)
{
    if(x>0)
    {
        T[x]=p;
        tati(St[x],x);
        tati(Dr[x],x);
    }
}

void tipar()
{
    int i;
    cout<<n<<' '<<R<<'\n';
    for(i=1;i<=n;i++)
        cout<<T[i]<<' ';
}

int main()
{
    citire();
    tati(R,0);
    tipar();
}
```

Lista nodurilorarborelui

3. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Afişaţi pe ecran lista nodurilor terminale, lista nodurilor cu un singur fiu şi lista nodurilor care au exact doi fii.

arbore.in	*rezultate*
8 1	Noduri terminale (frunze): 4 6 7 8
2 4 7 0 0 0 0 0	Noduri cu un fiu: 5
3 5 8 0 6 0 0 0	Noduri cu doi fii: 1 2 3

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101];
void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

void fii(int p,int k)
{
    int f=0;
    if(p>0)
    {
        if(St[p]>0) f++;
        if(Dr[p]>0) f++;
        if(f==k) cout<<p<<' ';
        fii(St[p],k);
        fii(Dr[p],k);
    }
}

int main()
{
    citire();
    cout<<"Noduri terminale (frunze): ";
    fii(R,0);
    cout<<'\n';

    cout<<"Noduri cu un fiu: ";
    fii(R,1);
    cout<<'\n';

    cout<<"Noduri cu doi fii: ";
    fii(R,2);
}
```

Numărul nodurilor

4. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte **n** elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau **0** dacă nu există subarborele.

Afişaţi pe ecran numărul nodurilor terninale, numărul nodurilor cu un singur fiu şi numărul nodurilor care au exact doi fii.

arbore.in	*rezultate*
8 1	Noduri terminale (frunze): 4
2 4 7 0 0 0 0 0	Noduri cu un fiu: 1
3 5 8 0 6 0 0 0	Noduri cu doi fii: 3

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101];
void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

int fii(int p,int k)
{
    int f=0;
    if(p>0)
    {
        if(St[p]>0) f++;
        if(Dr[p]>0) f++;
        if(f==k)
            return 1+fii(St[p],k)+fii(Dr[p],k);
        else
            return fii(St[p],k)+fii(Dr[p],k);
    }
}

int main()
{
    citire();
    cout<<"Noduri terminale (frunze): "<<fii(R,0);
    cout<<'\n';

    cout<<"Noduri cu un fiu: "<<fii(R,1);
    cout<<'\n';

    cout<<"Noduri cu doi fii: "<<fii(R,2);
}
```

Înălţimea unui arbore binar

5. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Afişaţi înălţimea arborelui binar citit.

arbore.in	*rezultate*
8 1 2 4 7 0 0 0 0 0 3 5 8 0 6 0 0 0	Inaltimea arborelui: 3

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101];

void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

int h(int p, int k)
{
    int a,b;
    if((St[p]==0)&&(Dr[p]==0)) return k;
    else
    {
        a=h(St[p],k+1);
        b=h(Dr[p],k+1);
        if(a>b) return a;
        else return b;
    }
}

int main()
{
    citire();
    cout<<"Inaltimea arborelui: "<<h(R,0);
}
```

Arbore binar strict

6. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Verificaţi dacă arborele citit este un arbore binar strict, adică fiecare nod are 0 sau 2 noduri fii.

arbore.in	*rezultate*
8 1 2 4 7 0 0 0 0 0 3 5 8 0 6 0 0 0	Nu este arbore binar strict

```
#include<fstream>
#include<iostream>
using namespace std;
int i,n,St[101],R,Dr[101];
void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

int fii(int p,int k)
{
    int f=0;
    if(p>0)
    {
        if(St[p]>0) f++;
        if(Dr[p]>0) f++;
        if(f==k)
            return 1+fii(St[p],k)+fii(Dr[p],k);
        else
            return fii(St[p],k)+fii(Dr[p],k);
    }
    else return 0;
}

int main()
{
    citire();
    if(fii(R,1)==0)
        cout<<"Arbore binar strict";
    else
        cout<<"Nu este arbore binar strict";
}
```

Arbore echilibrat

7. Se citesc din fisierul **arbore.in** de pe prima linie două numere **n** şi **R** reprezentând numărul de noduri şi rădăcina unui arbore binar. De pe a doua şi a treia linie se vor citi câte n elemente reprezentând rădăcinile subarborilor stâng şi respectiv drept al fiecărui nod, sau 0 dacă nu există subarborele.

Verificaţi dacă arborele citit este un arbore binar echilibrat. Un arbore este ***echilibrat*** dacă diferenţa dintre înălţimea oricăror doi subarbori este 0 sau 1.

arbore.in	*rezultate*
8 1 2 4 7 0 0 0 0 0 3 5 8 0 6 0 0 0	Arborele este echilibrat

```
#include<fstream>
#include<iostream>
#include<cmath>
using namespace std;
int i,n,St[101],R,Dr[101],ok=1;

void citire()
{
    ifstream f("arbore.in");
    f>>n>>R;
    for(i=1;i<=n;i++) f>>St[i];
    for(i=1;i<=n;i++) f>>Dr[i];
    f.close();
}

int h(int p, int k)
{
    int a,b;
    if((St[p]==0)&&(Dr[p]==0)) return k;
    else
    {
        a=h(St[p],k+1);
        b=h(Dr[p],k+1);
        if(abs(a-b)>1) ok=0;
        if(a>b) return a;
        else return b;
    }
}

int main()
{
    citire();
    h(R,0);
    if(ok==1)
        cout<<"Arborele este echilibrat";
    else
        cout<<"Arborele nu este echilibrat";
}
```