INFORMATICĂ · CLASA A XI-A · GRAFURI · LECȚIA 13

Arbori: definiție, arbori cu rădăcină, vectorul de tați

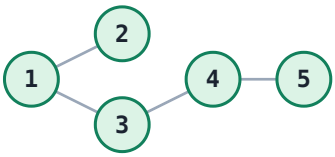
Durată: 60 min Nivel: mate-info Înainte: lecțiile xi-10

CE VEI ȘTI DUPĂ LECȚIE

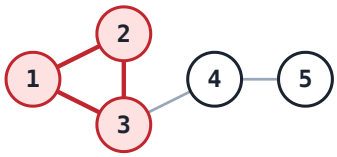
- Știi definiția arborelui (graf conex fără cicluri) și proprietățile lui (n − 1 muchii).
- Folosești terminologia arborilor cu rădăcină: rădăcină, tată, fiu, frate, frunză, nivel, înălțime.
- Reprezinți un arbore cu rădăcină prin vectorul de tați și îl folosești în programe.
- Cunoști noțiunile de bază despre arbori binari și parcurgerile lor.

1. Ce este un arbore

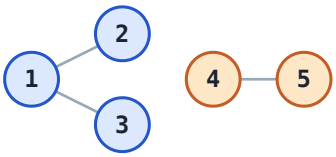
Un **arbore** este un graf neorientat **conex** și **fără cicluri**: între oricare două vârfuri există **exact un** lanț. Exemple din viață: arborele genealogic, structura de foldere dintr-un calculator, ierarhia unei firme, meniurile unei aplicații.



arbore: conex, 4 muchii = n − 1



NU: are ciclu (5 muchii)



NU: neconex (3 muchii)

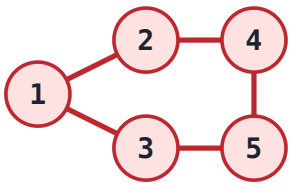
Arbore sau nu

LA BAC

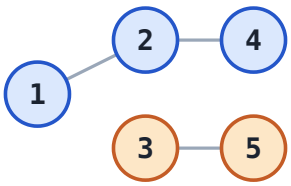
Pentru un graf cu n vârfuri, următoarele afirmații sunt **echivalente**: - G este arbore; - G este conex și are **n − 1 muchii**; - G nu are cicluri și are **n − 1 muchii**; - între oricare două vârfuri există un **singur** lanț elementar.

Consecințe: un arbore are cel puțin **două frunze** (vârfuri de grad 1); o **pădure** (mai mulți arbori) cu n vârfuri și p componente are **n − p** muchii.

De ce n − 1: pornești cu n vârfuri izolate (n componente). Fiecare muchie unește două componente (altfel apare un ciclu), deci scade numărul de componente cu 1. Pentru a ajunge la 1 componentă ai nevoie de exact n − 1 muchii.



adaug [4,5]: apare exact un ciclu



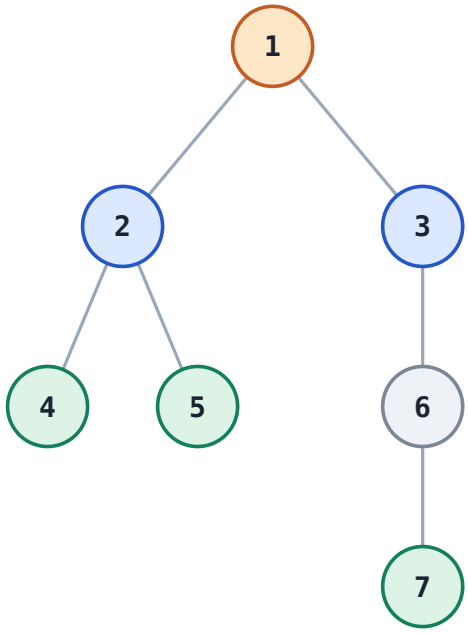
elimin [1,3]: exact două componente

Adaug sau elimin o muchie

- dacă **adaugi** o muchie într-un arbore, apare **exact un ciclu**;
- dacă **elimini** o muchie dintr-un arbore, rămân **exact două** componente (două arbori).

2. Arbori cu rădăcină

Alegem un vârf ca **rădăcină** și „agățăm" arborele de el, cu rădăcina sus. Apar relații de familie:

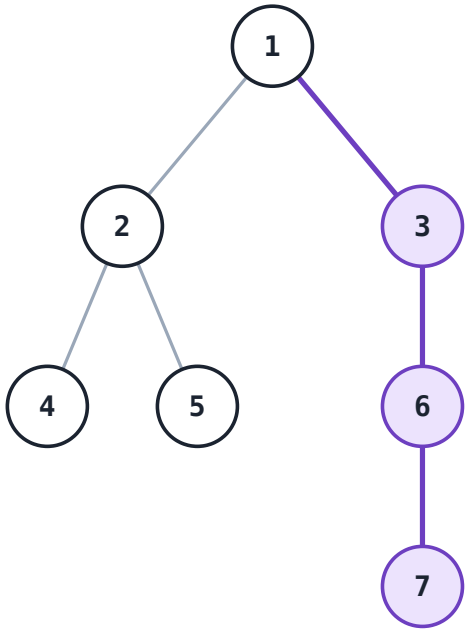


portocaliu: rădăcină · verde: frunze (4, 5, 7) · niveluri: 0 → 1 · 1 → 2, 3 · 2 → 4, 5, 6 · 3 → 7

Arbore cu rădăcină

Termen	Definiție	În figură
tată al lui x	vecinul lui x de pe lanțul spre rădăcină	tatăl lui 4 este 2
fii ai lui x	vecinii de „dedesubt"	fiii lui 2: 4, 5
frați	au același tată	4 și 5; 2 și 3
frunză	nu are fii	4, 5, 7
ascendenți / descendenți	strămoși / urmași	ascendenții lui 7: 6, 3, 1

Termen	Definiție	În figură
nivel al lui x	lungimea lanțului de la rădăcină la x (rădăcina: 0)	nivelul lui 6 este 2
înălțime	nivelul maxim	3
subarbore al lui x	x și toți descendenții lui	subarborele lui 3 = {3, 6, 7}



drumul de la 7 la rădăcină: 7 → 6 → 3 → 1 (urcăm cu t[]) · nivelul lui 7 = 3

Subarbore

3. Vectorul de tați

`t[x]` = tatăl lui x, iar pentru rădăcină `t[rad] = 0`. Este reprezentarea **cea mai folosită la Bac**.

	1	2	3	4	5	6	7
vârful	1	2	3	4	5	6	7
t[vârf]	**0**	1	1	2	2	3	6

rădăcina are t = 0 · frunzele (4, 5, 7) NU apar printre valori · de câte ori apare k = câți fii are k

Vectorul de tați

Cum „citești" vectorul `t = (0, 1, 1, 2, 2, 3, 6)`:

Întrebare	Cum o afli	Răspuns
rădăcina	vârful cu `t = 0`	1

Întrebare	Cum o afli	Răspuns
frunzele	vârfurile care **nu** apar ca valoare în `t`	4, 5, 7
fiii lui 2	indicii i cu `t[i] = 2`	4, 5
numărul de fii ai lui 1	de câte ori apare 1 în `t`	2
frații lui 2	alt i cu același `t`	3
nivelul lui 7	de câte ori urci până la rădăcină	3
drumul de la 7 la rădăcină	7, t[7] = 6, t[6] = 3, t[3] = 1	7 6 3 1

SFAT

Un vector de numere este vector de tați **valid** dacă: există exact o valoare 0 (rădăcina), toate celelalte valori sunt între 1 și n, și urcând din orice vârf ajungi la rădăcină (nu există ciclu). Verificarea cu „exact un 0" se cere des la grile (de exemplu: „care dintre vectori NU poate fi vector de tați").

Tot ce se află dintr-un vector de tați

Programul citește `n` și vectorul `t`, apoi afișează rădăcina, frunzele, nivelurile tuturor vârfurilor și înălțimea.

```cpp
#include <iostream>
using namespace std;

int t[101], n, areFii[101], niv[101];

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> t[i];

    for (int i = 1; i <= n; i++)
        if (t[i] == 0) cout << i << '\n';                // rădăcina

    for (int i = 1; i <= n; i++)
        if (t[i]) areFii[t[i]] = 1;                      // t[i] are cel puțin un fiu
    for (int i = 1; i <= n; i++)
        if (!areFii[i]) cout << i << ' ';                // frunzele
    cout << '\n';

    int inaltime = 0;
    for (int i = 1; i <= n; i++)
    {
        int nivel = 0, y = i;
        while (t[y] != 0) { y = t[y]; nivel++; }         // urcăm spre rădăcină
        niv[i] = nivel;
        if (nivel > inaltime) inaltime = nivel;
        cout << nivel << ' ';
    }
    cout << '\n' << inaltime;
    return 0;
}
```

DE CE AȘA?

De ce frunzele sunt cele care nu apar în `t`? Un vârf apare în `t[...]` doar ca tată al cuiva. Dacă nu apare nicăieri, nu are fii: frunză. **De ce `while (t[y] != 0)`?** Rădăcina are `t = 0`; fiecare pas în sus este un nivel. Pentru x = 7: 7 → 6 → 3 → 1, adică 3 pași.

Trace pentru nivelul vârfului 7 (`t = 0 1 1 2 2 3 6`):

Pas	y	t[y]	nivel
început	7	6	0
1	6	3	1
2	3	1	2
3	1	0	3 (oprire, t[1] = 0)

Vectorul complet al nivelurilor este `0 1 1 2 2 2 3` (vârful 6 are tatăl 3, iar 3 are tatăl 1: nivelul 2). Același lucru, **recursiv**, urmează chiar definiția: nivelul unui vârf este nivelul tatălui plus 1, iar rădăcina are nivelul 0.

```cpp
int nivel(int x)
{
    if (t[x] == 0) return 0;
    return 1 + nivel(t[x]);
}
```

4. Din muchii în vector de tați

La Bac și pe pbinfo, arborele se dă adesea prin **muchii + rădăcină**. Facem un DFS din rădăcină și, când coborâm din x în y, scriem `t[y] = x`.

```cpp
#include <iostream>
using namespace std;

int a[101][101], t[101], n, rad;

void dfs(int x)
{
    for (int y = 1; y <= n; y++)
        if (a[x][y] && y != t[x])         // vecin care nu e tatăl: e fiu
        {
            t[y] = x;
            dfs(y);
        }
}

int main()
{
    cin >> n >> rad;
    for (int k = 1; k < n; k++)           // un arbore are n-1 muchii
    {
        int x, y;
        cin >> x >> y;
        a[x][y] = a[y][x] = 1;
    }
    dfs(rad);
    for (int i = 1; i <= n; i++) cout << t[i] << ' ';
    return 0;
}
```

DE CE AȘA?

De ce nu avem nevoie de `viz[]`? Într-un arbore nu există cicluri; singurul vecin deja „văzut” este tatăl. De aceea `y != t[x]` înlocuiește marcajul. **De ce `t[rad] = 0` fără să scriem nimic?** Tabloul global este inițializat cu 0.

5. Strămoși, subarbori

Subarborele unui vârf

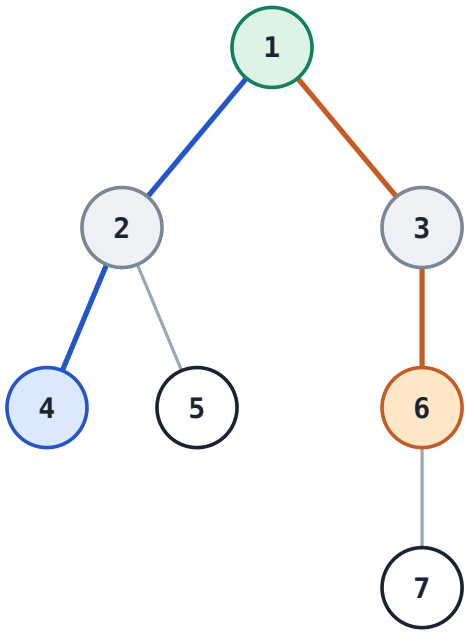
Vârful y este în subarborele lui x dacă urcând din y ajungem la x. Număr de vârfuri din subarbore:

```cpp
#include <iostream>
using namespace std;

int t[101], n;

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> t[i];
    int x;
    cin >> x;
    int nr = 0;
    for (int i = 1; i <= n; i++)
    {
        int y = i;
        while (y != 0 && y != x) y = t[y];     // urc până la x sau până dincolo de rădăcină
        if (y == x) nr++;                      // x este strămoș (sau chiar i)
    }
    cout << nr;
    return 0;
}
```

Cel mai apropiat strămoș comun



urcând din 4: 4, 2, 1 · urcând din 6: 6, 3, 1 · primul comun: 1

Strămoș comun

Marcăm toți strămoșii lui x, apoi urcăm din y până dăm de unul marcat.

```cpp
#include <iostream>
using namespace std;

int t[101], n, strabun[101];

int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> t[i];
    int x, y;
    cin >> x >> y;
    for (int v = x; v != 0; v = t[v]) strabun[v] = 1;     // x și toți strămoșii lui
    int v = y;
    while (!strabun[v]) v = t[v];                         // urcăm din y până la unul marcat
    cout << v;
    return 0;
}
```

DE CE AȘA?

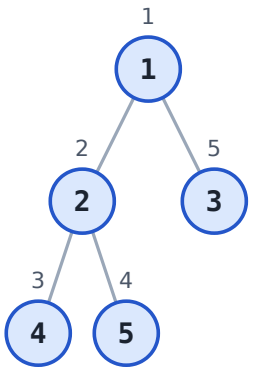
De ce se oprește cu siguranță? Rădăcina este strămoș al oricui, deci este marcată; urcând din y, în cel mai rău caz ajungem la ea. **De ce ne dă „cel mai apropiat”?** Urcăm din y pas cu pas, deci primul vârf marcat întâlnit este cel mai de jos.

Intrare / ieșire

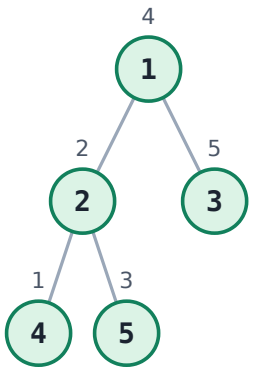
Problemă	Intrare	Ieșire
informații dintr-un vector	`7` / `0 1 1 2 2 3 6`	`1` · `4 5 7` · `0 1 1 2 2 2 3` · `3`
din muchii în tați	`7 1` + `1 2`, `1 3`, `2 4`, `2 5`, `3 6`, `6 7`	`0 1 1 2 2 3 6`
subarbore	același vector + `3`	`3`
strămoș comun	același vector + `4 6`	`1`

6. Arbori binari: o privire de ansamblu

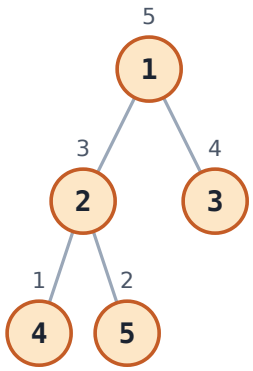
Într-un **arbore binar** fiecare vârf are **cel mult doi** fii: stâng și drept. Se parcurg în trei moduri, după locul rădăcinii: **preordine** (R S D), **inordine** (S R D), **postordine** (S D R).



preordine: R S D → 1 2 4 5 3



inordine: S R D → 4 2 5 1 3



postordine: S D R → 4 5 2 3 1

Parcurgeri

În lecția XI.20 le implementăm cu pointeri și memorie dinamică.

7. Greșeli frecvente

Greșeala	Efect	Corect
numeri nivelul de la 1	totul +1	rădăcina e pe nivelul **0**
înălțime = număr de vârfuri pe cel mai lung drum	cu 1 mai mult	număr de **muchii** (= nivelul maxim)
frunză = vârf cu grad 1 în arbore cu rădăcină	rădăcina cu un singur fiu are și ea grad 1	frunză = nu are fii
uiți `t[rad] = 0` la citire	`while` infinit	verifici că există exact un 0
arbore „cu n muchii”	nu e arbore (are un ciclu)	exact n − 1 muchii
frații = toți cu tată comun, inclusiv el	numeri greșit	frații lui x: vârfurile cu același tată, **fără** x

8. La BAC

LA BAC

Subiectul I conține aproape sigur o grilă despre arbori: „câte frunze / câți fii / care este înălțimea / care este vectorul de tați pentru figura dată / care vector NU poate fi vector de tați / câte muchii are / câți arbori cu n vârfuri”. Pe subiectul cu programare apar „din muchii în vector de tați”, „afișează frunzele”, „afișează nodurile de pe nivelul k”, „drumul între două noduri”. Dacă ai desenul, numerotează nivelurile înainte de a răspunde.

9. Exersează pe hârtie

1. Vectorul de tați este `t = (2, 0, 2, 1, 1, 3, 3, 4)`. Care este rădăcina? Câte frunze sunt? Care este înălțimea?

2. Un arbore are 20 de vârfuri. Câte muchii are? Câte muchii trebuie eliminate din graful complet K_{20} pentru a obține un arbore?

3. Care este numărul minim de frunze al unui arbore cu 10 vârfuri în care un vârf are gradul 4?

4. Poate fi `(0, 1, 1, 2, 2, 4)` un vector de tați? Dar `(0, 1, 1, 2, 2, 0)`? Dar `(2, 3, 2, 1, 0)`?

5. Un arbore cu rădăcină are 15 vârfuri, iar fiecare vârf care nu e frunză are exact 2 fii. Câte frunze are?

6. Scrie vectorul de tați pentru arborele cu rădăcina 3 și muchiile [1,2], [2,3], [3,4], [3,5], [5,6].

7. (BAC) Un arbore cu 10 vârfuri are 6 frunze. Care este numărul maxim de vârfuri cu cel puțin doi fii?

8. Pentru `t = (0, 1, 1, 2, 2, 3, 6)`, scrie nodurile de pe nivelul 2 și strămoșul comun al lui 5 și 7.

9. Câți arbori diferiți (ca grafuri etichetate) există cu 3 vârfuri? (Formula lui Cayley: n^{n-2}.)

10. Scrie un program care citește `n` și vectorul `t` și afișează numărul maxim de fii pe care îl are un vârf.

10. Mai departe (pentru cine vrea mai mult)

Formula lui Cayley. Numărul de arbori etichetați cu n vârfuri este n^{n-2}. Pentru n = 4: 16. Demonstrația folosește **codul Prüfer**: fiecare arbore corespunde unui șir de n − 2 numere.

Diametrul unui arbore. Cel mai lung lanț dintre două vârfuri: faci BFS din orice vârf, ajungi la cel mai îndepărtat u; faci BFS din u și cel mai îndepărtat de acolo (v) dă diametrul = d(u, v).

LCA rapid. Pentru multe întrebări de „strămoș comun", se precalculează strămoșii la distanța 2^0, 2^1, 2^2, ... (binary lifting): O(log n) pe întrebare.

Arborele parțial de cost minim. Dintr-un graf cu costuri alegi n − 1 muchii care formează un arbore și au suma minimă: algoritmii Prim și Kruskal (lecția XI.19).

Rezumat

- Arbore = conex + fără cicluri = conex cu n − 1 muchii = fără cicluri cu n − 1 muchii; cel puțin 2 frunze.
- Arbore cu rădăcină: tată, fii, frați, frunze, nivel (rădăcina = 0), înălțime = nivelul maxim, subarbore.
- Vectorul de tați: `t[rad] = 0`; frunzele nu apar în `t`; nivel = pași până la rădăcină.
- Din muchii: DFS din rădăcină, `t[y] = x`, fără `viz[]` (doar `y != t[x]`).

Probleme de lucru (pbinfo)

Începe cu acestea, în ordine: sunt alese să urmeze pașii din lecție.

- [ ] #636 **arbore** — ușoară
  Din lista muchiilor + rădăcina k → vectorul de tați.
  https://www.pbinfo.ro/probleme/636/arbore

- [ ] #637 **frunze** — ușoară
  https://www.pbinfo.ro/probleme/637/frunze

- [ ] #639 **inaltime** — ușoară
  https://www.pbinfo.ro/probleme/639/inaltime

Temă (pbinfo)

Trimite soluțiile pe pbinfo și notează punctajul obținut. Bifează ce ai terminat.

- [ ] #638 **nivele** — ușoară
  https://www.pbinfo.ro/probleme/638/nivele
- [ ] #640 **nrfii** — ușoară
  https://www.pbinfo.ro/probleme/640/nrfii
- [ ] #641 **subarbore** — ușoară
  https://www.pbinfo.ro/probleme/641/subarbore
- [ ] #642 **detdrum** — ușoară
  https://www.pbinfo.ro/probleme/642/detdrum
- [ ] #650 **knivel** — ușoară
  https://www.pbinfo.ro/probleme/650/knivel
- [ ] #652 **afisarefii** — ușoară
  https://www.pbinfo.ro/probleme/652/afisarefii
- [ ] #644 **detdrum2** — ușoară
  Drumul între două noduri: strămoșul comun.
  https://www.pbinfo.ro/probleme/644/detdrum2

Pentru cei care vor mai mult

Probleme mai grele sau de concurs, pentru cine vrea să meargă dincolo de BAC.

- [ ] #670 **preordine** — ușoară
  https://www.pbinfo.ro/probleme/670/preordine
- [ ] #2749 **tata** — medie
  https://www.pbinfo.ro/probleme/2749/tata

Grile din subiectele de BAC (pbinfo)

Exerciții de tip grilă din variantele de Bacalaureat, rezolvate online pe pbinfo, cu verificare automată a răspunsului.

- [ ] #65 **Exercițiul #65**
  câte frunze are arborele dat prin vectorul de tați
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=65

- [ ] #95 **Exercițiul #95**
  vectorul de tați pentru arborele din figură
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=95
- [ ] #143 **Exercițiul #143**
  numărul de frunze dintr-un arbore cu 8 noduri
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=143
- [ ] #180 **Exercițiul #180**
  care dintre vectori este un vector de tați valid
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=180
- [ ] #207 **Exercițiul #207**
  care vector NU poate fi vector de tați
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=207
- [ ] #364 **Exercițiul #364**
  câte valori nule are un vector de tați
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=364
- [ ] #366 **Exercițiul #366**
  numărul maxim de valori egale într-un vector de tați
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=366
- [ ] #1159 **Exercițiul #1159**
  înălțimea arborelui: lungimea celui mai lung lanț de la rădăcină
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1159
- [ ] #1067 **Exercițiul #1067**
  arbore cu 20 de noduri, 10 frunze: ce poate fi numărul de fii
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=1067
- [ ] #72 **Exercițiul #72**
  arbore cu 11 muchii: câte noduri are
  https://www.pbinfo.ro/?pagina=itemi-evaluare&id=72

Citește și exersează în plus

- Arbori cu rădăcină https://www.pbinfo.ro/articole/5982/arbori-cu-radacina
- Arbori binari https://www.pbinfo.ro/articole/25641/arbori-binari
- Arbore (teoria grafurilor) — definiții echivalente și proprietăți
  https://ro.wikipedia.org/wiki/Arbore_(teoria_grafurilor)
- USACO Guide — Rooted Trees — arbori cu rădăcină, subarbori, strămoși, cu probleme de concurs (engleză) https://usaco.guide/gold/intro-tree
- VisuAlgo — Binary Tree / Traversal — preordine, inordine, postordine animate (engleză)
  https://visualgo.net/en/bst

Răspunsuri

Exersează pe hârtie

1. `t[2] = 0`, deci rădăcina este **2**. Vârfurile care apar ca tată: 2, 1, 3, 4 → frunzele sunt 5, 6, 7, 8: **4 frunze**. Nivelurile: 2 → 0; 1, 3 → 1; 4, 5 (fii ai lui 1), 6, 7 (fii ai lui 3) → 2; 8 (fiu al lui 4) → 3. Înălțimea este **3**.

2. Un arbore cu 20 de vârfuri are **19** muchii. K_{20} are 20 · 19 / 2 = 190 de muchii, deci se elimină 190 − 19 = **171**.

3. Într-un arbore, numărul de frunze este f = 2 + Σ (d − 2), suma luată pe vârfurile cu gradul d ≥ 3. Cu un vârf de grad 4 se obține f ≥ 2 + 2 = **4** frunze (de exemplu un vârf legat de patru lanțuri care însumează celelalte 9 vârfuri).

4. `(0, 1, 1, 2, 2, 4)`: **da** (rădăcina 1; 6 are tatăl 4, 4 are tatăl 2, 2 are tatăl 1). `(0, 1, 1, 2, 2, 0)`: **nu**, are două valori 0 (două rădăcini). `(2, 3, 2, 1, 0)`: **nu**: `t[2] = 3` și `t[3] = 2` formează un ciclu, deci urcând din 2 nu se ajunge niciodată la rădăcina 5.

5. Fie f frunze și k vârfuri interne: f + k = 15 și numărul de muchii = 14 = 2k, deci k = 7 și **f = 8** frunze.

6. Rădăcina 3: `t[3] = 0`; `t[2] = 3`, `t[4] = 3`, `t[5] = 3`, `t[1] = 2`, `t[6] = 5`. Vectorul: **`(2, 3, 0, 3, 3, 5)`**.

7. Cele 6 frunze nu au fii; cei 4 vârfuri interne (rădăcina inclusă) împart cei 9 = n − 1 fii. Fiecare poate avea cel puțin 2 fii: de exemplu 2, 2, 2 și 3 fii (total 9). Maximul este **4**.

8. Nivelul 2: **4, 5, 6**. Strămoșul comun al lui 5 (5 → 2 → 1) și 7 (7 → 6 → 3 → 1): **1**.

9. 3^1 = **3** arbori (lanțurile 1-2-3, 2-1-3, 1-3-2, cu mijlocul diferit).

10. Numeri de câte ori apare fiecare valoare nenulă în `t` și iei maximul:

```
int fii[101] = {0}, mx = 0;
for (int i = 1; i <= n; i++) if (t[i]) fii[t[i]]++;
for (int i = 1; i <= n; i++) if (fii[i] > mx) mx = fii[i];
cout << mx;
```