1. Eliberati spatiul ocupat de un arbore binar

* **Idee solutie recursiva** – traversati arborele in postordine (SDR) si eliberati spatiul pentru nodurile din stanga si dreapta inainte de a elibera spatiul ocupat de radacina. Se elibereaza mai intai spatiul  ocupat de nodurile cele mai din stanga jos, dreapta jos...
* **Idee solutie iterativa** – parcurgeti arborele pe niveluri. Ideea e sa stergeti fiecare nod din coada dupa ce ati adaugat copii acelui nod in coada pentru a fi procesati. Se elibereaza mai intai spatiul  ocupat de nodurile cele mai de sus, tinand minte adresele copiilor.

2. Stergeti un subarbore dintr-un arbore. Radacina sub-arborelui e data prin adresa / valoarea sa (caz in care considerati ca valorile stocate sunt diferite).

3. Verificati ca un nod (dat prin valoarea stocata) se gaseste intr-un arbore binar. Numarati de cate ori este stocata aceeasi valoare in arbore. Stergeti nodurile care contin informatie duplicat astfel incat arborele sa ramana intr-o forma cat mai echilibrata.

4. Afisati toti stramosii unui nod. Folosesc parcurgerea in postordine. Verific pentru fiecare nod parcurs, daca nodul cautat se gaseste in subarborele stang sau drept si, daca da, ii afisez valoarea.

5. Gasiti cel mai apropiat stramos comun  a doua noduri (Lowest Common Ancestor - LCA).

LCA a doua noduri a si b este cel mai adanc (de jos nod) care ii are pe a si b ca descendenti.

LCA pentru 6 si 1 e 8.

* **Idee solutie iterativa** -  se salveaza caile de la  radacina la a si de la radacina la b in 2 vectori.

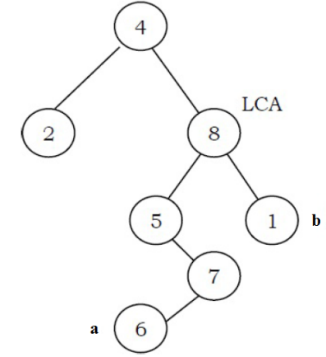
{4, 8, 5, 7, 6}

{4, 8, 1}

Apoi se parcurg vectorii pana cand se gasesc 2 valori diferite. Ultima valoare egala era cea a LCA.

* **Idee solutie recursiva** – trebuie gasit nodul din arbore care il are pe a intr-un subarbore si pe b in celalalt subarbore.  Se foloseste parcurgerea in postordine.

\*nodurile a si b sunt date prin adresele lor



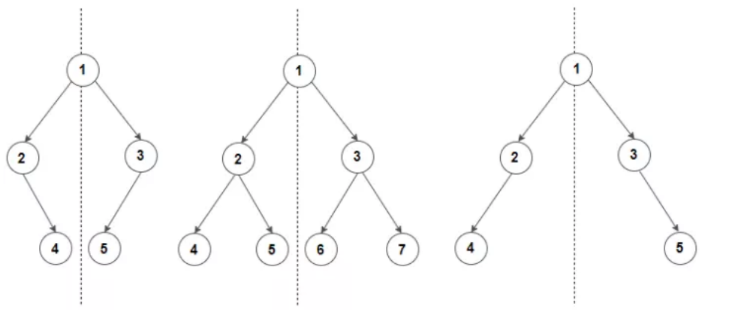
6. Verificati daca un un arbore binar este simetric (ca dispunere a nodurilor, nu valori ale lor)

Idee – arborele e simetric daca subarborii drept si stang sunt imagini in oglinda:

- amandoi subarborii exista sau nu

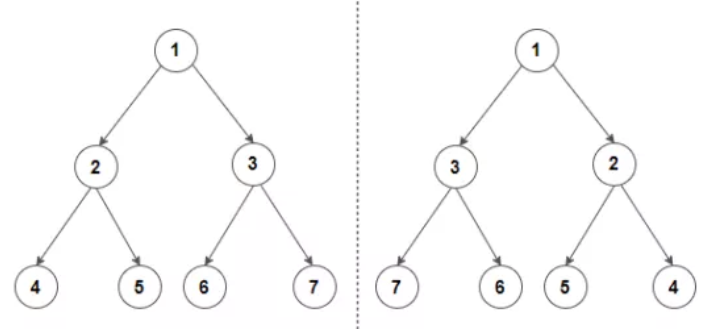
- subarborele stang arata ca cel drept in oglinda

- subarborele drept arata ca cel stang in oglinda



7. Convertiti un arbore binar in versiunea sa in oglinda.

Idee – parcurgeti arborele in postordine (SDR) si pentru fiecare nod inversati copii – stangul devine dreptul si dreptul stangul



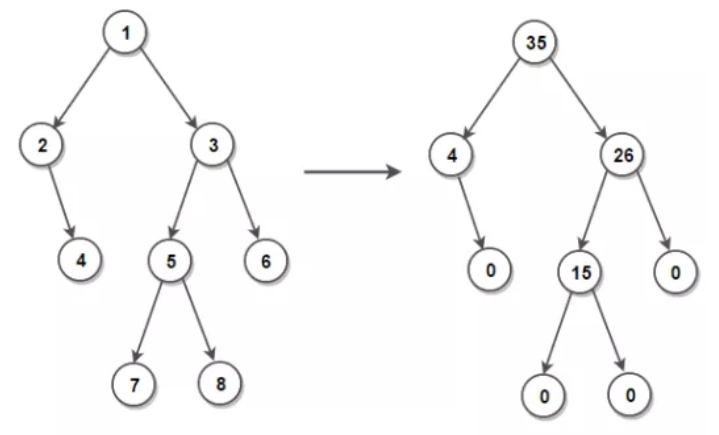
8. Transformati un arbore in arborele suma echivalent.

In arborele suma valoarea fiecarui element care nu e frunza este suma elementelor din subarborii drept si stang.

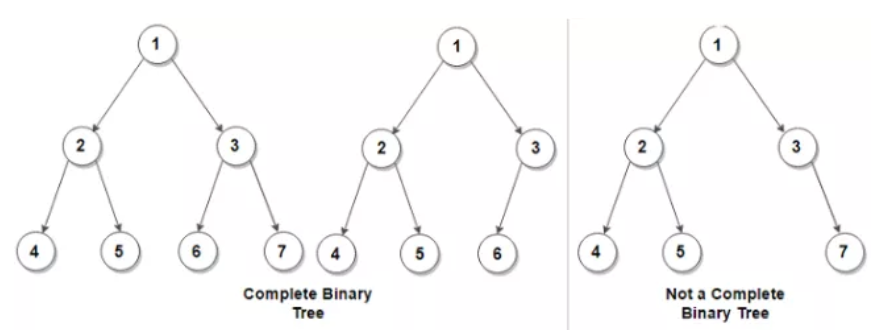
**Idee** – parcurgem arborele in postordine. Pentru fiecare nod care nu e frunza tinem minte valoarea veche si stocam in locul ei suma elementelor din subarborii drept si stang. Returnam suma subarborilor+valoarea veche.

9. Verificati daca un arbore este un arbore suma sau nu.

**Idee** – parcurgem arborele in postordine. Pentru fiecare nod care nu e frunza verificam daca e egal cu suma elementelor din subarborii drept si stang; la fiecare pas returnez suma din subarbori +valoarea nodului curent.



10. Verificati daca un arbore binar e complet  
**Idee** -  parcurgeti arborele pe niveluri. De fiecare data cand scoateti un nod din coada, verificati ca are copii in stanga si in dreapta. Daca un nod are doar un copil, atunci toate nodurile din coada trebuie sa nu mai aiba copii, altfel arborele nu e complet.



**Idee  -** Alta abordare se foloseste de reprezentarea arborelui sub forma de vector. Copilul stang al unui nod cu index i se gaseste pe pozitia 2i+1 si cel drept pe pozitia 2i+2. Contruim astfel vectorul si daca elementele adaugate se gasesc pe pozitii consecutive, arborele e complet.

