Structuri de Date și Algoritmi
Laboratorul 7: Arbori AVL

Mihai Nan

11 aprilie 2023

1. Introducere

Scopul acestui laborator îl reprezintă implementarea unui arbore binar de căutare echilibrat. Obiectivul este acela de a vă familiariza cu o variantă relativ simplă prin care putem echilibra un arbore binar de căutare.

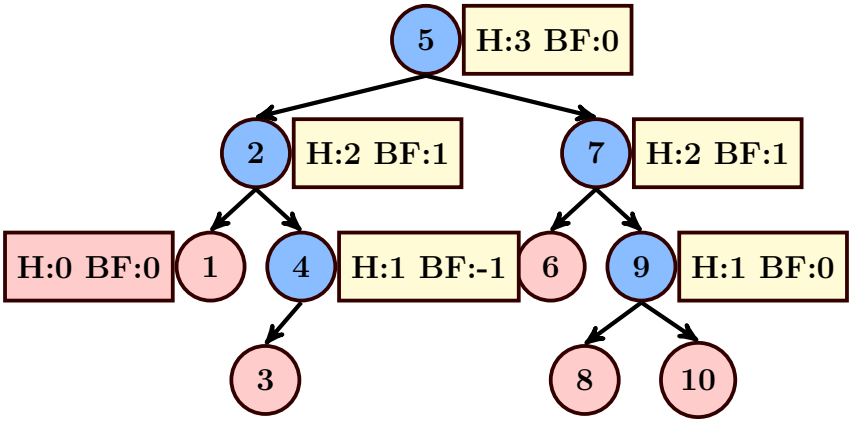
Laboratorul are 3 cerințe care urmăresc:

- implementarea celor 4 tipuri de rotații;
- implementarea operației de inserare a unui nod într-un arbore AVL;
- implementarea unei funcții care șterge un nod dintr-un arbore AVL.

2. Arbori binari de căutare echilibrați (AVL)

Factorul de echilibru al unui nod într-un arbore AVL este diferența dintre înălțimea subarborelui drept și înălțimea subarborelui stâng al acelui nod. În mod formal, dacă nodul este denumit cu N, iar înălțimea subarborelui stâng și înălțimea subarborelui drept sunt denumite cu h_{left} și, respectiv, h_{right}, atunci factorul de echilibru al nodului N este:

$$BalanceFactor(N) = h_{right} - h_{left}$$



Într-un arbore AVL, factorul de echilibru al fiecărui nod trebuie să fie un element al mulțimii $\{-1, 0, 1\}$, adică subarborele stâng și subarborele drept trebuie să fie aproximativ la fel de mari. Dacă factorul de echilibru al unui nod depășește acest interval, atunci arborele nu mai respectă proprietatea AVL și trebuie reechilibrat prin efectuarea unei rotații a nodurilor din arbore.

Reprezentarea structurii

Pentru a reprezenta un arbore AVL, vom folosi următoarea structură:

```
1  typedef int T;
2
3  typedef struct node {
4          T value;
5          int height;
6          struct node *left, *right;
7  } *Tree;
```

Avem nevoie de câmpul `height` în structura nodului într-un arbore AVL pentru a calcula factorul de echilibru al unui nod și pentru a menține proprietatea de echilibru a arborelui. Câmpul `height` stochează înălțimea subarborelui cu rădăcina în nodul respectiv.

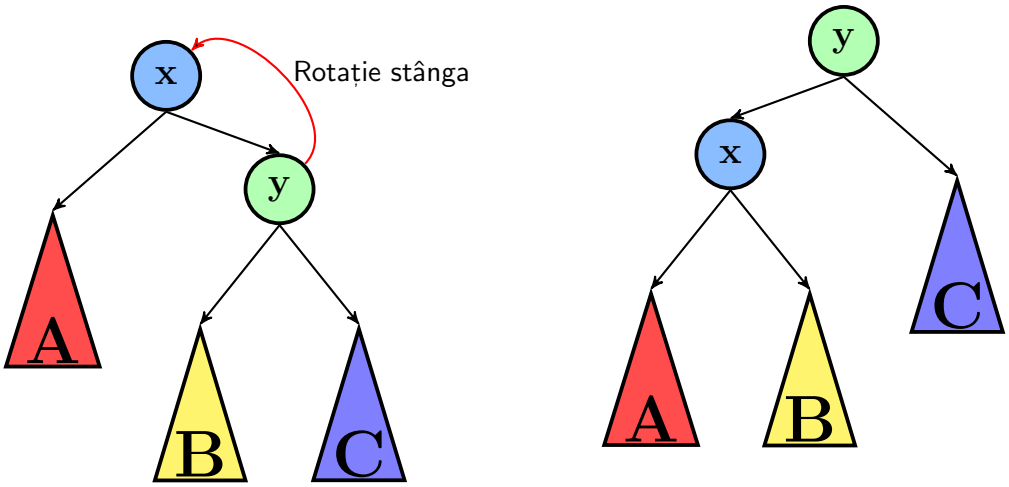
Dacă factorul de echilibru depășește o limită, în cazul arborelui AVL fiind 1, atunci trebuie să realizăm o serie de rotații pentru a readuce arborele la echilibru. Pentru a determina tipul acestor rotații, avem nevoie de înălțimile subarborilor, pe care le obținem din câmpurile `height` ale nodurilor. De asemenea, pentru a menține proprietatea de echilibru în timpul inserării și ștergerii unui nod, trebuie să actualizăm valorile câmpului `height` în mod corespunzător.

Cerința 1 (0p) Analizați antetele funcțiilor definite în fișierul `tree.h` și parcurgeți descrierile lor.

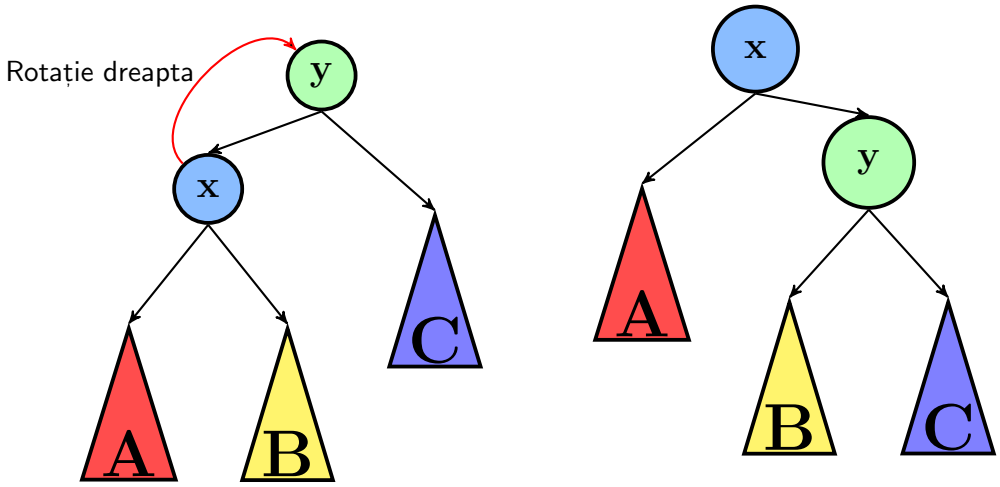
- `createTree`: Primește ca parametru o valoare și returnează un pointer la un nod frunză ce va conține valoarea primită ca argument și câmpul `height` setat corespunzător.
- `max`: Este o funcție care determină valoarea maximă.
- `height`: Este funcția care întoarce înălțimea unui arbore. Vom considera că înălțimea pentru arborele vid (`NULL`) este 0.
- `balancedFactor`: Primește ca parametru un nod și întoarce factorul de echilibru pentru acel nod. Factorul de echilibru o să fie calculat după formula:

```
BF(root) = height(root->right) - height(root->left)
```

- `updateHeight`: Este o funcție care primește ca argument un nod pentru care actualizează valoarea câmpului `height` pe baza valorilor din nodurile copil.
- `leftRotation`: Funcție care primește ca parametru un nod și realizează operația de rotație stânga.



- `rightRotation`: Funcție care primește ca parametru un nod și realizează operația de rotire dreapta.



- `insert`: Primește un arbore și un `elem`. Funcția inserează valoarea în arbore, dacă ea nu există deja, ținând cont de proprietățile unui arbore binar de căutare și apoi verifică dacă arborele este echilibrat. În cazul în care arborele obținut nu este unul care îndeplinește condiția AVL, atunci aplică rotațiile necesare.

- `minimum`: Funcție care determină nodul ce conține elementul minim dintr-un arbore binar de căutare. Funcția va întoarce `NULL` pentru arborele vid.

- `delete`: Funcție care primește ca argumente rădăcina unui arbore și o valoare și șterge nodul care conține respectiva valoare din arbore. Funcția va întoarce rădăcina arborelui rezultat după ștergere. Exact ca în cazul inserarii, trebuie să ne asigurăm că arborele rezultat îndeplinește condiția AVL.

- `destroyTree`: Funcție care dealocă întreaga memorie alocată pentru un arbore binar. Pointerul la rădăcina arborelui primit ca parametru va pointa către `NULL` după ce se va apela funcția

Cerința 2 (2p) Implementați funcțiile de mai sus în urmatoarea ordine: `createTree`, `max`, `height`, `balanceFactor`, `updateHeight`, `minimum` și `destroyTree`.

Cerința 3 (4p) Implementați funcțiile `leftRotation` și `rightRotation` după care implementați funcția de inserare a unui nod într-un arbore AVL. **Arborele rezultat trebuie să îndeplinească proprietatea de arbore AVL!** Analizați cele 4 cazuri prezentate la curs.

Cerința 4 (4p) Implementați funcția `delete` care va șterge un nod dintr-un arbore AVL. **Arborele rezultat trebuie să îndeplinească proprietatea de arbore AVL!** De asemenea, va trebui să analizăm cele 4 cazuri prezentate la curs pentru inserare.