Laborator 7 – Treapuri si trie

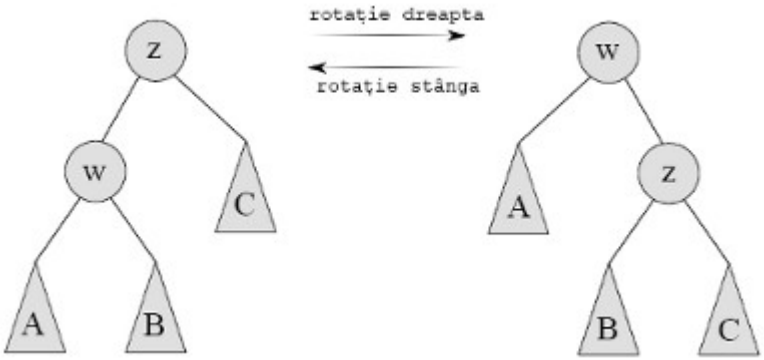
Marti 16-18

I. TREAPURI

Arborii binari de cautare echilibrati presupun, pe langa pasii studiati data trecuta, operatii de balansare pentru a echilibra arborele. Acestea sunt apelate recursiv de fiecare data dupa inserarea unui nod, respectiv stergerea unui nod – se insereaza un nod recursiv, apoi se balanseaza arborele recursiv pana se ajunge inapoi la radacina.

Exista mai multe tipuri de arbori echilibrati, in functie de modul de a face rotatiile . AVL-urile fac rotatii daca inaltimea subarborelui drept difera de inaltimea subarborelui stang cu mai mult de o unitate. In functie de caz, exista 4 rotatii posibile.

O varianta mai simpla, cu doar 2 rotatii, o gasim la Treapuri. Treapurile sunt un tip de arbori care folosesc algoritmi aleatori – fiecarui nod i se atribuie la inserare pe langa cheie inca o prioritate random. Treapul este arbore binar de cautare pentru chei si heap pentru prioritati – fiecare nod trebuie sa aiba prioritatile mai mari decat nodurile copiilor. Se demonstreaza ca in acest fel arborele ramane echilibrat si inaltimea medie a arborelui este O(log n). Daca in urma unei inserari, unul din copii nu respecta regula de la prioritati, se face o rotatie in felul urmator:



Pentru insert – dupa ce s-a adaugat un nod, trebuie balansat arborele recursiv pana la radacina.

Functia de delete este un pic diferita. Se doreste ca nodul de sters sa devina o frunza.

Daca nu are niciun copil (e deja frunza), se sterge pur si simplu nodul.

Daca avem cel putin un copil, se roteste astfel incat cel cu prioritate mai mare sa ia locul nodului pe care dorim sa il stergem, apoi se apeleaza recursiv pentru a sterge nodul (care acum s-a mutat cu un nivel mai jos, in urma rotatiei). Treapul este util pentru query-uri de tip – aflati cea de-a k-a cheie din arbore in ordine crescatoare, in complexitate logaritmica. Pentru asta, trebuie mentinute informatii suplimentare – numarul total de noduri continute de un nod si parintele unui nod.

Dandu-se urmatoarea structura:

```
typedef struct Treap_node {
        int key, priority;
        struct Treap *left ,*right, *parent;
        int nr_nodes; // numarul de noduri din subarborele curent
} Treap_node ;
```

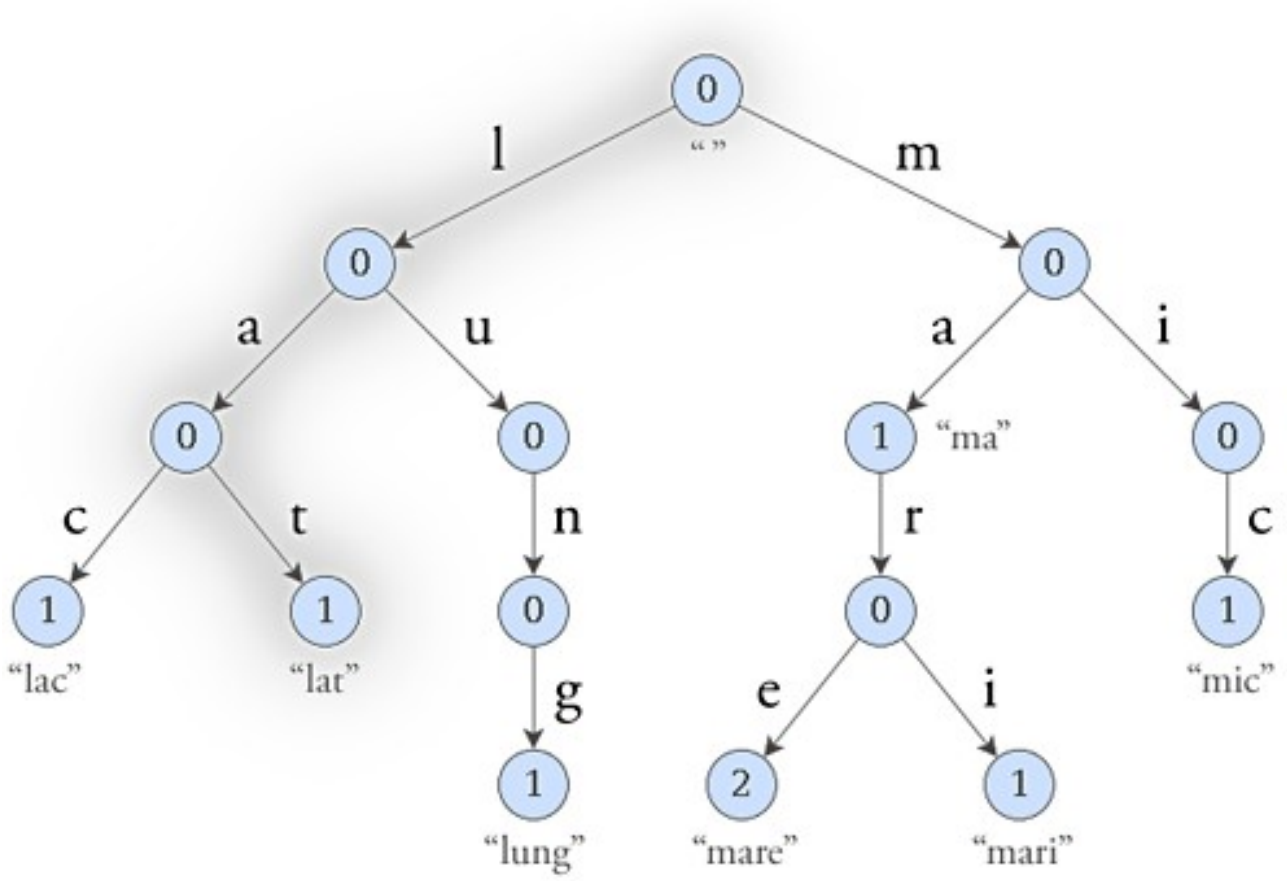
modificati functiile de inserare si stergere de data trecuta astfel incat sa contina operatiile specifice pentru treapuri.

-> **query de tip a k-a valoare in ordine crescatoare**

-> Treap cu chei implicite

insert(D, x, poz);
-> split
-> join

II. Trie
Un trie este o structura de date care ne ajuta pentru situatii in care vrem sa vedem toate cuvintele dintr-un dictionar care incep cu acelasi prefix. Fiecare nod tine 26 de copii, corespunzatori unei litere din alfabet, astfel incat daca se adauga cuvinte care incep cu aceleasi caractere, vor urma aceeasi cale in arbore. Pentru fiecare cuvant, putem retine si numarul de aparitii in arbore.



Dandu-se structura:
typdef struct Trie{
	int count, nrfii;
	struct Trie* child[26];
}
Sa se scrie operatiile de init, insert, delete si free. Campul de count este 0 daca nu se termina un cuvant in nodul respectiv, respectiv 1 sau mai mult daca avem un cuvant sau mai multe cu valoarea respectiva Campul nrfii ne ajuta sa stim cand putem sterge un nod – daca nu mai are copii si count = 0.

III. Din fisierul "lab7.in" se citesc N si K, apoi N cuvinte. Sa se afiseze N linii cu K numere, reprezentand numarul de cuvinte care au un prefix comun de lungime 1,2... K cu cuvantul i (exceptandu-l pe i).
Ex:
Input:
N= 4, K = 3

ana
anais
andrei
alina
Output:
3 2 1 0
3 2 1 0
3 2 0 0
3 0 0 0

Linkuri utile:
https://www.infoarena.ro/problema/trie
https://www.infoarena.ro/treapuri