Tema Nr. 6: Permutarea Josephus

Timp alocat: 4 ore

Implementare

Se cere implementarea **corecta** si **eficienta** a unui algoritm de complexitate *O(nlgn)* pentru determinarea permutarii *Josephus(n,m)*, atunci cand *m* nu e constant (problema 14.2 (b) din carte[1]).

Se cere sa folositi un *arbore binar* de *cautare* **echilibrat**. Fiecare nod din arbore trebuie extins cu un camp *size* (dimensiunea sub-arborelului ce are nodul ca radacina). Primul pas e construirea unui arbore binar de cautare **echilibrat** cu cheile 1,2,...,n (*hint*: *divide et impera*). Apoi, la fiecare pas, trebuie *selectat* si *sters* al k-lea element din arbore.

Pseudo-codul algoritmului:

```
JOSEPHUS(n,m)
    T = BUILD_TREE(n)
    j ←1
    for k ←n downto 1 do
        j ←(( j + m − 2) mod k) + 1
        x ←OS-SELECT(root[T], j )
        print key[x]
        OS-DELETE(T, x)
```

Pseudo-codul pentru OS-SELECT poate fi gasit la Capitolul 14.1 din carte[1]. Pentru OS-DELETE, puteti folosi stergerea dintr-un arbore binar de cautare, fara a creste inaltimea arborelui (De ce nu trebuie sa re-balansati arborele?). Nu uitati sa pastrati campul *size* consistent o data cu stergerile din arbore (exista mai multe abordari prin care puteti modifica campul *size* fara a creste complexitatea algoritmului). Pentru BUILD_TREE, trebuie sa construiti un arbore binar de cautare echilibrat cu cheile 1, 2, …, n. Nu uitati sa initializati campul *size* in BUILD_TREE.

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*

Evaluare

! Inainte de a incepe sa lucrati pe partea de evaluare, asigurati-va ca aveti un **algoritm corect**! Corectitudinea algoritmilor va trebui demonstrata pe date de intrare de dimensiuni mici. Pentru Josephus(7,3) afisati (cu *preety print*) arborele echilibrat dupa fiecare pas din algoritm (dupa BUILD_TREE si dupa fiecare OS_DELETE).

Pentru evaluare, variati *n* de la 100 la 10000 cu un pas de 100; pentru fiecare *n*, alegeti *m=n/2*.

Evaluati complexitatea algoritmului ca si suma atribuirilor si a comparatiilor pentru fiecare valoare a lui *n*. Ati atins O(nlgn)?