

# Proiectarea algoritmilor

## Lucrare de laborator nr. 3

### Căutarea în secvențe sortate și în arbori binari de căutare

## Cuprins

1	Căutarea binară	1
2	Căutare în arbori binari de căutare	2
3	Sarcini de lucru și barem de notare	2

## 1 Căutarea binară

### Problema căutării binare

*Intrare:*  $n, (a_0, \dots, a_{n-1}), z$  numere întregi;  
secvența  $(a_0, \dots, a_{n-1})$  este sortată crescător,  
adică  $a_i \leq a_{i+1}, i \in \{0, \dots, n-2\}$ .

*Ieșire:*  $poz = \begin{cases} k \in \{i \mid a_i = z\} & \text{dacă } \{i \mid a_i = z\} \neq \emptyset, \\ -1 & \text{altfel.} \end{cases}$

Presupunem că secvența  $(a_0, \dots, a_{n-1})$  este memorată în tabloul  $(a[i] \mid 0 \leq i \leq n-1)$ . Esența căutării binare constă în compararea elementului căutat cu elementul din mijlocul zonei de căutare și în cazul în care elementul căutat nu este egal cu acesta, se restrânge căutarea la subzona din stânga sau din dreapta, în funcție de rezultatul comparării. Dacă elementul căutat este mai mic decât cel din mijlocul zonei de căutare, se alege subzona din stânga, altfel subzona din dreapta. Inițial, zona de căutare este tabloul  $a$ . Convenim să notăm cu  $i_{stg}$  indicele elementului din stânga zonei de căutare în tablou,  $i_{dr}$  indicele elementului din dreapta zonei de căutare în tablou.

```
function cautareBinara(a, n, z)
    istg ← 0
    idr ← n-1
    while (istg ≤ idr) do
        imed ← ⌊(istg+idr)/2⌋
        if (a[imed]=z)
            then return imed
        else if (a[imed]>z)
            then idr ← imed-1 /* se cauta in stanga */
            else istg ← imed+1 /* se cauta in dreapta */
    return -1
end
```

Dimensiunea problemei căutării binare este dată de dimensiunea  $n$  a secvenței în care se face căutarea. Presupunem că toate operațiile necesită o unitate de timp. Calculul timpului de execuție al algoritmului constă în determinarea numărului de execuții ale blocului de instrucțiuni asociat cu instrucțiunea `while`. Se observă că, după fiecare iterație a buclei `while`, dimensiunea zonei de căutare se înjumătățește.

Cazul cel mai favorabil este obținut când  $a \lfloor \frac{n-1}{2} \rfloor = z$  și se efectuează două comparații și trei atribuiri. Rezultă  $T_{A_4}^{fav}(n) = 2 + 3 = 5$ .

Cazul cel mai nefavorabil este în situația în care tabloul  $a$  nu conține valoarea căutată. Pentru simplitate, se consideră  $n = 2^k$ , unde  $k$  este numărul de înjumătățiri. Rezultă  $k = \log_2 n$  și printr-o majorare,  $T_{A_4}(n) \leq c \log_2 n + 1$ , unde  $c$  este o constantă,  $c \geq 1$ .

Spațiul necesar execuției algoritmului  $A_4$  este  $n + 7$  (tabloul  $a$ , constantele 0 și  $-1$ , variabilele  $i_{stg}$ ,  $i_{dr}$ ,  $i_{med}$ ,  $n$  și  $z$ ).

## 2 Căutare în arbori binari de căutare

Un *arbore binar de căutare* este un arbore binar cu proprietățile:

- informațiile din noduri sunt elemente dintr-o mulțime total ordonată;
- pentru fiecare nod  $v$ , valorile memorate în subarboarele stâng sunt mai mici decât valoarea memorată în  $v$ , iar valorile memorate în subarboarele drept sunt mai mari decât valoarea memorată în  $v$ .

```
function cautArboreBinar(t, a)
    p ← t
    while ((p ≠ NULL) and (a ≠ p->elt)) do
        if (a < p->elt)
            then p ← p->stg
            else p ← p->drp
    return p
end
```

Funcția poz ia valoarea *NULL*, dacă  $a \notin S$  și adresa nodului care conține pe  $a$  în caz contrar.

Operațiile de inserare și de ștergere trebuie să păstreze invariantă următoarea proprietate: *valorile din lista inordine a nodurilor arborelui trebuie să fie în ordine crescătoare.*

## 3 Sarcini de lucru și barem de notare

### Sarcini de lucru:

1. Scrieți o funcție C/C++ care implementează algoritmul de căutare binară.
2. Scrieți o funcție C/C++ care implementează algoritmul de căutare într-un arbore binar de căutare.
3. Contorizați numărul de comparații. Comentați rezultatele obținute.

### Barem de notare:

1. Funcția C/C++ care implementează algoritmul de căutare binară: 2p
2. Funcția C/C++ care implementează algoritmul de căutare într-un arbore binar de căutare: 3p

3. Contorizarea numărului de comparații: 4p

4. Baza: 1p

## **Bibliografie**

[1] Lucanu, D. și Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.