Proiectarea algoritmilor

Căutarea în secvențe sortate și în arbori binari de căutare

Lucrare de laborator nr. 3

Cuprins

Căutarea binară

Problema căutării binare

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

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

- 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 $\texttt{i}_{stg}$ indicele elementului din stânga zonei de căutare în tablou, $\texttt{i}_{dr}$ indicele elementului din dreapta zonei de căutare în tablou.

Căutarea binară - pseudocod

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

Analiza algoritmului de căutare binară

- Dimensiunea problemei căutării binare este dată de dimensiunea n a secvenței în care se face căutarea. Și de această dată 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 vectorul 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 $\texttt{i}_{stg}$, $\texttt{i}_{dr}$, $\texttt{i}_{med}$, `n` și `z`).

Căutare în arbori binari de cătare oarecare

- 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 subarborele stâng sunt mai mici decât valoarea memorată în v, iar valorile memorate în subarborele 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.

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 un arbore binar de cătare.
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 cătare: 3p
3. Contorizarea numărului de comparații: 4p
4. Baza: 1p

Bibliografie

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