Arbori binari - a.b. stricti si a.b. echilibrati AVL
Aplicatii la analiza performantei cautarii binare

Reamintim urm. proprietati pt. un **arbore binar strict** T:
$N_E(T)$ (sau N_E) = numarul de frunze ale lui T;
$N_I(T)$ (sau N_I) = numarul de noduri interne ale lui T;
$L_E(T)$ (sau L_E) = lungimea externa a lui T;
$L_I(T)$ (sau L_I) = lungimea interna a lui T;

Proposition 0.1 *In oricare a.b.s. T avem relatia:*

$$N_E = N_I + 1.$$

Proposition 0.2 *In oricare a.b.s. T avem relatia:*

$$L_E = L_I + 2N_I.$$

Proposition 0.3 *In oricare a.b.s. T de inaltime $h(T) = d$ avem relatia:*

$$N_E \leq 2^d.$$

Corrollary 0.1 *In oricare a.b.s. T de inaltime $h(T) = d$ avem relatia:*

$$d \geq \lceil log_2 N_E \rceil.$$

Proposition 0.4 *In familia a.b. stricti cu numar de frunze fixat, N_E, lungimea externa minima se atinge pentru aceia care au frunzele repartizate pe cel mult doua niveluri adiacente.*

Proposition 0.5 *Lungimea externa minima a unui a.b.s. cu N_E frunze este:*

$$L_E^{min} = N_E \lfloor log_2 N_E \rfloor + 2(N_E - 2^{\lfloor log_2 N_E \rfloor}).$$

Din *Demonstratie:* daca $d = h(T)$= inaltimea unui a.b.s. T pe care se atinge lungimea externa minima, avem 2 cazuri (cf. Prop. 0.4):

cazul (a): Toate frunzele sunt la acelasi nivel, $d = h(T)$, daca $N_E = 2^d$.

cazul (b): Frunzele nu sunt toate la acelasi nivel. Dar atunci ele sunt repartizate doar pe nivelurile $d-1$ (fie y nr. de frunze de la acest nivel), si d (fie $2x$ nr. de frunze de la acest nivel, x= nr. de noduri interne de la nivelul $d-1$). Se rezolva sistemul (1) $x + y = 2^{d-1}$, (2) $2x + y = N_E$. Avem:

nr. de frunze la nivelul $d - 1 = y = 2^d - N_E$.

nr. de frunze la nivelul $d = 2x = 2N_E - 2^d$.

Proposition 0.6 *Intr-un a.b.s.* lungimea externa medie $L_E^{medie} = L_E/N_E$ *are marginea inferioara*

$$L_E^{medie} \geq \lfloor log_2 N_E \rfloor.$$

(Alte) Aplicatii ale a.b.s.
Analiza performantei cautarii binare

Problema: Cautam o cheie x intr-o multime total ordonata $X = \{x_1 < x_2 < \ldots < x_n\}$.

Algoritm de cautare:

(1) Se compara x cu o cheie oarecare x_k din X. Daca $x = x_k$ am terminat (cautare cu succes). Daca nu,

(2) daca $x < x_k$ se cauta x in multimea ordonata $X_k^s = \{x_1 < x_2 < \ldots < x_{k-1}\}$ (se reia de la (1) cu $X = X_k^s$).

(3) daca $x > x_k$ se cauta x in multimea ordonata $X_k^d = \{x_{k+1} < \ldots < x_n\}$ (se reia de la (1) cu $X = X_k^d$).

Arbore binar de cautare $T(X)$ asociat alg. de m. sus (si multimii X):

(1) $info(root(T(X))) = x_k$

(2) $left(T(X)) = T(X_k^s)$ este a.b.c. asociat m. X_k^s

(3) $right(T(X)) = T(X_k^d)$ este a.b.c. asociat m. X_k^d

Algoritmul de cautare binara: la fiecare pas, k se alege in raport cu $n = |X|$, ca si $k = \lfloor n/2 \rfloor$. (i.e. $k = p$ pentru $n = 2p$ sau $n = 2p + 1$.)

Arbore binar de decizie asociat unei cautari a lui x in X= a.b.c. $T(X)$, pe care il extindem canonic la un a.b.s. (punem frunze in locul subarborilor vizi).

Cautarea cu succes: se termina intr-unul din nodurile interioare, cel ce contine prima cheie $x_k = x$.

Cautarea fara succes: se termina intr-o frunza (fost subarbore vid, in care s-ar fi inserat cheia x daca faceam cautare-cu-inserare).

Performanta cautarii cu succes= L_I a a.b. de decizie asociat cautarii respective.

Performanta cautarii fara succes= L_E a a.b. de decizie asociat cautarii respective.

Cardinalul multimii X in care se cauta este $n = N_I$ al a.b. de decizie asociat cautarii respective.

Din Propozitia 0.2 $L_E = L_I + 2\ N_I$. (deci performatele cautarii cu si fara succes sunt legate...)

Theorem 0.1 *Intre arborii de decizie asociati unui algoritm de cautare intr-o multime cu n chei, sunt* optimi *(i.e. au L_E minima) cei asociati cautarii binare.*

Dem.: Stim din Prop. 0.4 ca au L_E minima acei a.b.s. care au frunzele repartizate pe cel mult doua niveluri adiacente.

Theorem 0.2 *Urm. afirmatii sunt adevarate:*

1. *Daca* $2^{k-1} \leq n < 2^k$ *o* **cautare cu succes** *folosind alg. cautare binara necesita cel mult* k *comparatii.*

2. *Daca* $n = 2^k - 1$ *o* **cautare fara succes** *folosind alg. cautare binara necesita exact* k *comparatii.*

3. *Daca* $2^{k-1} \leq n < 2^k - 1$ *o* **cautare fara succes** *folosind alg. cautare binara necesita fie exact* $k-1$*, fie exact* k *comparatii.*

Dem.: Din structura a.b. de decizie asociat algoritmului si proprietatile sale.
Relatia 1.: margine sup. pt. inaltime.
Pt. relatiile 2. si 3., se fol. Prop. 0.4, ..., frunzele sunt repartizate
- fie pe un singur nivel (cazul 2) (Daca $n = 2^k - 1$, acest nivel e chiar k)
- fie pe c. mult 2 niveluri adiacente (cazul 3), acestea vor fi $k-1$ si k.
q.e.d.

Theorem 0.3 *Algoritmul de cautare binara* **minimizeaza nr. mediu de comparatii***, atat in cazul cautarii cu succes, cat si in cazul cautarii fara succes.*

C_n = numarul mediu de comparatii la cautarea cu succes intr-o multime cu n chei, fiecare din cele n chei este egal probabila.

C'_n = numarul mediu de comparatii la cautarea fara succes intr-o multime cu n chei; fiecare din cele $n+1$ intervale dintre chei este egal probabil.

Lemma 0.1

$$C_n = (1 + \frac{1}{n})C'_n - 1.$$

Dem: Fie T a.b.s. de decizie asociat cautarii binare intr-o multime cu n chei. Are $N_I(T) = n$. Avem:

$$C_n = \frac{L_I(T)}{n} + 1.$$

$$C'_n = \frac{L_E(T)}{n+1}.$$

Manipulari algebrice si folosim Propozitia 0.2 ca sa ajungem la formula din enuntul lemei.
Din formula: C_n si C'_n se minimizeaza simultan.
q.e.d.