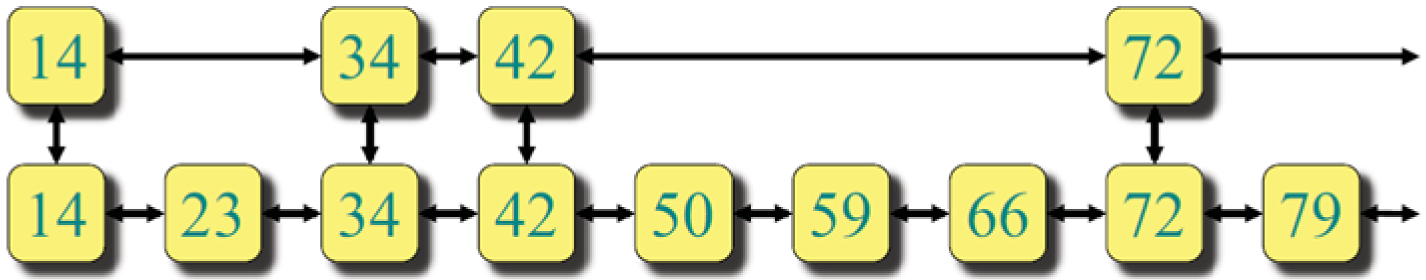
Cursul 7

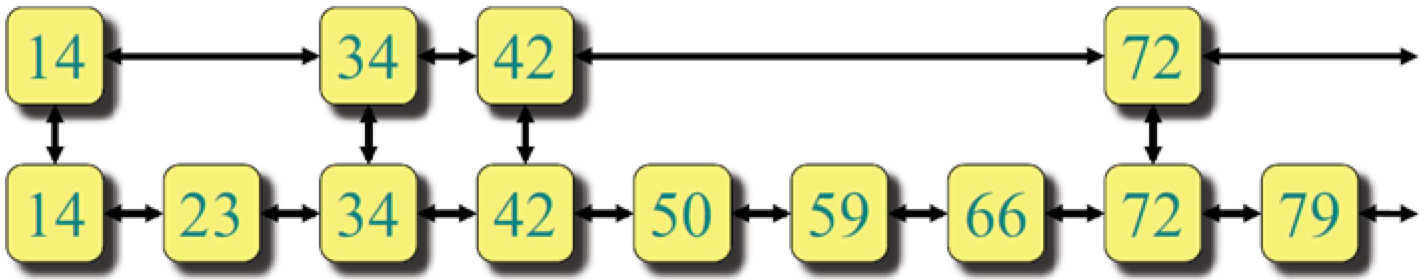
Randomized Data Structures

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Skip lists

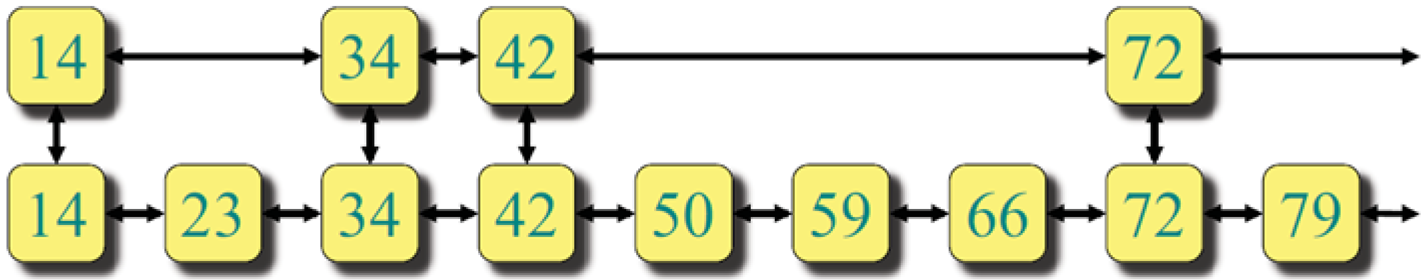


Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

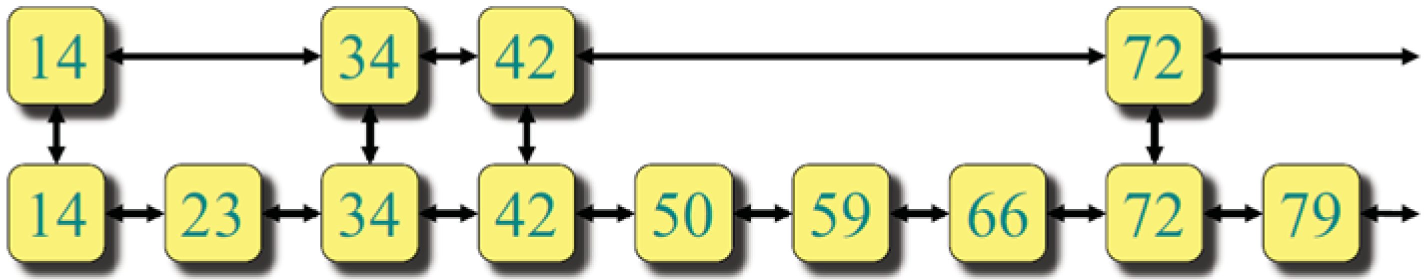
Costul pentru 2 nivele : $2\sqrt{n}$

Costul pentru 3 nivele: $3\sqrt[3]{n}$

…

Costul pentru k nivele:

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

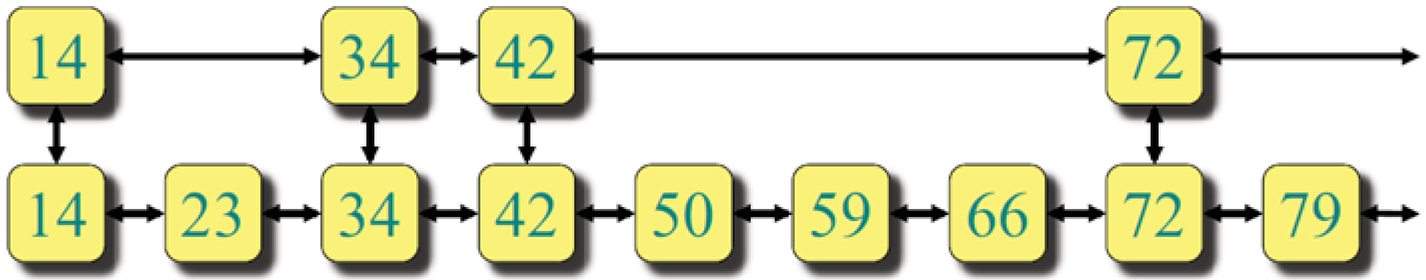
Costul pentru 2 nivele : $2\sqrt{n}$

Costul pentru 3 nivele: $3\sqrt[3]{n}$

…

Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

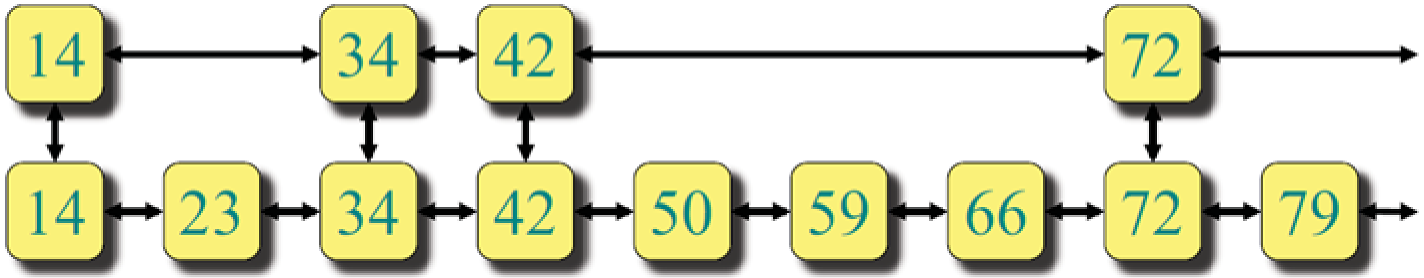
Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Costul pentru $c * \lg(n)$ nivele ar fi

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

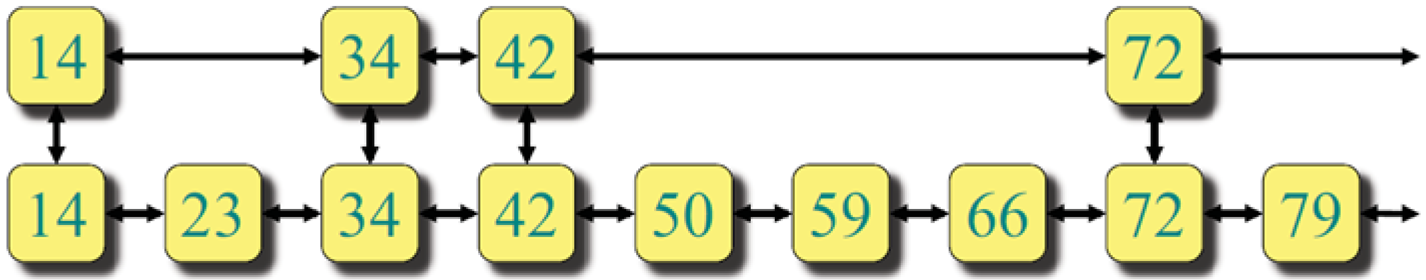
Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Costul pentru $c * \lg(n)$ nivele ar fi: $c * \lg(n) * \sqrt[\lg(n)]{n} =$

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

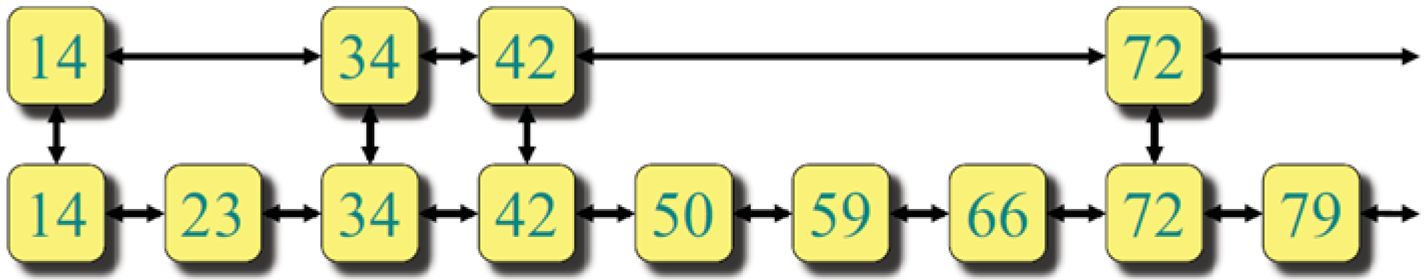
Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Costul pentru $c * \lg(n)$ nivele ar fi: $c * \lg(n) * \sqrt[\lg(n)]{n} = \boldsymbol{c * e * \lg(n)}$

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

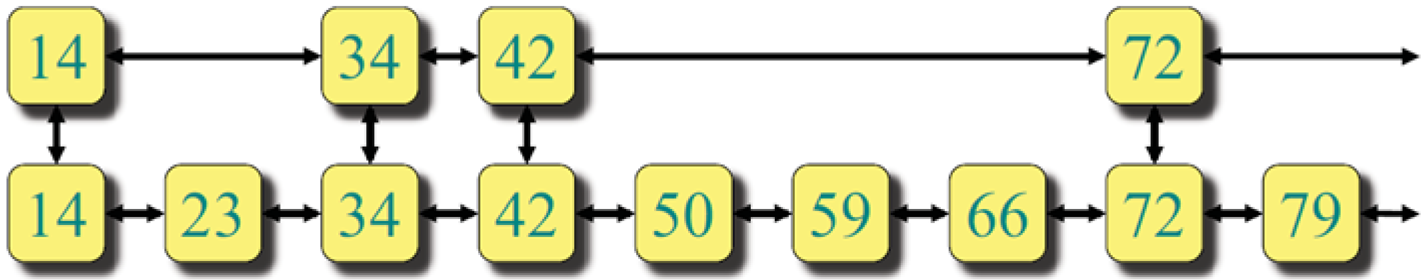
Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Costul pentru $c * \lg(n)$ nivele ar fi: $c * \lg(n) * \sqrt[\lg(n)]{n} = \boldsymbol{c * e * \lg(n)}$

$$\lg\left(\sqrt[\lg(n)]{n}\right) = \lg\left(n^{\frac{1}{\lg(n)}}\right) = \frac{1}{\lg(n)} * \lg(n) = 1;$$

$$deci \sqrt[\lg(n)]{n} = e$$

Skip lists



Fie L_2 (nivelul inferior) si L_1 (nivelul superior) a unei structure de forma celei din slide-ul 17 din curs.

Căutarea în această structură este $|L_1| + |L_2|/|L_1|$

Unde: $|L_2| = n; |L_1| = \sqrt{n}$

Costul pentru k nivele: $\mathrm{k}\sqrt[k]{n}$

Costul pentru $c * \lg(n)$ nivele ar fi: $c * \lg(n) * \sqrt[\lg(n)]{n} = \boldsymbol{c * e * \lg(n)}$

Concluzie: Pentru un numar de nivele O(lg(n)), cautarea are complexitate logaritmica ~ O(lg(n)).

Skip lists: Operații

- Select (X):

Plec de la elementul de pe nivelul cel mai de sus, iterez până găsesc cel mai mare element $e \leq X$, cobor un nivel și continui iterațiile (către dreapta, respectiv în jos) până când ajung la nivelul cel mai de jos unde îl găsesc (sau nu) pe X.

Skip lists: Operații

- Select (X): ~ O(?)

Plec de la elementul de pe nivelul cel mai de sus, iterez până găsesc cel mai mare element $e \leq X$, cobor un nivel și continui iterațiile (către dreapta, respectiv în jos) până când ajung la nivelul cel mai de jos unde îl găsesc (sau nu) pe X.

Skip lists: Operații

- Select (X): ~ O(?)

Plec de la elementul de pe nivelul cel mai de sus, iterez până găsesc cel mai mare element $e \leq X$, cobor un nivel și continui iterațiile (către dreapta, respectiv în jos) până când ajung la nivelul cel mai de jos unde îl găsesc (sau nu) pe X.

Skip lists: Operații

- Inserare (X):

Caut poziția lui X. Dau cu banul pentru a îl promova pe un nivel superior.

Skip lists: Operații

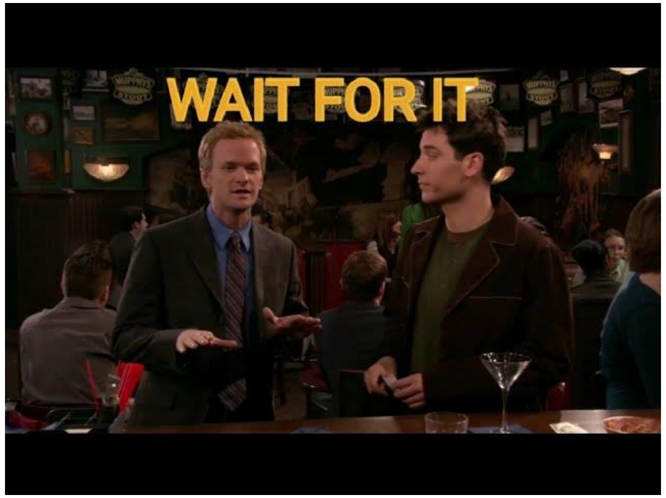
- Inserare (X): ~ O(?)

Caut poziția lui X. Dau cu banul pentru a îl promova pe un nivel superior.

Skip lists: Operații

- Inserare (X): ~ O(?)

Caut poziția lui X. Dau cu banul pentru a îl promova pe un nivel superior.



Skip lists: Operații

- Ștergere (X):

Caut pe X.

Șterg pe X de pe toate nivelele -> O(”numărul de nivele”)

Skip lists: Operații

- Ștergere (X): ~ O(?)

Caut pe X.

Șterg pe X de pe toate nivelele -> O(”numărul de nivele”)

Skip lists: Operații

- Ștergere (X): ~ O(?)

Caut pe X.

Șterg pe X de pe toate nivelele -> O(”numărul de nivele”)

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Definim conceptul de “**probabil**”:

Un eveniment E se numește “probabil” dacă există c>1 astfel încât $\Pr[E] \geq 1 - O(1/n^c)$

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Definim conceptul de "**probabil**":

Un eveniment E se numește "probabil" dacă există c>1 astfel încât $\Pr[E] \geq 1 - O(1/n^c)$

Union-Binding:

Avem n evenimente: $E_1, E_2, \dots, E_n$,
atunci: $\Pr(E_1 \cup E_2 \cup \cdots \cup E_n) \leq \Pr(E_1) + \Pr(E_2) + \cdots + \Pr(E_n)$.

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații = **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații = **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

Skip lists: Operații

Lema A:

Un Skip List cu *n* noduri "probabil" are O(lg(n)) nivele.

Demonstrație:

$nr\ de\ nivele \leq c * \lg(n)$
$\Pr(nr\ de\ nivele \leq c * \lg(n)) = 1 - \Pr(nr\ de\ nivele\ > c * \lg(n))$
$\Pr(nr\ de\ nivele\ >\ c * \lg(n)) < (½)^{c*\lg(n)} =\ 1/(n^c)$
$deci$
$\Pr(\boldsymbol{nr\ de\ nivele \leq c} * \lg(\boldsymbol{n})) = 1 - \Pr(nr\ de\ nivele\ > c * \lg(n)) > \mathbf{1 - 1}/(\boldsymbol{n^c})$
$Evenimentul\ \text{“}nr\ de\ nivele \leq c * \lg(n)\text{”}\ este\ \underline{probabil}$

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

Skip lists: Operații

Teorema lui Chernoff:

Fie Y o variabila aleatoare care numără cate aruncări ce au rezultat "pajură" (sau "stema") din m aruncari, iar probabilitatea de a pica pajura este p. Pentru orice r>0 avem

$$\mathbf{Pr}(\boldsymbol{Y} >= \boldsymbol{Expected}[\boldsymbol{Y}] + \boldsymbol{r}) \leq \ \boldsymbol{e}^{-(\mathbf{2}\boldsymbol{r}\mathbf{\hat{}}\mathbf{2})/\boldsymbol{m}}$$

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(nr\ de\ aruncari\ ce\ rezulta\ in\ pajura\ >\ c*\lg(n))$$
$$= 1 - \Pr(nr\ aruncari\ pajura \leq c*\lg(n)) =$$
$$= 1 - \Pr(nr\ de\ aruncari\ stema \geq (d-c)\lg(n))$$

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(nr\ de\ aruncari\ ce\ rezulta\ in\ pajura\ >\ c*\lg(n))$$
$$= 1 - \Pr(nr\ aruncari\ pajura \le c*\lg(n)) =$$
$$= 1 - \Pr(nr\ de\ aruncari\ stema \ge (d-c)\lg(n))$$

Fie Y variabila aleatoare ce numără de cate ori a picat stema. m=d*log(n). p=½

$$\Pr(nr\ de\ aruncări\ stema \ge (d-c)\lg(n)) = \Pr(Y \ge (d-c)\lg(n)) = \cdots$$

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(nr\ de\ aruncari\ ce\ rezulta\ in\ pajura\ >\ c * \lg(n))$$
$$= 1 - \Pr(nr\ aruncari\ pajura \le c * \lg(n)) =$$
$$= 1 - \Pr(nr\ de\ aruncari\ stema \ge (d-c)\lg(n))$$

Fie Y variabila aleatoare ce numără de cate ori a picat stema. m=d*log(n). p=½

$$\Pr(nr\ de\ aruncări\ stema \ge (d-c)\lg(n)) = \Pr(Y \ge (d-c)\lg(n)) = \cdots$$
$$\ldots = \Pr(Y >= Expected[Y] + (d/2 - c)\lg(n))$$

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(nr\ de\ aruncări\ stema \geq\ (d-c)\lg(n)) = \Pr(Y \geq\ (d-c)\lg(n)) = \Pr(Y >= Expected[Y] + (d/2-c)\lg(n))$$

știu că Expected[Y]=(d/2)lg(n)

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(nr\ de\ aruncări\ stema \geq\ (d-c)\lg(n)) = \Pr(Y \geq\ (d-c)\lg(n)) = \Pr(Y >= Expected[Y] + (d/2 - c)\lg(n))$$

știu că Expected[Y]=(d/2)lg(n)

aleg d=8c

Skip lists: Operații

Lema B

pentru orice c exista o constanta d astfel încât din d*lg(n) aruncări probabil se obțin >c*lg(n) pajuri.

$$\Pr(Y \geq Expected[Y] + (d/2 - c)\lg(n)) = \Pr(Y \geq Expected[Y] + 3c * \lg(n))$$

$$\Pr(Y \geq Expected[Y] + 3c * \lg(n)) \leq e^{-\frac{2(3c*\lg(n))^2}{8c*\lg(n)}}$$

$$\Pr(Y >= Expected[Y] + 3c * \lg(n)) \leq e^{-\frac{(3c*\lg(n))^2}{4c*\lg(n)}}$$

$$\Pr(Y >= Expected[Y] + 3c * \lg(n)) \leq e^{-\frac{9(c*\lg(n))}{4}} = e^{c*\lg(n)*\left(-\frac{9}{4}\right)} < e^{-c*\lg(n)} = \cdots$$

$$\ldots = n^{-c}\{-c\} = 1/(n^c).$$

$$deci\ \boldsymbol{Pr(nr\ de\ aruncari\ ce\ rezulta\ in\ pajura\ >\ c * lg(n)) >= 1 - 1/(n^c).}$$

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

A) Dar “nr de mișcări pe verticala” este cel mult O(lg(n)) deoarece probabil am atâtea nivele.

B) numărul de mișcări pe orizontala, pana când probabil ajung pe nivelul superior este probabil de cel mult d*lg(n) adică este inclus in O(lg(n))

Skip lists: Operații

Demonstrație

A) Dar “nr de mișcări pe verticala” este cel mult O(lg(n)) deoarece probabil am atâtea nivele.

B) numărul de mișcări pe orizontala, pana când probabil ajung pe nivelul superior este probabil de cel mult d*lg(n) adică este inclus in O(lg(n))

$$\Pr(A\&B) = 1 - \Pr(!\,(A\&B))$$
$$\Pr\big(!\,(A\&B)\big) = \Pr(!\,AU!\,B) \leq \Pr(!\,A) + \Pr(!\,B)$$
$$\leq 1/(n^c) + 1/(n^c)\ \mathit{este\ de\ ordinul}\ O(1/(n\hat{}c))$$
$$\boldsymbol{Pr(A\&B) = 1 - O(1/(n^c))}$$

Skip lists: Operații

Teoremă:

Cătarea se face ***probabil*** în timp **O(lg(n))**

Demonstrație

Numărul de operații **nr de mișcări pe verticală** + **nr de mișcări pe orizontală**

A) Dar “nr de mișcări pe verticala” este cel mult O(lg(n)) deoarece probabil am atâtea nivele.

B) numărul de mișcări pe orizontala, pana când probabil ajung pe nivelul superior este probabil de cel mult d*lg(n) adică este inclus in O(lg(n))

$$\boldsymbol{Pr(A\&B) = 1 - O(1/(n^c))}$$

Inseamna ca toata cautarea se face in timp logaritmic.

De aici reiese timpul logaritmic si pentru celelalte operatii.


Legendary

Bloom Filters

Bloom filters:

- m – array size
- k – numărul de funcții de hashing $(h_1, h_2, \dots h_k\ ), h_i: D \rightarrow \{1,2, \dots m\}$, uniform distribuit atât pe codomeniu cât și în raport cu celelalte hash-uri
- n - numărul de elemente de inserat

Bloom filters:

- m – array size
- k – numărul de funcții de hashing $(h_1, h_2, \dots h_k)$, $h_i : D \rightarrow \{1,2, \dots m\}$, uniform distribuit atât pe codomeniu cât și în raport cu celelalte hash-uri
- n - numărul de elemente de inserat

Funcții:

- Insert (element) – inserează un element
- Check (element) – verifică dacă un element este inserat. Răspunsuri posibile: $\begin{cases} NU \mid 100\%\ corect \\ DA \mid probabil\ corect \end{cases}$

Bloom filters:

Exemplu:

- m – 10
- k – 3: h_1, h_2, h_3
- n – 3: “pug”, “cat”, “bibilică”

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0

Insert(“pug”):

$$h_1(\text{"pug"})=8;\ \mathrm{h}_2(\text{"pug"}) = 2;\ h_3(\text{"pug"}) = 5$$

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	0

Insert("pug"):

$$h_1(\text{"pug"})=8;\ \mathrm{h}_2(\text{"pug"}) = 2; h_3(\text{"pug"}) = 5$$

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
0	1	0	0	1	0	0	1	0	0

Insert(“cat”):

$$h_1(\text{"cat"})=1;\, \mathrm{h}_2(\text{"cat"}) = 5;\, h_3(\text{"cat"}) = 3$$

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
1	1	1	0	1	0	0	1	0	0

Insert(“cat”):

$$h_1(\text{"cat"})=1;\, \mathrm{h}_2(\text{"cat"}) = 5;\, h_3(\text{"cat"}) = 3$$

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
1	1	1	0	1	0	1	1	0	0

Insert(“bibilica”):

$$h_1(\text{"bibilica"})=5;\ \mathrm{h}_2(\text{"bibilica"}) = 1; h_3(\text{"bibilica"}) = 7$$

Bloom filters:

Exemplu:

1	2	3	4	5	6	7	8	9	10
1	1	1	0	1	0	1	1	0	0

check(“bibilica”): - YES

$$h_1(\text{"bibilica"})=5; \mathrm{h}_2(\text{"bibilica"}) = 1; h_3(\text{"bibilica"}) = 7$$

check(“car”): - NO

$$h_1(\text{"car"})=3; \mathrm{h}_2(\text{"car"}) = 8; h_3(\text{"car"}) = 9$$

check(“mouse”): - YES – false positive

$$h_1(\text{"mouse"})=7; \mathrm{h}_2(\text{"mouse"}) = 1; h_3(\text{"mouse"}) = 3$$

Bloom filters

- m – array size
- k – numărul de funcții de hashing
- n - numărul de elemente de inserat

Probabilitatea de false positive:

$$P = (1 - \left(1 - \frac{1}{m}\right)^{kn})^k$$

Pentru o probabilitate de false pozitive *p* și un număr de *n* elemente inserate, atunci lungimea șirului trebuie să fie

$$m = - \frac{n \, lnP}{(\ln 2)^2}$$

Dat fiind m și n, numărul optim de funcții de hashing este

$$k = \frac{m}{n} ln2$$