# Skip Lists

Conrado Martínez
U. Politècnica de Catalunya

RA-MIRI 2023–2024

# Part

1 Skip lists

# Skip lists

W. Pugh

- Skip lists were invented by William Pugh (C. ACM, 1990) as a simple alternative to balanced trees
- The algorithms to search, insert, delete, etc. are very simple to understand and to implement, and they have very good expected performance—independent of any assumption on the input

# Skip lists

W. Pugh

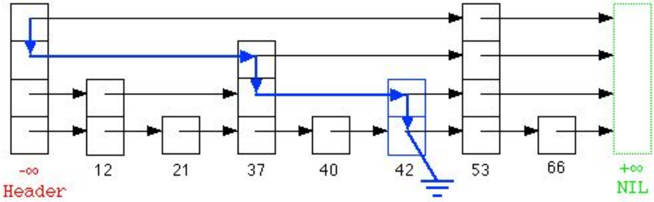
- Skip lists were invented by William Pugh (C. ACM, 1990) as a simple alternative to balanced trees
- The algorithms to search, insert, delete, etc. are very simple to understand and to implement, and they have very good expected performance—independent of any assumption on the input

# Skip lists

A skip list $S$ for a set $X$ consists of:

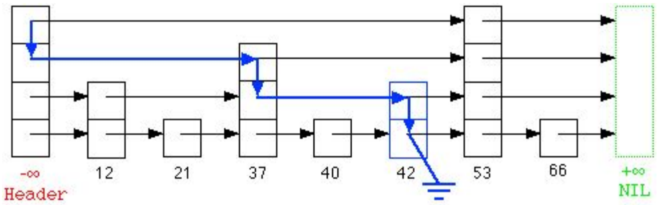
1. A sorted linked list $L_1$, called level 1, contains all elements of $X$
2. A collection of non-empty sorted lists $L_2, L_3, \ldots$, called level 2, level 3, $\ldots$ such that for all $i \geqslant 1$, if an element $x$ belongs to $L_i$ then $x$ belongs to $L_{i+1}$ with probability $p$, for some $0 < p < 1$

# Skip lists




To implement this, we store the items of $X$ in a collection of nodes each holding an item and a variable-size array of pointers to the item's successor at each level; an additional dummy node gives access to the first item of each level

# Skip lists




To implement this, we store the items of $X$ in a collection of nodes each holding an item and a variable-size array of pointers to the item's successor at each level; an additional dummy node gives access to the first item of each level

# Skip lists

- The level or height of a node $x$, height$(x)$, is the number of lists it belongs to.
- It is given by a geometric r.v. of parameter $p$:

$$\mathbb{P}[\text{height}(x) = k] = pq^{k-1}, \qquad q = 1 - p$$

# Skip lists

- The level or height of a node $x$, height$(x)$, is the number of lists it belongs to.
- It is given by a geometric r.v. of parameter $p$:

$$\mathbb{P}[\text{height}(x) = k] = pq^{k-1}, \qquad q = 1 - p$$

# Skip lists

- The height of the skip list $S$ is the number of non-empty lists,

$$\text{height}(S) = \max_{x \in S}\{\text{height}(x)\}$$

- The random variable $H_n$ giving the height of a random skip list of $n$ is the maximum of $n$ i.i.d. $\text{Geom}(p)$
- Several performance measures of skip lists are expressed in terms of the probabilistic behavior of a sequence of $n$ i.i.d. geometric r.v. of parameter $p$

# Skip lists

- The height of the skip list $S$ is the number of non-empty lists,

$$\text{height}(S) = \max_{x \in S}\{\text{height}(x)\}$$

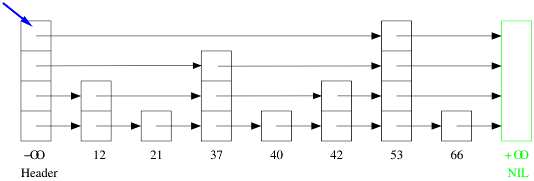
- The random variable $H_n$ giving the height of a random skip list of $n$ is the maximum of $n$ i.i.d. Geom$(p)$
- Several performance measures of skip lists are expressed in terms of the probabilistic behavior of a sequence of $n$ i.i.d. geometric r.v. of parameter $p$

# Skip lists

- The height of the skip list $S$ is the number of non-empty lists,
$$\text{height}(S) = \max_{x \in S}\{\text{height}(x)\}$$
- The random variable $H_n$ giving the height of a random skip list of $n$ is the maximum of $n$ i.i.d. Geom$(p)$
- Several performance measures of skip lists are expressed in terms of the probabilistic behavior of a sequence of $n$ i.i.d. geometric r.v. of parameter $p$

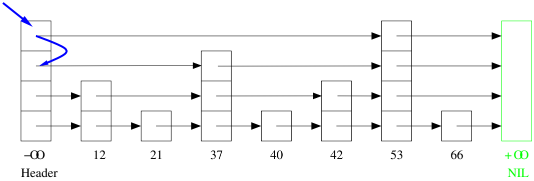
# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$



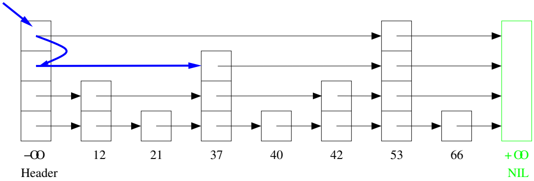
# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$



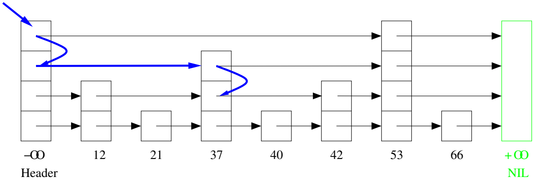
# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$



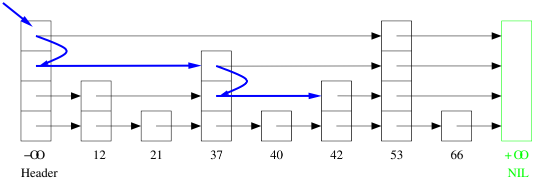
# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$



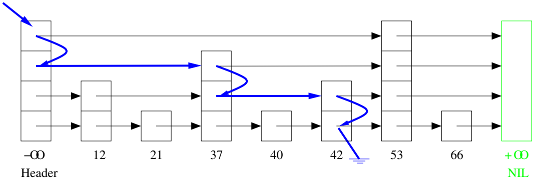
# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$



# Searching in a skip list

Searching for an item $x$, $42 < x \leqslant 53$

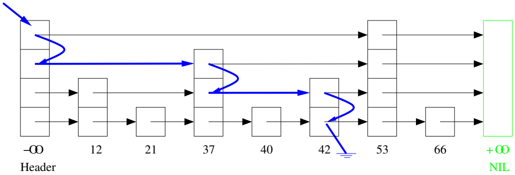


# Searching in a skip list

```
procedure SEARCH(S, x)
    p ← S.header
    ℓ ← S.height
    while ℓ ≠ 0 do
        if p.item < x then
            p ← p.next[ℓ]
        else
            ℓ ← ℓ − 1
        end if
    end while
end procedure
```

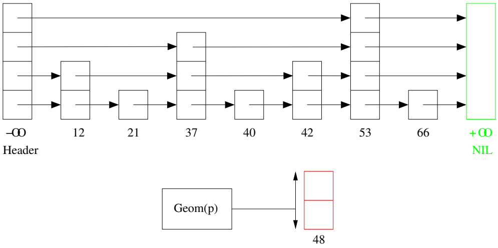
# Insertion in a skip list

Inserting an item $x = 48$



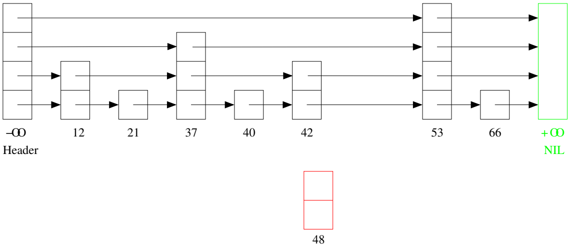
# Insertion in a skip list

Inserting an item $x = 48$



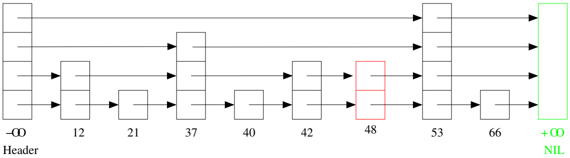
# Insertion in a skip list

Inserting an item $x = 48$



# Insertion in a skip list

Inserting an item $x = 48$



# Implementing skip lists

```
template <typename Key, typename Value>
class Dictionary {
public:
   ...
private:
  struct node_skip_list {
     Key _k;
     Value _v;
     int _height;
     vector<node_skip_list*> _next;

     node_skip_list(const Key& k, const Value& v, int h) :
                   _k(k), _v(v), _height(h),
                   _next(h, nullptr) {
     }
  };
  node_skip_list* _header;
  int _height;
  double _p; // e.g., _p = 0.5
  ...
};
```

# Implementing skip lists

```
template <typename Key, typename Value>
void Dictionary<Key,Value>::lookup(const Key& k,
     bool& exists, Value& v) const throw(error) {

   node_skip_list* p = lookup_skip_list(_header, _height-1, k);
   if (p == nullptr)
      exists = false;
   else {
      exists = true;
      v = p -> _v;
   }
}

template <typename Key, typename Value>
Dictionary<Key,Value>::node_skip_list*
   Dictionary<Key,Value>::lookup_skip_list(
      node_skip_list* p,
      int l, const Key& k) const throw() {

   while (l >= 0)
      if  (p -> _next[l] == nullptr or k <= p ->_next[l] -> _k)
         --l;
      else
         p = p -> _next[l];

   if (p -> _next[0] == nullptr or p -> _next[0] -> _k != k)
      // k is not present
      return nullptr;
   else // k is present, return pointer to the node
      return p -> _next[0];
}
```

# Implementing skip lists

To insert a new item we go through four phases:

1) Search the given key. The search loop is slightly different from before, since we need to keep track of the last node seen at each level before descending from that level to the one immediately below.
2) If the given key is already present we only update the associated value and finish.

# Implementing skip lists

```
template <typename Key, typename Value>
void Dictionary<Key,Value>::insert_skip_list(...) {
   node_skip_list* p = _header;
   int l = _height - 1;
   vector<node_skip_list*> pred(_height);
   while (l >= 0)
      if  (p -> _next[l] == nullptr or k <= p ->_next[l] -> _k) {
         pred[l] = p; // <====== keep track of predecessor at level l
         --l;
      } else {
         p = p -> _next[l];
      }

   if (p -> _next[0] == nullptr or p -> _next[0] -> _k != k) {
      // k is not present, add new node here
      ...
   }
   else // k is present, update associated value
      p -> _next[0] -> _v = v;
}
```

# Implementing skip lists

3) When $k$ is not present, create a new node with $k$ and $v$, and assign a random level $r$ to the new node, using geometric distribution
4) Link the new node in the first $r$ lists, adding empty lists if $r$ is larger than the maximum level of the skip list

# Implementing skip lists

```
template <typename Key, typename Value>
class Dictionary {
public:
   ...
private:
   ...
   Random _rng; // associate a random number generator
                // to the skip list
};

template <typename Key, typename Value>
void Dictionary<Key,Value>::insert_skip_list(...) {

   ...
   // adding new node
   // generate random height
   int h = 1; while (_rng() > _p) ++h;
   node_skip_list* nn = new node_skip_list(k, v, h);
   if (h > _height) {
      // add new levels to the header
      // make pred[i] = _header for all i = _height .. h-1
      ...
   }

   // link the new node to h linked lists
   for (int i = h - 1; i >= 0; --i) {
       nn -> _next[i] = pred[i] -> _next[i];
       pred[i] -> _next[i] = nn;
   }
}
```

# Implementing skip lists

```
...
if (h > _height) {
  node_skip_list* _new_header =
     new node_skip_list(_header -> _k, _header -> _v, h);
  vector<node_skip_list*> new_pred(h, nullptr);

   // copying
   for (int i = _height - 1; i >= 0; --i) {
      _new_header -> _next[i] = _header -> _next[i];
      new_pred[i] = pred[i];
   }

   // empty upper levels
   for (int i = h - 1; i >= _height; --i) {
      _new_header -> _next[i] = nullptr;
      new_pred[i] = _new_header;
   }

   // delete old header
   delete _header;

   // update the header and vector
   // of predecessors
   _header = _new_header;
   pred = new_pred;
   _height = h;
}
...
```

# Performance of skip lists

A preliminary rough analysis considers the search path backwards. Imagine we are at some node $x$ and level $i$:

- The height of $x$ is $> i$ and we come from level $i + 1$ since the sought key $k$ is smaller than the key of the successor of $x$ at level $i + 1$
- The height of $x$ is $i$ and we come from $x$'s predecessor at level $i$ since $k$ is larger or equal to the key at $x$

# Performance of skip lists

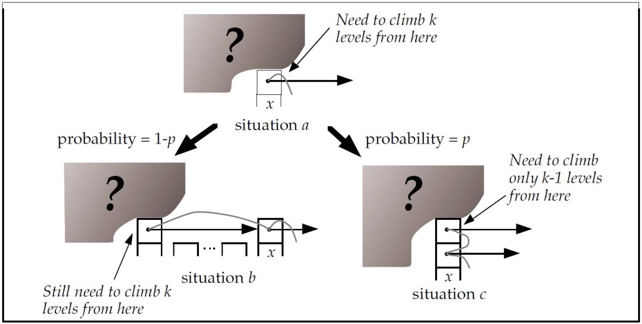



Figure from W. Pugh's *Skip Lists: A Probabilistic Alternative to Balanced Trees* (C. ACM, 1990)—the meaning of $\mathrm{p}$ is the opposite of what we have used!

# Performance of skip lists

The expected number $C(k)$ of steps to “climb” $k$ levels in an infinite list

$$
\begin{aligned}
C(k) &= p(1 + C(k)) + (1 - p)(1 + C(k - 1)) \\
&= 1 + pC(k) + qC(k - 1) = \frac{1}{q}(1 + qC(k - 1)) \\
&= \frac{1}{q} + C(k - 1) = k/q
\end{aligned}
$$

since $C(0) = 0$.

# Performance of skip lists

The analysis above is pessimistic since the list is not infinite and we might “bump” into the header. Then all remaining backward steps to climb up to a level $k$ are vertical—no more horizontal steps. Thus the expected number of steps to climb up to level $L_n$ is

$$\leqslant (L_n - 1)/q$$

# Performance of skip lists

- $L_n$ = the level for which the expected number of nodes that have height $\geqslant L_n$ is $\leqslant 1/q$
- Probability that a node has height $\geqslant k$ is

$$\begin{aligned}\mathbb{P}[\text{height}(x_i) \geqslant k] &= \sum_{i \geqslant k} pq^{i-1} \\ &= pq^{k-1} \sum_{i \geqslant 0} q^i = q^{k-1}\end{aligned}$$

- Number of nodes with height $\geqslant k$ is a binomial r.v. with parameters $n$ and $q^{k-1}$, hence the expected number is $nq^{k-1}$
- Then

$$nq^{L_n - 1} = 1/q \implies L_n = \log_q(1/n) = \log_{1/q} n$$

# Performance of skip lists

- $L_n$ = the level for which the expected number of nodes that have height $\geqslant L_n$ is $\leqslant 1/q$
- Probability that a node has height $\geqslant k$ is

$$\mathbb{P}[\text{height}(x_i) \geqslant k] = \sum_{i \geqslant k} pq^{i-1}$$
$$= pq^{k-1} \sum_{i \geqslant 0} q^i = q^{k-1}$$

- Number of nodes with height $\geqslant k$ is a binomial r.v. with parameters $n$ and $q^{k-1}$, hence the expected number is $nq^{k-1}$
- Then

$$nq^{L_n - 1} = 1/q \implies L_n = \log_q(1/n) = \log_{1/q} n$$

# Performance of skip lists

- $L_n$ = the level for which the expected number of nodes that have height $\geqslant L_n$ is $\leqslant 1/q$
- Probability that a node has height $\geqslant k$ is

$$\begin{aligned}\mathbb{P}[\text{height}(x_i) \geqslant k] &= \sum_{i \geqslant k} pq^{i-1} \\ &= pq^{k-1} \sum_{i \geqslant 0} q^i = q^{k-1}\end{aligned}$$

- Number of nodes with height $\geqslant k$ is a binomial r.v. with parameters $n$ and $q^{k-1}$, hence the expected number is $nq^{k-1}$
- Then

$$nq^{L_n - 1} = 1/q \implies L_n = \log_q(1/n) = \log_{1/q} n$$

# Performance of skip lists

- $L_n$ = the level for which the expected number of nodes that have height $\geqslant L_n$ is $\leqslant 1/q$
- Probability that a node has height $\geqslant k$ is

$$\begin{aligned}\mathbb{P}[\text{height}(x_i) \geqslant k] &= \sum_{i \geqslant k} p q^{i-1} \\ &= p q^{k-1} \sum_{i \geqslant 0} q^i = q^{k-1}\end{aligned}$$

- Number of nodes with height $\geqslant k$ is a binomial r.v. with parameters $n$ and $q^{k-1}$, hence the expected number is $nq^{k-1}$
- Then

$$nq^{L_n - 1} = 1/q \implies L_n = \log_q(1/n) = \log_{1/q} n$$

# Performance of skip lists

Then the expected number of steps remaining to reach $H_n$ (=the height of a random skip list of size $n$) are

- we need $\mathbb{E}[H_n] - L_n$ vertical steps
- we need not more horizontal steps than nodes with height $\geqslant L_n$, the expected number is $\leqslant 1/q$, by definition

# Performance of skip lists

- Recall that the probability that $H_n > k$ is
$$1 - \left(1 - q^k\right)^n \leqslant nq^k$$
- To bound the expected height $\mathbb{E}[H_n]$
$$\begin{aligned}
\mathbb{E}[H_n] &= \sum_{k \geqslant 0} \mathbb{P}[H_n > k] = \sum_{k=0}^{L_n - 1} \mathbb{P}[H_n > k] + \sum_{k=L_n}^{\infty} \mathbb{P}[H_n > k] \\
&\leqslant L_n + \sum_{k=L_n}^{\infty} \mathbb{P}[H_n > k] \leqslant L_n + n \sum_{k=L_n}^{\infty} q^k \\
&= L_n + nq^{L_n} \sum_{k \geqslant 0} q^k = L_n + nq^{L_n} \frac{1}{1-q} = L_n + \frac{1}{p},
\end{aligned}$$
since $nq^{L_n} = 1$, by definition.
- It follows that the expected additional vertical steps need to reach $H_n$ from $L_n$ is
$$\mathbb{E}[H_n] - L_n \leqslant 1/p$$

# Performance of skip lists

Summing up, the expected path length of a search is

$$\leqslant (L_n - 1)/q + 1/q + 1/p = \frac{1}{q}\log_{1/q} n + 1/p$$

On the other hand, the average number of pointers per node is $1/p$ so there is a trade-off between space and time:

- $p \to 0, q \to 1 \implies$ very tall "nodes", short horizontal cost
- $p \to 1, q \to 0 \implies$ flat skip lists
- Pugh suggests $p = 3/4$, optimal choice minimizes factor $(q \ln(1/q))^{-1}$ is
  $q = e^{-1} = 0.36\ldots, p = 1 - e^{-1} \approx 0.632\ldots$

# A more refined analysis

- The cost of insertions, deletions and searches is essentially that of searching, with

$$\text{Cost of search} = \text{\# of forward steps} + \text{height}(S)$$

- More formally, with $X = \{x_1, x_2, \dots, x_n\}$, $x_0 = -\infty < x_1 < \cdots < x_n < x_{n+1} = +\infty$, for $0 \leqslant k \leqslant n$,

$$\begin{aligned} C_{n,k} &= F_{n,k} + H_n \qquad \text{cost of searching a key in } (x_k, x_{k+1}] \\ F_{n,k} &= \text{\# of forward steps to } (x_k, x_{k+1}] \\ H_n &= \text{height of the skip list} \end{aligned}$$

# A more refined analysis

- The cost of insertions, deletions and searches is essentially that of searching, with

$$\text{Cost of search} = \text{\# of forward steps} + \text{height}(S)$$

- More formally, with $X = \{x_1, x_2, \ldots, x_n\}$, $x_0 = -\infty < x_1 < \cdots < x_n < x_{n+1} = +\infty$, for $0 \leqslant k \leqslant n$,

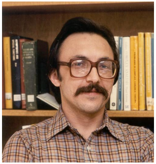
$$\begin{aligned} C_{n,k} &= F_{n,k} + H_n && \text{cost of searching a key in } (x_k, x_{k+1}] \\ F_{n,k} &= \text{\# of forward steps to } (x_k, x_{k+1}] \\ H_n &= \text{height of the skip list} \end{aligned}$$

# Analysis of the height

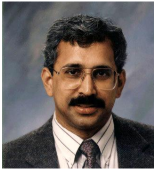
$$
\begin{aligned}
a_i &= \text{height}(x_i) \sim Geom(p) \\
H_n &= \text{height}(S) = \max\{a_1, \ldots, a_n\} \\
\mathbb{E}[H_n] &= \sum_{k>0} \mathbb{P}[H_n > k] = \sum_{k>0} (1 - \mathbb{P}[H_n \leqslant k]) \\
&= \sum_{k>0} \left(1 - \prod_{1 \leqslant i \leqslant n} \mathbb{P}[a_i \leqslant k]\right) = \sum_{k>0} (1 - (\mathbb{P}[a_i \leqslant k])^n) \\
&= \sum_{k>0} \left(1 - \left(1 - q^k\right)^n\right)
\end{aligned}
$$

with $q := 1 - p$.

# Analysis of the height



W. Szpankowski



V. Rego

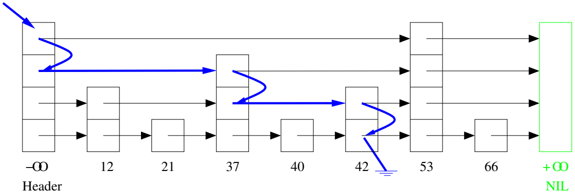
*Theorem (Szpankowski and Rego,1990)*

$$\mathbb{E}[H_n] = \log_Q n + \frac{\gamma}{L} - \frac{1}{2} + \chi(\log_Q n) + O(1/n)$$

*with* $Q := 1/q$, $L := \ln Q$, $\chi(t)$ *a fluctuation of period 1, mean 0 and small amplitude.*

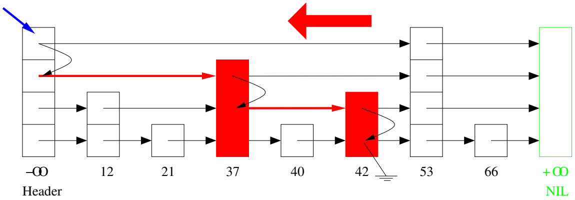
# Analysis of the forward cost

The number of forward steps $F_{n,k}$ is the number of weak left-to-right maxima in $a_k, a_{k-1}, \ldots, a_1$, with $a_i = \text{height}(x_i)$



# Analysis of the forward cost

The number of forward steps $F_{n,k}$ is the number of weak left-to-right maxima in $a_k, a_{k-1}, \ldots, a_1$, with $a_i = \text{height}(x_i)$



# Analysis of the forward cost

- Total unsuccessful search cost

$$C_n = \sum_{0 \leqslant k \leqslant n} C_{n,k} = nH_n + F_n$$

- Total forward cost

$$F_n = \sum_{0 \leqslant k \leqslant n} F_{n,k}$$

# Analysis of the forward cost

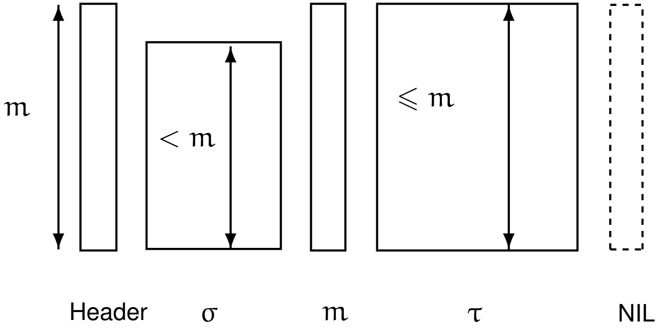
- Total unsuccessful search cost

$$C_n = \sum_{0 \leqslant k \leqslant n} C_{n,k} = nH_n + F_n$$

- Total forward cost

$$F_n = \sum_{0 \leqslant k \leqslant n} F_{n,k}$$

# Analysis of the forward cost




A recursive decomposition of the skip list $S$

# Analysis of the forward cost

- $F(S)$ = total forward cost of the skip list $S$
- The recursive decomposition $S = \langle \sigma, m, \tau \rangle$ gives

$$F(S) = F(\sigma) + F(\tau) + |\tau| + 1$$

- Then the expected forward cost can be calculated as

$$\mathbb{E}[F_n] = \sum_{S\text{:skip list of size } n} F(S) \cdot \mathbb{P}[S]$$

with

$$\mathbb{P}[S] = \mathbb{P}[\sigma] \cdot pq^{m-1} \cdot \mathbb{P}[\tau]$$

# Analysis of the forward cost

- $F(S)$ = total forward cost of the skip list $S$
- The recursive decomposition $S = \langle \sigma, m, \tau \rangle$ gives

$$F(S) = F(\sigma) + F(\tau) + |\tau| + 1$$

- Then the expected forward cost can be calculated as

$$\mathbb{E}[F_n] = \sum_{S\text{:skip list of size } n} F(S) \cdot \mathbb{P}[S]$$

with

$$\mathbb{P}[S] = \mathbb{P}[\sigma] \cdot pq^{m-1} \cdot \mathbb{P}[\tau]$$

# Analysis of the forward cost

- $F(S)$ = total forward cost of the skip list $S$
- The recursive decomposition $S = \langle \sigma, m, \tau \rangle$ gives

$$F(S) = F(\sigma) + F(\tau) + |\tau| + 1$$

- Then the expected forward cost can be calculated as

$$\mathbb{E}[F_n] = \sum_{S\text{:skip list of size } n} F(S) \cdot \mathbb{P}[S]$$

with

$$\mathbb{P}[S] = \mathbb{P}[\sigma] \cdot pq^{m-1} \cdot \mathbb{P}[\tau]$$

# Analysis of the forward cost

- The recurrence is complicated but can be solved exactly

$$\mathbb{E}[F_n] = \frac{p}{q} \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{1}{Q^{k-1} - 1},$$

$$q := 1 - p, Q := 1/q$$

- And then its asymptotic behavior analyzed using the same techniques as in the analysis of $\mathbb{E}[H_n]$

# Analysis of the forward cost

- The recurrence is complicated but can be solved exactly

$$\mathbb{E}[F_n] = \frac{p}{q} \sum_{k=2}^{n} \binom{n}{k} (-1)^k \frac{1}{Q^{k-1} - 1},$$

$$q := 1 - p, Q := 1/q$$

- And then its asymptotic behavior analyzed using the same techniques as in the analysis of $\mathbb{E}[H_n]$

# Analysis of the forward cost

P. Kirschenhofer H. Prodinger

*Theorem (Kirschehofer, Prodinger, 1994)*

*The expected forward cost in a random skip list of size $n$ is*

$$\mathbb{E}[F_n] = (Q-1)n\left(\log_Q n + \frac{\gamma - 1}{L} - \frac{1}{2} + \frac{1}{L}\chi(\log_Q n)\right) + O(\log n),$$

*with $Q := 1/q$, $L = \ln Q$ and $\chi$ a periodic fluctuation of period 1, mean 0 and small amplitude.*

# To learn more


[1] L. Devroye.
A limit theory for random skip lists.
*The Annals of Applied Probability*, 2(3):597–609, 1992.

[2] P. Kirschenhofer and H. Prodinger.
The path length of random skip lists.
*Acta Informatica*, 31(8):775–792, 1994.

[3] P. Kirschenhofer, C. Martínez and H. Prodinger.
Analysis of an Optimized Search Algorithm for Skip Lists.
*Theoretical Computer Science*, 144:199–220, 1995.

# To learn more (2)


[1] T. Papadakis, J. I. Munro, and P. V. Poblete.
Average search and update costs in skip lists.
*BIT*, 32:316–332, 1992.

[2] H. Prodinger.
Combinatorics of geometrically distributed random variables: Left-to-right maxima.
*Discrete Mathematics*, 153:253–270, 1996.

[3] W. Pugh.
Skip lists: a probabilistic alternative to balanced trees.
*Comm. ACM*, 33(6):668–676, 1990.