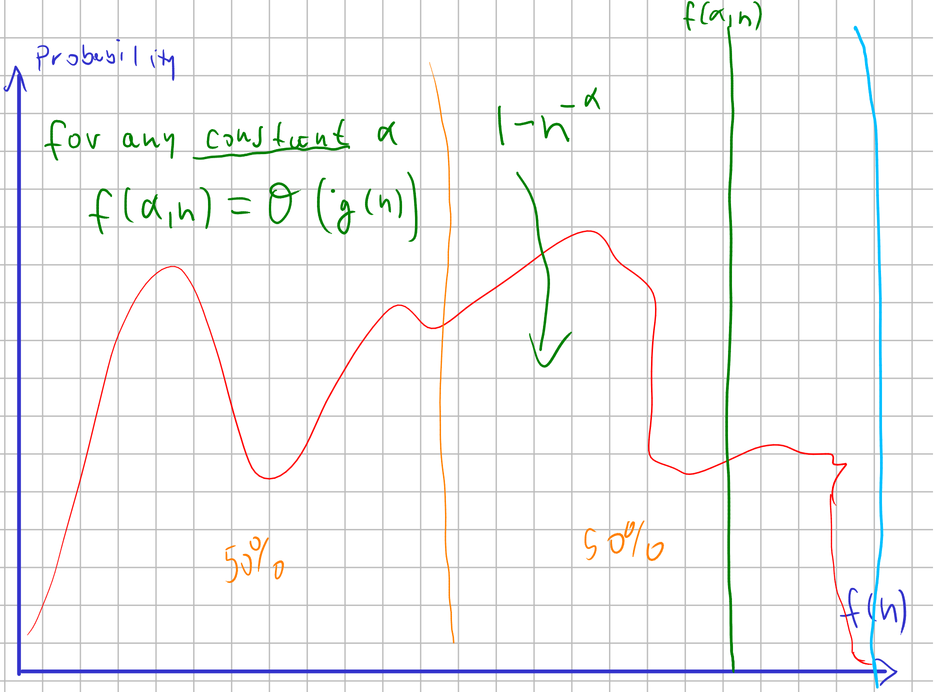
High Probability

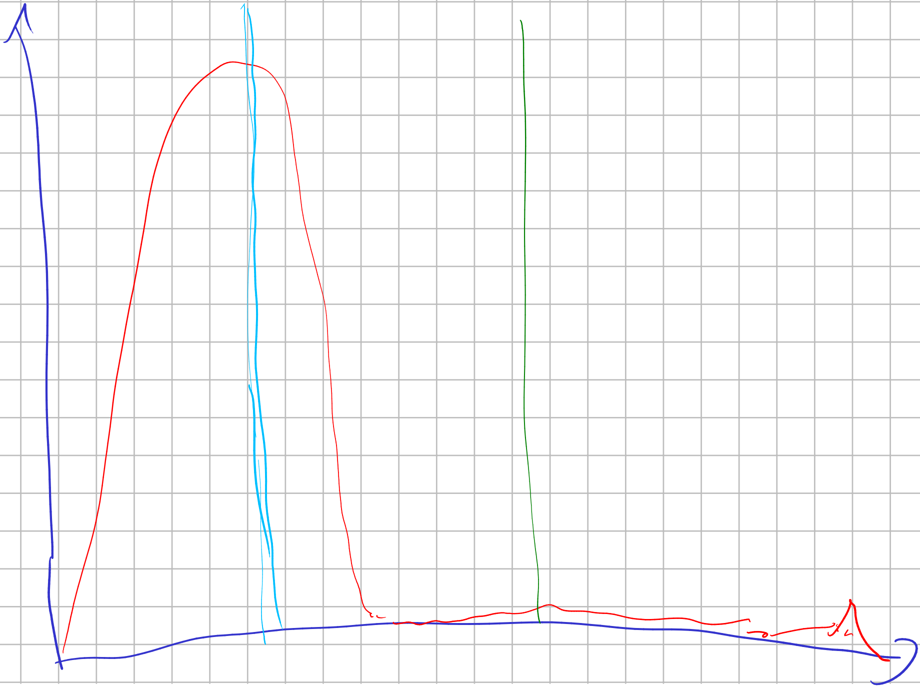
Definition 18 (High Probability)

We say a **randomized** algorithm has running time $\mathcal{O}(\log n)$ with high probability if for any constant α the running time is at most $\mathcal{O}(\underbrace{\log n}_{g(n)})$ with probability at least $1 - \frac{1}{n^\alpha}$.

Here the $\mathcal{O}$-notation hides a constant that may depend on α.



Probability
for any constant α
$f(\alpha, n) = \Theta(g(n))$
$1 - n^{-\alpha}$
$f(\alpha, n)$
50%
50%
$f(n)$



High Probability

Suppose there are polynomially many events $E_1, E_2, \ldots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

High Probability

Suppose there are polynomially many events $E_1, E_2, \dots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

Then the probability that all E_i hold is at least

$$\Pr[E_1 \wedge \cdots \wedge E_\ell]$$

High Probability

Suppose there are polynomially many events $E_1, E_2, \ldots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

Then the probability that all E_i hold is at least

$$\Pr[E_1 \wedge \cdots \wedge E_\ell] = 1 - \Pr[\bar{E}_1 \vee \cdots \vee \bar{E}_\ell]$$

High Probability

Suppose there are polynomially many events $E_1, E_2, \ldots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

Then the probability that all E_i hold is at least

$$\begin{aligned}\Pr[E_1 \wedge \cdots \wedge E_\ell] &= 1 - \Pr[\bar{E}_1 \vee \cdots \vee \bar{E}_\ell] \\ &\geq 1 - n^c \cdot n^{-\alpha}\end{aligned}$$

High Probability

Suppose there are polynomially many events $E_1, E_2, \ldots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

Then the probability that all E_i hold is at least

$$\begin{aligned}
\Pr[E_1 \wedge \cdots \wedge E_\ell] &= 1 - \Pr[\bar{E}_1 \vee \cdots \vee \bar{E}_\ell] \\
&\geq 1 - n^c \cdot n^{-\alpha} \\
&= 1 - n^{c-\alpha} \ .
\end{aligned}$$

High Probability

Suppose there are polynomially many events $E_1, E_2, \dots, E_\ell$, $\ell = n^c$ each holding with high probability (e.g. E_i may be the event that the i-th search in a skip list takes time at most $\mathcal{O}(\log n)$).

$f(\alpha', n) = \Theta(\log n)$

Then the probability that all E_i hold is at least

given α

$\alpha' := \alpha + c$

$$\begin{aligned}\Pr[E_1 \wedge \dots \wedge E_\ell] &= 1 - \Pr[\bar{E}_1 \vee \dots \vee \bar{E}_\ell] \\ &\geq 1 - n^c \cdot n^{-\alpha'} \\ &= 1 - n^{c-\alpha'} . \\ &= 1 - n^{-\alpha}\end{aligned}$$

This means $\Pr[E_1 \wedge \dots \wedge E_\ell]$ holds with high probability.

7.5 Skip Lists

Lemma 19

A search (and, hence, also insert and delete) in a skip list with n *elements takes time* $\mathcal{O}(\log n)$ *with high probability (w. h. p.).*

7.5 Skip Lists

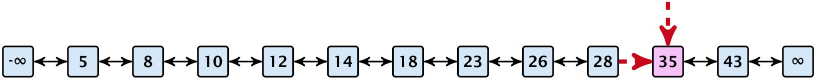
Backward analysis:

7.5 Skip Lists

Backward analysis:

7.5 Skip Lists

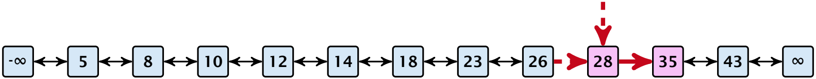
Backward analysis:



Backward analysis:

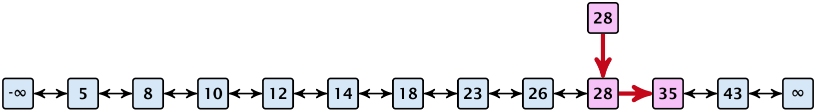
7.5 Skip Lists

Backward analysis:



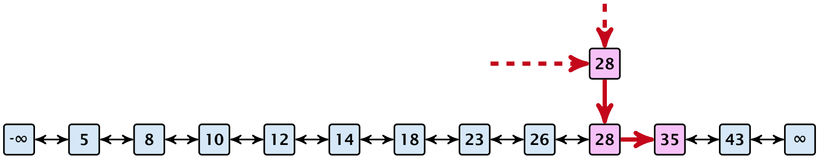
7.5 Skip Lists

Backward analysis:



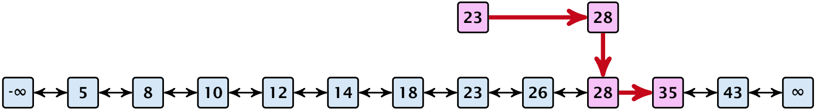
7.5 Skip Lists

Backward analysis:



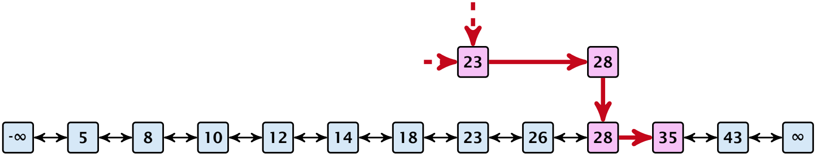
7.5 Skip Lists

Backward analysis:



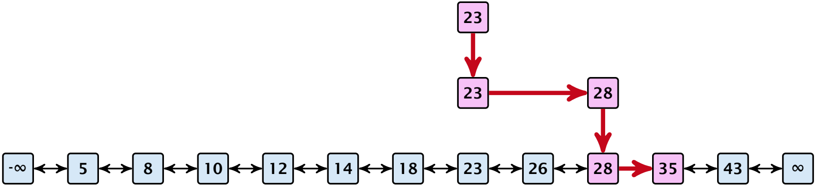
7.5 Skip Lists

Backward analysis:



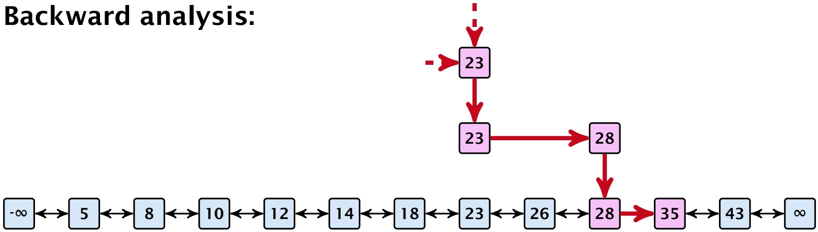
7.5 Skip Lists

Backward analysis:



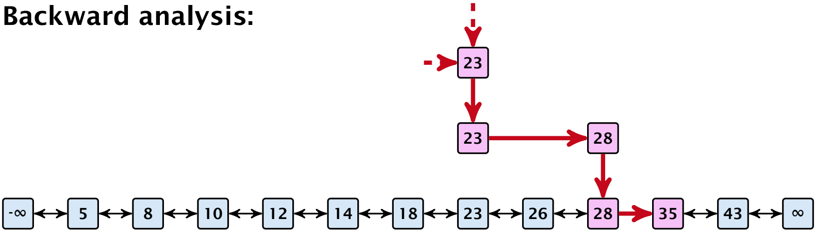
7.5 Skip Lists

Backward analysis:



7.5 Skip Lists

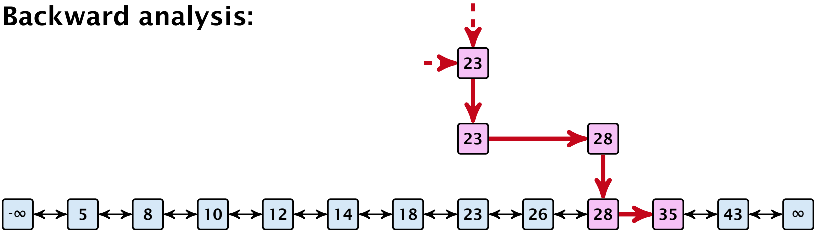
Backward analysis:



At each point the path goes up with probability $1/2$ and left with probability $1/2$.

7.5 Skip Lists

Backward analysis:



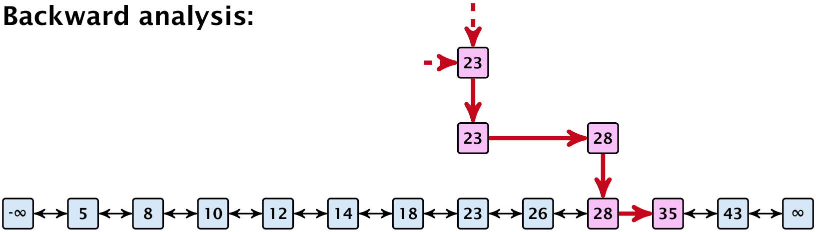
At each point the path goes up with probability $1/2$ and left with probability $1/2$.

We show that w.h.p:

- A "long" search path must also go very high.

7.5 Skip Lists

Backward analysis:



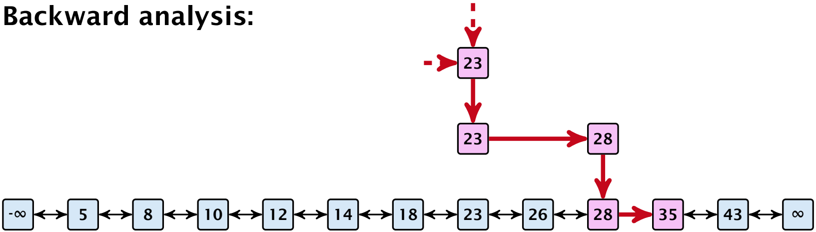
At each point the path goes up with probability $1/2$ and left with probability $1/2$.

We show that w.h.p:

- A “long” search path must also go very high.
- There are no elements in high lists.

7.5 Skip Lists

Backward analysis:



At each point the path goes up with probability $1/2$ and left with probability $1/2$.

We show that w.h.p:

- A "long" search path must also go very high.
- There are no elements in high lists.

From this it follows that w.h.p. there are no long paths.

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

7.5 Skip Lists

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1}$$

$$\left(\frac{n}{k}\right)^k \le \frac{n}{k} \quad \underbrace{\frac{n-1}{k-1}}_{\le \frac{n}{k}} \quad \frac{n-2}{k-2}$$

7.5 Skip Lists

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \ge \left(\frac{n}{k}\right)^k$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \geq \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \geq \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \ge \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!} \le \frac{n^k}{k!}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \geq \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!} \leq \frac{n^k}{k!} = \frac{n^k \cdot k^k}{k^k \cdot k!}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \ge \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!} \le \frac{n^k}{k!} = \frac{n^k \cdot k^k}{k^k \cdot k!}$$

$$= \left(\frac{n}{k}\right)^k \cdot \frac{k^k}{k!}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \ge \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!} \le \frac{n^k}{k!} = \frac{n^k \cdot k^k}{k^k \cdot k!}$$

$$= \left(\frac{n}{k}\right)^k \cdot \frac{k^k}{k!} \le \left(\frac{n}{k}\right)^k \cdot \sum_{i \ge 0} \frac{k^i}{i!}$$

Estimation for Binomial Coefficients

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot \ldots \cdot (n-k+1)}{k \cdot \ldots \cdot 1} \ge \left(\frac{n}{k}\right)^k$$

$$\binom{n}{k} = \frac{n \cdot \ldots \cdot (n-k+1)}{k!} \le \frac{n^k}{k!} = \frac{n^k \cdot k^k}{k^k \cdot k!}$$

$$= \left(\frac{n}{k}\right)^k \cdot \frac{k^k}{k!} \le \left(\frac{n}{k}\right)^k \cdot \sum_{i \ge 0} \frac{k^i}{i!} = \left(\frac{en}{k}\right)^k$$

7.5 Skip Lists

7.5 Skip Lists

Let $E_{z,k}$ denote the event that a search path is of length z (number of edges) but does not visit a list above L_k.

7.5 Skip Lists

Let $E_{z,k}$ denote the event that a search path is of length z (number of edges) but does not visit a list above L_k.

In particular, this means that during the construction in the backward analysis we see at most k heads (i.e., coin flips that tell you to go up) in z trials.

7.5 Skip Lists

$$\Pr[E_{z,k}]$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \le \Pr[\overset{A}{\text{at most } k \text{ heads in } z \text{ trials}}] \le \Pr\Big[\bigvee_{\vec m} E_{\vec m}\Big]$$

$$\le \binom{z}{k} 2^{-(z-k)} \qquad \le \sum_{\vec m} \Pr[E_{\vec m}]$$

mark $z-k$ positions

$\Pr[\text{only see tails in marked pos.}] = 2^{-(z-k)}$

$\vec m$ is marking

$E_{\vec m}$: event only tails in marked pos of $\vec m$

7.5 Skip Lists

$$\Pr[E_{z,k}] \le \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\le \binom{z}{k} 2^{-(z-k)} \le \left(\frac{ez}{k}\right)^k 2^{-(z-k)}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \le \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\le \binom{z}{k} 2^{-(z-k)} \le \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \le \left(\frac{2ez}{k}\right)^k 2^{-z}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\leq \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha}$$

$2^{-\alpha\gamma \log n}$

$n^{-\gamma\alpha}$

$2^{-\beta\gamma \log n} =$

7.5 Skip Lists

$$\Pr[E_{z,k}] \le \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\le \binom{z}{k} 2^{-(z-k)} \le \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \le \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \ge 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\le \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \le \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\leq \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \leq \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

$$\leq \left(\frac{2e(\beta + \alpha)}{2^\beta}\right)^k n^{-\alpha}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\leq \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \leq \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

$$\leq \left(\frac{2e(\beta + \alpha)}{2^\beta}\right)^k n^{-\alpha}$$

now choosing $\beta = 6\alpha$ gives

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\leq \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \leq \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

$$\leq \left(\frac{2e(\beta+\alpha)}{2^\beta}\right)^k n^{-\alpha}$$

now choosing $\beta = 6\alpha$ gives

$$\leq \left(\frac{42\alpha}{64^\alpha}\right)^k n^{-\alpha}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \leq \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\leq \binom{z}{k} 2^{-(z-k)} \leq \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \leq \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \geq 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\leq \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \leq \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

$$\leq \left(\frac{2e(\beta + \alpha)}{2^\beta}\right)^k n^{-\alpha}$$

now choosing $\beta = 6\alpha$ gives

$$\leq \left(\frac{42\alpha}{64^\alpha}\right)^k n^{-\alpha} \leq n^{-\alpha}$$

7.5 Skip Lists

$$\Pr[E_{z,k}] \le \Pr[\text{at most } k \text{ heads in } z \text{ trials}]$$

$$\le \binom{z}{k} 2^{-(z-k)} \le \left(\frac{ez}{k}\right)^k 2^{-(z-k)} \le \left(\frac{2ez}{k}\right)^k 2^{-z}$$

choosing $k = \gamma \log n$ with $\gamma \ge 1$ and $z = (\beta + \alpha)\gamma \log n$

$$\le \left(\frac{2ez}{k}\right)^k 2^{-\beta k} \cdot n^{-\gamma\alpha} \le \left(\frac{2ez}{2^\beta k}\right)^k \cdot n^{-\alpha}$$

$$\le \left(\frac{2e(\beta+\alpha)}{2^\beta}\right)^k n^{-\alpha}$$

now choosing $\beta = 6\alpha$ gives

$$\le \left(\frac{42\alpha}{64^\alpha}\right)^k n^{-\alpha} \le n^{-\alpha}$$

for $\alpha \ge 1$.

7.5 Skip Lists

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n2^{-(k+1)} \leq n^{-(\gamma-1)} .$$

$$\leq n \cdot 2^{-k} \leq n \cdot n^{-\gamma}$$

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n2^{-(k+1)} \leq n^{-(\gamma-1)} \ .$$

For the search to take at least $z = 7\alpha\gamma \log n$ steps either the event $E_{z,k}$ or the event A_{k+1} must hold.

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n2^{-(k+1)} \leq n^{-(\gamma-1)} \ .$$

For the search to take at least $z = 7\alpha\gamma \log n$ steps either the event $E_{z,k}$ or the event A_{k+1} must hold.
Hence,

$$\Pr[\text{search requires } z \text{ steps}]$$

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n2^{-(k+1)} \leq n^{-(\gamma-1)} \ .$$

For the search to take at least $z = 7\alpha\gamma \log n$ steps either the event $E_{z,k}$ or the event A_{k+1} must hold.
Hence,

$$\Pr[\text{search requires } z \text{ steps}] \leq \Pr[E_{z,k}] + \Pr[A_{k+1}]$$

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = 7\alpha\gamma \log n$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n2^{-(k+1)} \leq n^{-(\gamma-1)} \ .$$

For the search to take at least $z = 7\alpha\gamma \log n$ steps either the event $E_{z,k}$ or the event A_{k+1} must hold.
Hence,

$$\begin{aligned} \Pr[\text{search requires } z \text{ steps}] &\leq \Pr[E_{z,k}] + \Pr[A_{k+1}] \\ &\leq n^{-\alpha} + n^{-(\gamma-1)} \end{aligned}$$

7.5 Skip Lists

So far we fixed $k = \gamma \log n$, $\gamma \geq 1$, and $z = \underline{7\alpha\gamma \log n}$, $\alpha \geq 1$.

This means that a search path of length $\Omega(\log n)$ visits a list on a level $\Omega(\log n)$, w.h.p.

Let A_{k+1} denote the event that the list L_{k+1} is non-empty. Then

$$\Pr[A_{k+1}] \leq n 2^{-(k+1)} \leq n^{-(\gamma-1)} \ .$$

For the search to take at least $z = 7\alpha\gamma \log n$ steps either the event $E_{z,k}$ or the event A_{k+1} must hold.
Hence,

$$\begin{aligned} \Pr[\text{search requires } z \text{ steps}] &\leq \Pr[E_{z,k}] + \Pr[A_{k+1}] \\ &\leq n^{-\alpha} + n^{-(\gamma-1)} \end{aligned}$$

This means, the search requires ~~at most z~~ $O(\log n)$ steps, w. h. p.

$$n^{-\alpha} + n^{1-\gamma}$$

$$\leq \max\{2n^{-\alpha}, 2n^{1-\gamma}\}$$

$$2n^{-\alpha} \leq n^{-\delta}$$

$\alpha = \delta + 1$

$\gamma = \delta + 2$

$$n^{-\delta} \qquad (10^6)^{-\delta}$$

7.6 Hashing

Dictionary:

- $S.\mathbf{insert}(x)$: Insert an element x.
- $S.\mathbf{delete}(x)$: Delete the element pointed to by x.
- $S.\mathbf{search}(k)$: Return a pointer to an element e with $\mathrm{key}[e] = k$ in S if it exists; otherwise return null.

So far we have implemented the search for a key by carefully choosing split-elements.

Then the memory location of an object x with key k is determined by successively comparing k to split-elements.

Hashing tries to directly compute the memory location from the given key. The goal is to have constant search time.

7.6 Hashing

Dictionary:

- $\boldsymbol{S}.\,\textbf{insert}(\boldsymbol{x})$: Insert an element x.
- $\boldsymbol{S}.\,\textbf{delete}(\boldsymbol{x})$: Delete the element pointed to by x.
- $\boldsymbol{S}.\,\textbf{search}(\boldsymbol{k})$: Return a pointer to an element e with $\text{key}[e] = k$ in S if it exists; otherwise return null.

So far we have implemented the search for a key by carefully choosing split-elements.

Then the memory location of an object x with key k is determined by successively comparing k to split-elements.

Hashing tries to directly compute the memory location from the given key. The goal is to have constant search time.

7.6 Hashing

Dictionary:

- $S.\textbf{insert}(x)$: Insert an element x.
- $S.\textbf{delete}(x)$: Delete the element pointed to by x.
- $S.\textbf{search}(k)$: Return a pointer to an element e with $\mathrm{key}[e] = k$ in S if it exists; otherwise return null.

So far we have implemented the search for a key by carefully choosing split-elements.

Then the memory location of an object x with key k is determined by successively comparing k to split-elements.

Hashing tries to directly compute the memory location from the given key. The goal is to have constant search time.

7.6 Hashing

Dictionary:

- $S.\textbf{insert}(x)$: Insert an element x.
- $S.\textbf{delete}(x)$: Delete the element pointed to by x.
- $S.\textbf{search}(k)$: Return a pointer to an element e with $\mathrm{key}[e] = k$ in S if it exists; otherwise return null.

So far we have implemented the search for a key by carefully choosing split-elements.

Then the memory location of an object x with key k is determined by successively comparing k to split-elements.

Hashing tries to directly compute the memory location from the given key. The goal is to have constant search time.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \dots, n-1]$ hash-table.
- Hash function $h : U \to [0, \dots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \le |U|$.
- Array $T[0, \ldots, n-1]$ hash-table.
- Hash function $h : U \to [0, \ldots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \ldots, n-1]$ hash-table.
- Hash function $h : U \to [0, \ldots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \le |U|$.
- Array $T[0, \dots, n-1]$ hash-table.
- Hash function $h : U \to [0, \dots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \ldots, n-1]$ hash-table.
- Hash function $h : U \to [0, \ldots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \ldots, n-1]$ hash-table.
- Hash function $h : U \to [0, \ldots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \dots, n-1]$ hash-table.
- Hash function $h : U \to [0, \dots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

7.6 Hashing

Definitions:

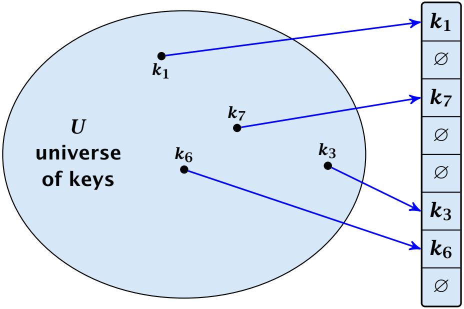
- Universe U of keys, e.g., $U \subseteq \mathbb{N}_0$. U very large.
- Set $S \subseteq U$ of keys, $|S| = m \leq |U|$.
- Array $T[0, \ldots, n-1]$ hash-table.
- Hash function $h : U \to [0, \ldots, n-1]$.

The hash-function h should fulfill:

- Fast to evaluate.
- Small storage requirement.
- Good distribution of elements over the whole table.

Direct Addressing

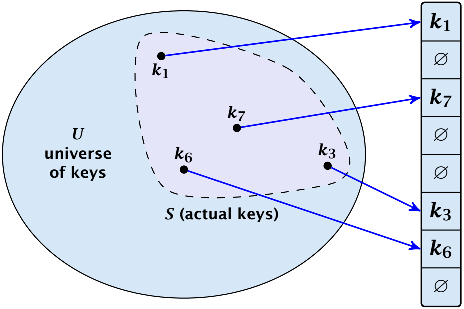
Ideally the hash function maps all keys to different memory locations.



This special case is known as Direct Addressing. It is usually very unrealistic as the universe of keys typically is quite large, and in particular larger than the available memory.

Perfect Hashing

Suppose that we know the set S of actual keys (no insert/no delete). Then we may want to design a simple hash-function that maps all these keys to different memory locations.



Such a hash function h is called a perfect hash function for set S.

Collisions

If we do not know the keys in advance, the best we can hope for is that the hash function distributes keys evenly across the table.

Problem: Collisions

Usually the universe U is much larger than the table-size n.

Hence, there may be two elements k_1, k_2 from the set S that map to the same memory location (i.e., $h(k_1) = h(k_2)$). This is called a collision.

Collisions

If we do not know the keys in advance, the best we can hope for is that the hash function distributes keys evenly across the table.

Problem: Collisions
Usually the universe U is much larger than the table-size n.

Hence, there may be two elements k_1, k_2 from the set S that map to the same memory location (i.e., $h(k_1) = h(k_2)$). This is called a collision.

Collisions

If we do not know the keys in advance, the best we can hope for is that the hash function distributes keys evenly across the table.

Problem: Collisions
Usually the universe U is much larger than the table-size n.

Hence, there may be two elements k_1, k_2 from the set S that map to the same memory location (i.e., $h(k_1) = h(k_2)$). This is called a collision.

Collisions

Typically, collisions do not appear once the size of the set S of actual keys gets close to n, but already when $|S| \geq \omega(\sqrt{n})$.

Lemma 20

The probability of having a collision when hashing m elements into a table of size n under uniform hashing is at least

$$1 - e^{-\frac{m(m-1)}{2n}} \approx 1 - e^{-\frac{m^2}{2n}} .$$

Uniform hashing:

Choose a hash function uniformly at random from all functions $f : U \rightarrow [0, \ldots, n-1]$.

Collisions

Typically, collisions do not appear once the size of the set S of actual keys gets close to n, but already when $|S| \geq \omega(\sqrt{n})$.

Lemma 20

The probability of having a collision when hashing m elements into a table of size n under uniform hashing is at least

$$1 - e^{-\frac{m(m-1)}{2n}} \approx 1 - e^{-\frac{m^2}{2n}} \ .$$

Uniform hashing:

Choose a hash function uniformly at random from all functions $f : U \to [0, \ldots, n-1]$.

Collisions

Typically, collisions do not appear once the size of the set S of actual keys gets close to n, but already when $|S| \geq \omega(\sqrt{n})$.

Lemma 20

The probability of having a collision when hashing m elements into a table of size n under uniform hashing is at least

$$1 - e^{-\frac{m(m-1)}{2n}} \approx 1 - e^{-\frac{m^2}{2n}} \; .$$

Uniform hashing:

Choose a hash function uniformly at random from all functions $f : U \to [0, \ldots, n-1]$.

$n^{|U|}$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\Pr[A_{m,n}]$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\Pr[A_{m,n}] = \prod_{\ell=1}^{m} \frac{n-\ell+1}{n}$$

$$1 \quad \frac{n-(\ell-1)}{n}$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$\leq e^{-j/n}$

$$\Pr[A_{m,n}] = \prod_{\ell=1}^{m} \frac{n-\ell+1}{n} = \prod_{j=0}^{m-1} \left(1 - \frac{j}{n}\right)$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\Pr[A_{m,n}] = \prod_{\ell=1}^{m} \frac{n-\ell+1}{n} = \prod_{j=0}^{m-1} \Bigl(1-\frac{j}{n}\Bigr)$$

$$\leq \prod_{j=0}^{m-1} e^{-j/n}$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\begin{aligned}\Pr[A_{m,n}] &= \prod_{\ell=1}^{m} \frac{n-\ell+1}{n} = \prod_{j=0}^{m-1}\left(1-\frac{j}{n}\right)\\ &\le \prod_{j=0}^{m-1} e^{-j/n} = e^{-\sum_{j=0}^{m-1}\frac{j}{n}}\end{aligned}$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\Pr[A_{m,n}] = \prod_{\ell=1}^{m} \frac{n-\ell+1}{n} = \prod_{j=0}^{m-1}\Big(1-\frac{j}{n}\Big)$$

$$\leq \prod_{j=0}^{m-1} e^{-j/n} = e^{-\sum_{j=0}^{m-1}\frac{j}{n}} = e^{-\frac{m(m-1)}{2n}} \quad .$$

Collisions

Proof.

Let $A_{m,n}$ denote the event that inserting m keys into a table of size n does not generate a collision. Then

$$\Pr[A_{m,n}] = \prod_{\ell=1}^{m} \frac{n-\ell+1}{n} = \prod_{j=0}^{m-1} \Big(1 - \frac{j}{n}\Big)$$
$$\leq \prod_{j=0}^{m-1} e^{-j/n} = e^{-\sum_{j=0}^{m-1} \frac{j}{n}} = e^{-\frac{m(m-1)}{2n}} \;.$$

Here the first equality follows since the ℓ-th element that is hashed has a probability of $\frac{n-\ell+1}{n}$ to not generate a collision under the condition that the previous elements did not induce collisions. □