Advanced Algorithms
COMP4121

Aleks Ignjatovic

School of Computer Science and Engineering
University of New South Wales

Skip Lists

Skip Lists

- A recent data structure introduced in 1989 by William Pugh.
- Yes, I know, it is recent for people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees).
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

Skip Lists

- A recent data structure introduced in 1989 by William Pugh.
- Yes, I know, it is recent for people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees).
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

- A recent data stru roduced in 1989 by William Pugh.
- Yes, I know, it is re people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

Skip Lists

- A recent data structure introduced in 1989 by William Pugh.
- Yes, I know, it is recent for people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

- A recent data structure introduced in 1989 by William Pugh.
- Yes, I know, it is recent for people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

Skip Lists

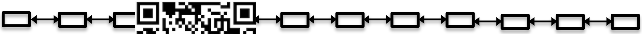
- A recent data struc[illegible]roduced in 1989 by William Pugh.
- Yes, I know, it is re[illegible]people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

- A recent data structure introduced in 1989 by William Pugh.
- Yes, I know, it is recent for people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

- A recent data struc… …roduced in 1989 by William Pugh.
- Yes, I know, it is re… …people of my age, not at all recent for you ...
- A randomised data structure with benefits of balanced trees (e.g., AVL or Red - Black trees):
  - $O(\log n)$ **expected** time for INSERT and SEARCH;
  - $O(1)$ time for MIN, MAX, SUCC, PRED;
  - Can be enhanced so that finding the k^{th} element in the list also runs in $O(\log n)$ time.
- Much easier to code and in practice also tend to be faster and use less space than balanced trees.
- Skip Lists have replaced balanced trees in many applications.

Skip Lists

- Consider a doubly linked list:

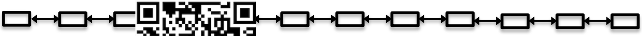


- MIN, MAX, SUCC, PRED run in time $O(1)$.
- However, SEARCH, INSERT, DELETE run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels.

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Consider a doubly linked list:

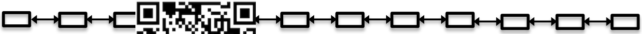


- MIN, MAX, SUCC, PRED run in time $O(1)$.
- However, SEARCH, INSERT, DELETE run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Consider a doubly linked list:

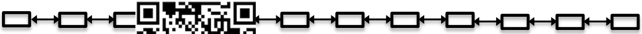


- MIN, MAX, SUCC, PRED run in time $O(1)$.
- However, SEARCH, INSERT, DELETE run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels.

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

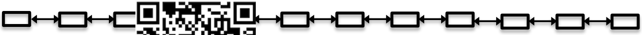
- Consider a doubly linked list:



- Min, Max, Succ, [illegible] n in time $O(1)$.
- However, Search, Insert, Delete run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels.

- This is something like the express elevators in skyscrapers which do not stop on every floor.

- Consider a doubly linked list:

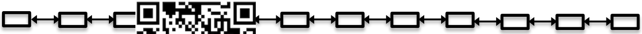


- Min, Max, Succ, [illegible]n in time $O(1)$.
- However, Search, Insert, Delete run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels.

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Consider a doubly linked list:



- MIN, MAX, SUCC, PRED run in time $O(1)$.
- However, SEARCH, INSERT, DELETE run in time $O(n)$.
- The culprit is searching.

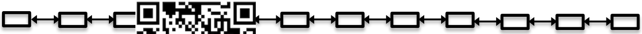
- Can we modify doubly linked links to make search $O(\log n)$ expected time?

- **Idea:** make shortcuts on several levels:

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Consider a doubly linked list:

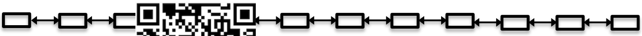


- MIN, MAX, SUCC, PRED run in time $O(1)$.
- However, SEARCH, INSERT, DELETE run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels:

- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Consider a doubly linked list:



- Min, Max, Succ, [illegible]n in time $O(1)$.
- However, Search, Insert, Delete run in time $O(n)$.
- The culprit is searching.
- Can we modify doubly linked links to make search $O(\log n)$ expected time?
- **Idea:** make shortcuts on several levels:

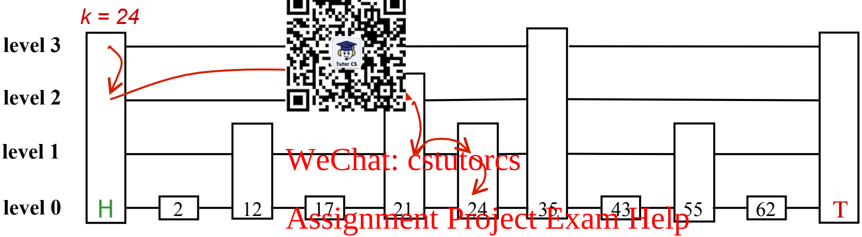
- This is something like the express elevators in skyscrapers which do not stop on every floor.

Skip Lists

- Searching for k:

- Start from the head H and go as far right as you can, without exceeding k, using the highest possible level of links;
- then drop one level down and repeat the procedure using lower level links.
- How can we ensure such a search procedure runs in time $O(\log n)$?
- Can we link every other link on the second level, every fourth link on the third level, every eight on the fourth level and so on...

- Searching for k:



- Start from the head H and go as far right as you can, without exceeding k, using the highest possible level of links;
- then drop one level down and repeat the procedure using lower level links.
- How can we ensure such a search procedure runs in time $O(\log n)$?
- Can we link every other link on the second level, every fourth link on the third level, every eight on the fourth level and so on...

Skip Lists

- Searching for k:

- Start from the head H and go as far right as you can, without exceeding k, using the highest possible level of links;
- then drop one level down and repeat the procedure using lower level links.
- How can we ensure such a search procedure runs in time $O(\log n)$?
- Can we link every other link on the second level, every fourth link on the third level, every eight on the fourth level and so on...

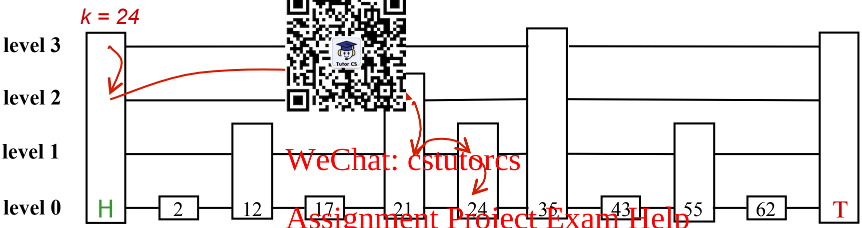
Skip Lists

- Searching for k:

- Start from the head H and go as far right as you can, without exceeding k, using the highest possible level of links;
- then drop one level down and repeat the procedure using lower level links.
- How can we ensure such a search procedure runs in time $O(\log n)$?
- Can we link every other link on the second level, every fourth link on the third level, every eight on the fourth level and so on...

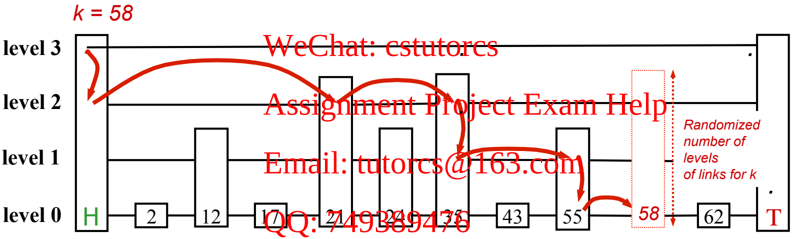
Skip Lists

- Searching for k:



- Start from the head H and go as far right as you can, without exceeding k, using the highest possible level of links;
- then drop one level down and repeat the procedure using lower level links.
- How can we ensure such a search procedure runs in time $O(\log n)$?
- Can we link every other link on the second level, every fourth link on the third level, every eight on the fourth level and so on...

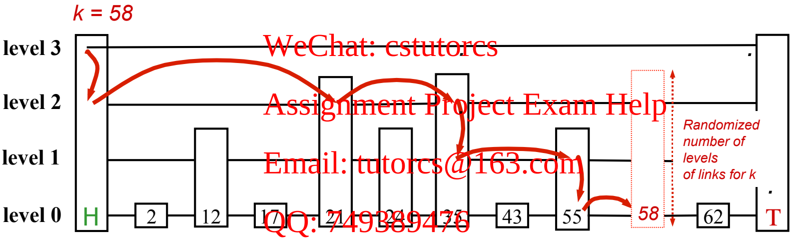
- Problem: insertions and deletions will destroy such a structure...
- We need dynamica lancing structure.
- This is where rand comes into play, ensuring that in the long run the struct ns (essentially) balanced.



- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

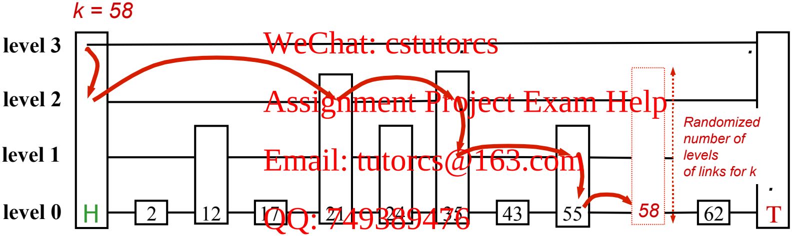
Skip Lists

- Problem: insertions and deletions will destroy such a structure...
- We need dynamically self-balancing structure.
- This is where randomization comes into play, ensuring that in the long run the structure remains (essentially) balanced.



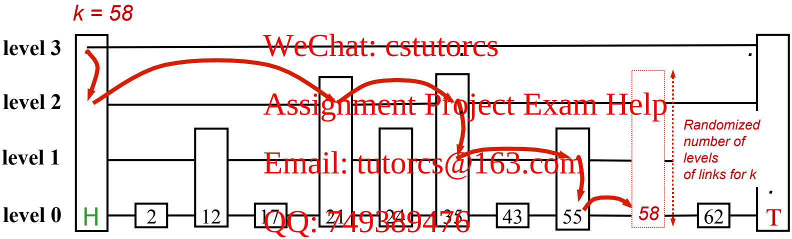
- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

- Problem: insertions and deletions will destroy such a structure...
- We need dynamically self-balancing structure.
- This is where randomisation comes into play, ensuring that in the long run the structure remains (essentially) balanced.



- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

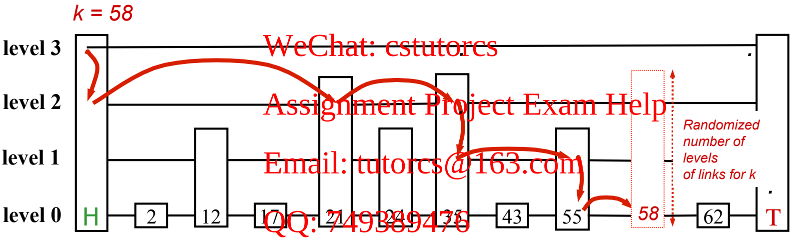
- Problem: insertions and deletions will destroy such a structure...
- We need dynamically self-balancing structure.
- This is where randomization comes into play, ensuring that in the long run the structure remains (essentially) balanced.



- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

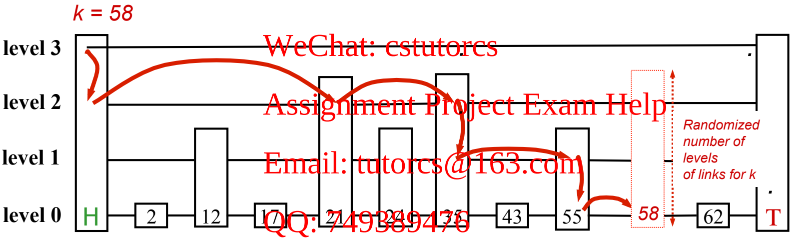
Skip Lists

- Problem: insertions and deletions will destroy such a structure...
- We need dynamically self-balancing structure.
- This is where randomisation comes into play, ensuring that in the long run the structure remains (essentially) balanced.



- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

- Problem: insertions and deletions will destroy such a structure...
- We need dynamically self-balancing structure.
- This is where randomisation comes into play, ensuring that in the long run the structure remains (essentially) balanced.



- **To Insert** k: first search to find the right place. Then toss a coin until you get a head, and count the number of tails t that you got.
- Insert k and link it at levels $0 - t$ from the bottom up.

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we search for an element?
- The probability of getting i consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

Skip Lists

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we search for an element?
- The probability of getting i consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

Skip Lists

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we search for an element?
- The probability of [illegible] consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

Skip Lists

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we search for an element?
- The probability of getting i consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.

- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

Skip Lists

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we search for an element?
- The probability of getting i consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we se[illegible]n element?
- The probability of [illegible] consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total expected number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

- **Deleting** an element is just like in a standard doubly linked list, but taking care of all pointers affected.
- How fast can we se[illegible]n element?
- The probability of [illegible] consecutive tails when flipping a coin i times is $1/2^i$.
- Thus, an element has links on levels $0 - i$ (and possibly also on higher levels) with probability $1/2^i$.
- If n elements belong to a set with a probability p each, then the expected size of that set is $n\,p$.
- Thus, an n element Skip List has on average $n/2^i$ elements with links on level i.
- Since an element has links only on levels $0 - i$ with probability $1/2^{i+1}$, the total **expected** number of link levels per element is

$$\sum_{i=0}^{\infty} \frac{i+1}{2^{i+1}} = \sum_{i=1}^{\infty} \frac{i}{2^i} = 2$$

Skip Lists

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected number of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by the Markov inequality the probability of having at least one element on level i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2 \log n$ is smaller than $n/2^{2 \log n} = n/(2^{\log n})^2 = n/n^2 = 1/n$.
- More generally, the probability to have an element on level $k \log n$ is smaller than $n/2^{k \log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.
- Thus, the probability that level $k \log n$ is nonempty is smaller than $1/n^{k-1}$.
- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k \log n$?

Skip Lists

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected … of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by th… inequality the probability of having at least one elemen… i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2\log n$ is smaller than $n/2^{2\log n} = n/2^{\log n^2} = n/n^2 = 1/n$.
- More generally, the probability to have an element on level $k \log n$ is smaller than $n/2^{k\log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.
- Thus, the probability that level $k\log n$ is nonempty is smaller than $1/n^{k-1}$.
- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k \log n$?

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected number of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by the Markov inequality the probability of having at least one element on level i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2 \log n$ is smaller than $n/2^{2\log n} = n/2^{\log n^2} = n/n^2 = 1/n$.

- More generally, the probability to have an element on level $k \log n$ is smaller than $n/2^{k \log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.

- Thus, the probability that level $k \log n$ is nonempty is smaller than $1/n^{k-1}$.

- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k \log n$?

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected number of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by the Markov inequality the probability of having at least one element on level i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2\log n$ is smaller than $n/2^{2\log n} = n/2^{\log n^2} = n/n^2 = 1/n$.

- More generally, the probability to have an element on level $k\log n$ is smaller than $n/2^{k\log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.

- Thus, the probability that level $k\log n$ is nonempty is smaller than $1/n^{k-1}$.
- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k\log n$?

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected number of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by the Markov inequality the probability of having at least one element on level i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2\log n$ is smaller than $n/2^{2\log n} = n/2^{\log n^2} = n/n^2 = 1/n$.

- More generally, the probability to have an element on level $k \log n$ is smaller than $n/2^{k\log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.

- Thus, the probability that level $k \log n$ is nonempty is smaller than $1/n^{k-1}$.

- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k \log n$?

- Let $\#(i)$ denote the number of elements on level i.
- Since the expected number of elements having a link at level i is $E[\#(i)] = \frac{n}{2^i}$, by the Markov inequality the probability of having at least one element on level i satisfies

$$P(\#(i) \geq 1) \leq \frac{E[\#(i)]}{1} = \frac{n}{2^i}$$

- Thus, the probability to have an element on level $2 \log n$ is smaller than $n/2^{2 \log n} = n/2^{\log n^2} = n/n^2 = 1/n$.

- More generally, the probability to have an element on level $k \log n$ is smaller than $n/2^{k \log n} = n/2^{\log n^k} = n/n^k = 1/n^{k-1}$.

- Thus, the probability that level $k \log n$ is nonempty is smaller than $1/n^{k-1}$.

- What is the expected value E of k such that k is the least integer so that the number of levels is $\leq k \log n$?

Skip Lists

$$E = \sum_{k=1}^{\infty} \frac{k}{n^{k-1}} = \left(\frac{n}{n-1}\right)^2$$

(using the same tricks that ... e such a sum)

- Thus, the expected number of levels is barely larger than $\log n$.
- If an element has a link at a level i then with probability $1/2$ it also has a link at level $i+1$.
- Thus, the expected number of elements between any two consecutive elements with a link at level $i+1$ which have links only up to level i is smaller than

$$\frac{0}{2} + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \ldots = 1$$

So once on level i, on average we will have to inspect only two elements on that level before going to a lower level.

Skip Lists

$$E(\ldots) = \sum_{k=1}^{\infty} \frac{k}{n^{k-1}} = \left(\frac{n}{n-1}\right)^2$$

(using the same tricks [illegible] such a sum)

- Thus, the expected number of levels is barely larger than $\log n$.

- If an element has a link at a level i then with probability $1/2$ it also has a link at level $i+1$.

- Thus, the expected number of elements between any two consecutive elements with a link at level $i+1$ which have links only up to level i is smaller than

$$\frac{0}{2} + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \ldots = 1$$

So once on level i, on average we will have to inspect only two elements on that level before going to a lower level.

$$E = \sum^{\infty} \frac{k}{n^{k-1}} = \left(\frac{n}{n-1}\right)^2$$

(using the same tricks th... e such a sum)

- Thus, the expected number of levels is barely larger than $\log n$.
- If an element has a link at a level i then with probability $1/2$ it also has a link at level $i+1$.

- Thus, the expected number of elements between any two consecutive elements with a link on level $i+1$ which have links only up to level i is smaller than

$$\frac{0}{2} + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \ldots = 1$$

So once on level i, on average we will have to inspect only two elements on that level before going to a lower level.

$$E \ldots \sum^{\infty} \frac{k}{n^{k-1}} = \left(\frac{n}{n-1}\right)^2$$

(using the same tricks t[illegible]e such a sum)

- Thus, the expected number of levels is barely larger than $\log n$.
- If an element has a link at a level i then with probability $1/2$ it also has a link at level $i+1$.
- Thus, the expected number of elements between any two consecutive elements with a link on level $i+1$ which have links only up to level i is smaller than

$$\frac{0}{2} + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \ldots = 1$$

So once on level i, on average we will have to inspect only two elements on that level before going to a lower level.

$$E \ldots \sum^{\infty} \frac{k}{\ldots^{\ldots-1}} = \left(\frac{n}{n-1}\right)^2$$

(using the same tricks t[illegible]e such a sum)

- Thus, the expected number of levels is barely larger than $\log n$.
- If an element has a link at a level i then with probability $1/2$ it also has a link at level $i+1$.
- Thus, the expected number of elements between any two consecutive elements with a link on level $i+1$ which have links only up to level i is smaller than

$$\frac{0}{2} + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \ldots = 1$$

So once on level i, on average we will have to inspect only two elements on that level before going to a lower level.

- To summarise, on average, there will be fewer than $2\log n$ levels to go down, with visiting on average only two elements per each level.
- Consequently, on average, the search will be in time $O(\log n)$.
- For an element with links on levels $0 - i$ we have to store $2(i+1)$ pointers to other elements and the expected number of elements with highest link on level i is $O(n/2^{i+1})$. Thus, total expected space for all pointers is

$$O\left(\sum_{i=0}^{\infty} 2(i+1)\frac{n}{2^{i+1}}\right) = O\left(2n\sum_{i=0}^{\infty}\frac{i+1}{2^{i+1}}\right) = O(4n) = O(n)$$

- Unless we must ensure that the worst case performance of search is $O(\log n)$, Skip Lists are a better option than BST.

- To summarise, on average, there will be fewer than $2\log n$ levels to go down, with visiting on average only two elements per each level.
- Consequently, on average, the search will be in time $O(\log n)$.
- For an element with links on levels $0 - i$ we have to store $2(i+1)$ pointers to other elements and the expected number of elements with highest link on level i is $O(n/2^{i+1})$. Thus, total expected space for all pointers is expected

$$O\left(\sum_{i=0}^{\infty} 2(i+1)\frac{n}{2^{i+1}}\right) = O\left(2n\sum_{i=0}^{\infty}\frac{i+1}{2^{i+1}}\right) = O(4n) = O(n)$$

- Unless we must ensure that the worst case performance of search is $O(\log n)$, Skip Lists are a better option than BST.

- To summarise, on average, there will be fewer than $2\log n$ levels to go down, with visiting on average only two elements per each level.
- Consequently, on average, the search will be in time $O(\log n)$.
- For an element with links on levels $0 - i$ we have to store $2(i+1)$ pointers to other elements and the expected number of elements with highest link on level i is $O(n/2^{i+1})$. Thus, total expected space for all pointers does not exceed

$$O\left(\sum_{i=0}^{\infty} 2(i+1)\frac{n}{2^{i+1}}\right) = O\left(2n\sum_{i=0}^{\infty}\frac{i+1}{2^{i+1}}\right) = O(4n) = O(n)$$

- Unless we must ensure that the worst case performance of search is $O(\log n)$, Skip Lists are a better option than BST.

- To summarise, on average, there will be fewer than $2\log n$ levels to go down, with visiting on average only two elements per each level.
- Consequently, on average, the search will be in time $O(\log n)$.
- For an element with links on levels $0 - i$ we have to store $2(i+1)$ pointers to other elements and the expected number of elements with highest link on level i is $O(n/2^{i+1})$. Thus, total expected space for all pointers does not exceed

$$O\left(\sum_{i=0}^{\infty} 2(i+1)\frac{n}{2^{i+1}}\right) = O\left(2n\sum_{i=0}^{\infty}\frac{i+1}{2^{i+1}}\right) = O(4n) = O(n)$$

- Unless we must ensure that the worst case performance of search is $O(\log n)$, Skip Lists are a better option than BST.

An improvement of Skip Lists

Homework:

- Note that accessing the k^{th} largest element is still $O(n)$.
- Add something to the structure so that accessing the k^{th} largest element is also $O(\log n)$ expected time.

Homework:

- Note that accessing the k^{th} largest element is still $O(n)$.
- Add something to the structure so that accessing the k^{th} largest element is also $O(\log n)$ expected time.