Searching a key $x$ in a sorted linked list

1. cell *p = head;
2. while ($p \rightarrow \text{next} \rightarrow \text{key} < x$) $p = p \rightarrow \text{next}$;
3. return $p$; // (which is either equal or larger than $x$)

Note:
- The $-\infty$ and $\infty$ elements are not "real" keys.
  - They are in the list to prevent checking special cases
- Sometimes we prefer to return the element proceeding the one containing $x$. Then line 2 is replaced with
  while ($p \rightarrow \text{next} \rightarrow \text{key} < x$) $p = p \rightarrow \text{next}$

inserting a key into a Sorted linked list

To insert 35 -
- $p = \text{find}(35)$; // find the proceeding element – the next one is > 35
- CELL *$p1 = (\text{CELL} *) \text{malloc}($\text{sizeof}$(\text{CELL}))$;
- $p1 \rightarrow \text{key}=35$;
- $p1 \rightarrow \text{next} = p \rightarrow \text{next}$;
- $p \rightarrow \text{next} = p1$;

deleting a key from a sorted list

To delete 37 -
- $p = \text{find}(37)$; // Again find proceeding element
- CELL *$p1 = p \rightarrow \text{next}$;
- $p \rightarrow \text{next} = p1 \rightarrow \text{next}$;
- free($p1$);
A data structure for maintaining keys in a sorted order

Rules:
- Consists of several levels.
- All keys appear in level 1.
- Each level is a sorted list.
- If key $x$ appears in level $i$, then it also appears in all levels below level $i$.
- First element in each level has key $-\infty$.
- Last element has key $+\infty$.
- First element in upper level is pointed to by variable $top$.

More rules:
- An element in level $i > 1$ points (via down pointer) to the element with the same key in the level below.
- Elements in the lowest level have down-pointer = NULL.
- Also maintain a counter specifying the number of levels.

Finding an element with key $x$
- $p=top$;
- while(1) {
  - while $(p.next <= x)$ $p=p.next$;
  - if $(p.down == NULL)$ return $p$;
  - $p=p.down$;
- }

If the key $x$ is in SL, we return a pointer to the lowest element contain $x$. If $x$ is not in SL, return pointer to lowest predecessor.
A "perfect" SkipList

A SL is perfect if between every two consecutive keys of level \( i \) there is exactly one key of level \( i-1 \).

Scheme for creation a well-performing SL

- Start from Level 1 (lowers level)
- For \( i=2,3, \ldots \) generation of Level \( i \), we scan the keys in level \( i-1 \). Each second key is "promoted" to participate in level \( i \) as well.

Most SL are not perfect. Hard to maintain.

Search in a "perfect" SkipList

Another example

\[
\begin{align*}
p & = \text{top} \\
\text{while}(1) & \\
& \quad \text{while}(p \rightarrow \text{next} \rightarrow \text{key} \leq x) \\
& \quad \text{if}(p \rightarrow \text{down} == \text{NULL}) \text{return } p \\
& \quad p \rightarrow \text{p} \rightarrow \text{down} \\
\end{align*}
\]

Inserting an element - cont.

- If \( k \) is larger than the current number of levels, add new levels (and update \( \text{top} \) and \( \text{num_of_levels} \) counter)
- Example - insert(119) when \( k=4 \)
- Heuristic: Add at most one new level (not needed for the analysis)
Determining $k$

- $k$ - the number of levels at which an element $x$ participate.
- Use a random function $OurRnd()$ --- returns 1 or 0 (True/False) with equal probability.
  - $k=1$
  - While ($OurRnd()==1$) $k++$

Deleteing a key $x$

- Find $x$ in all the levels it participates, using $find(x)$.
- During the “find”, delete $x$ from each level it participates using the standard “delete from a linked list” method.
- If one or more of the upper levels become empty, remove them (and update $top$ and $num\_of\_levels$)

“expected” space requirement

- **Claim**: The expected number of elements is $O(n)$.

- The term “expected” here refers to the experiments we do while tossing the coin (or calling $OurRnd()$). No assumption about input distribution.

- So imagine a given set, given set of operations insert/del/find, but we repeat many time the experiments of constructing the SL, and count the #elements.

Facts about SL

- **Def**: The **height** of the SL is the number of levels
- **Claim**: The expected number of levels is $O(\log n)$ (here $n$ is the number of keys)
- **Proof** (A rigorous proof coming later)
  - The number of elements participate in the lowest level is $n$.
  - Since the probability of an element to participates in level 2 is $\frac{1}{2}$, the expected number of elements in level 2 is $n/2$.
  - Since the probability of an element to participates in level 3 is $\frac{1}{4}$, the expected number of elements in level 3 is $n/4$.
  - …
  - The probability of an element to participate in level $j$ is $(1/2)^{j-1}$ so number of elements in this level is $n/2^{j-1}$
  - So after $\log(n)$ levels, no element is left.
Facts about SL

- **Claim**: The expected number of elements is $O(n)$.
- (here $n$ is the number of keys)
- **Proof** (Real proof – later)
  - The total number of elements is
    $$n + n/2 + n/4 + n/8 \ldots$$
  - $\leq n(1 + 1/2 + 1/4 + 1/8 + \ldots) = 2n$

To reduce the worst case scenario, we verify during insertion that
$k$ (the number of levels that an element participates in) is $\leq \log n$

“Conclusion”: The expected storage is $O(n)$

More facts

- **Thm**: The expected time for find/insert/delete is $O(\log n)$
- **Proof** For all Insert and Delete, the time is $\leq$ expected #elements scanned during find($x$) operation.
- Will show: Need to scan expected $O(\log n)$ elements.

Thm: Expected time for `find` operation is $O(\log n)$

- **Proof** – we know that there are $O(\log n)$ levels. Will show that we spend $O(1)$ time in each level.
- Assume during find($x$), we scanned $t$ elements, (for $t > 8$) in level $r$. Assume first that $r$ is not the upper level.
- (the search visited $b_1$ branched down to $b_2$ and then visited $b_2, b_3$)
- Assume first that $r$ is not the upper level.
- (not sure what happened before or after)

Bounding time for insert/delete/find

- Putting it together: The expected number of elements scanned in each level is $O(1)$
- There are $O(\log n)$ levels
- Total time is $O(\log n)$
- As stated, getting bounds for time for insert/delete are similar

The probability that none of these 7 elements reached level $r+1$ (why?)

None of these 7 elements reached level $r+1$ (why?)

The probability that none of these 7 elements reached level $r+1$ is $1/2^7$. For larger value of $7$ – very slim.
How likely is that the SL is too tall?

- Let's ask how likely it is that the \#levels is $Z \log_2 n$, where $Z=1,2,3,\ldots$
- That is, we estimate the probability that the height of the SL is
  - $\log_2 n$
  - $2 \log_2 n$
  - $3 \log_2 n$
  - $4 \log_2 n$
  - $\ldots$

Reminder from probability

- Assume that $A, B$ are two events. Let
  - $\Pr(A)$ be the probability that $A$ happens,
  - $\Pr(B)$ be the probability that $B$ happens
  - $\Pr(A \cup B)$ is the probability that either event $A$ happens or event $B$ happens (or both).
- So probably that at least one of them happened is $\Pr(A) + \Pr(B) - \Pr(A \cap B) \leq \Pr(A) + \Pr(B)$
- Similarly, for 3 Events $A_1, A_2, A_3$. The probability that at least one of them happened $\Pr(A_1 \cup A_2 \cup A_3) \leq \Pr(A_1) + \Pr(A_2) + \Pr(A_3)$

Example: In a roulette, the result is a number $k$ between 1..38
- Event $A$: $k$ is even. $\Pr(A) = \Pr(k \text{ is even}) = 19/38 = 0.5$
- Event $B$: $k$ is divided by 3. $\Pr(B) = 12/38 = 0.315$
- $\Pr(A \text{ or } B) = \Pr(A \cap B) = \Pr((k \text{ is divided by 2}) \text{ or } (k \text{ is divided by 3})) \leq 0.5 + 0.315 = 0.815$

But how likely is that the SL is too tall?

- Assume the keys in the SL are \{$x_1, x_2, \ldots, x_n$\}
- The probability that $x_j$ participates in $\geq k+1$ levels is $2^{-k}$.
- (same probability for all $x_j$).
- Define: $A_j$ is the event that $x_j$ participates in $\geq k+1$ levels.
- $\Pr(A_j) = 2^{-k}$
- Define: $A$ is the event that $x_j$ participates in $\geq k+1$ levels.
- $\Pr(A) = 2^{-k}$ (for every $j$)
- If the height of SL $\geq k+1$ then at least one of the $x_j$'s participates in $\geq k+1$ levels.
- The probability that any $x$ (one or more) participates in $\geq k+1$ levels is $\leq \Pr(A_1) + \Pr(A_2) + \ldots + \Pr(A_n) = n \cdot 2^{-k}$
- This is the probability that the height of the SL is $\geq k+1$.

But how likely is that the SL is tall?

- The probability that any $x$ participates in at least $k$ levels is $\leq n \cdot 2^{-k}$. Then the height of the SL $\geq k+1$.
- Ignore the ‘$+1$’
- If none of the $x_j$’s is at level $\geq k$ then the height is $\leq k$.
- Recall $y^{ab} = (y^a)^b = (y^b)^a$
  - $2^{\log_2 n} = n$ and $2^{5(\log_2 n)} = (n)^5$
- Write $k = -1 + Z \log_2 n$.
- Want to find: The probability that the height is $Z$ times $\log_2 n$.
- That is, Twice $\log_2 n$, 3 time $\log_2 n$, 4 times $\log_2 n$...
But how likely is that the SL is tall?

- The probability that any $x$ participates in at least $k$ levels is $\leq n2^k$. Then the height of the SL $\geq k+1$.
- Want to find: The probability that the height is $Z$ times $\log_2 n$.
- Twice $\log_2 n$, 3 times $\log_2 n$, 4 times $\log_2 n$ ...
- Then $2^k \leq (2 \log_2 n)^2 = n^2 \implies n_1 = n^2$
- So $n_2^k \leq n / n_1 = 1/n$
- This is the probability that the height of SL $\geq Z \log_2 n$
- Example: $n=1000$.

In other words (and with hand-waving)

- Assume we have a set of $n>1000$ keys, and we keep rebuilding Skiplists for them.
- Call a SL bad if its height $> 7 \log_2 n$
- First build SL$_1$.
- Then build SL$_2$ (for the same keys)
- Then ...(and so on)
- Then SL$_M$ where $M = 10^{20}$
- Then less than 200 of them are bad (with high probability)

Random Variable (light version)

- Assume we perform an experiment (Flipping a coin). Let $R$ be the result – Face or Tail (F/T).
- We could define a random variable which (in this course) is a value that depends on the result of the experiment.
- Preferably, set to `1` if some condition is satisfied, and is `0` otherwise.
- Define $X$ to be a random variable, set to `1` if $R$ is Face; `0` if $R$ is Tail.
- Define $Y$ to be another random variable, set to `1` if $R$ is Face, `1` if $R$ is Tail.
- We could ask what is the probability that $X=1$. Denote $Pr(X=1)$
- If coin is fair, $Pr(X=1)$ is $0.5$, $Pr(X=2)$=$Pr(X=3)$=$Pr(X=17)$=$0$
- $Pr(Y=1)$=$Pr(Y=2)$=$0$ ; $Pr(Y=5)$=$0.5$ ; $Pr(Y=-3)$=$0.5$

Random Variable (light version)

- Assume we perform an experiment (tossing a dice). Let $R$ be the result – one of the number 1,2,3,4,5,6.
- We could define a random variable which (in this course) is a value that depends on the result of the experiment.
- Preferably, set to `1` if some condition is satisfied, and is `0` otherwise.
- Define $F$ to be a random variable, set to `1` if $R$ is even; `0` if $R$ is odd.
- Define $Q$ to be another random variable, which is `1` if $R$ is even; `0` else.
- We could ask what is the probability that $F=1$. Denote $Pr(F=1)$
- If dice is fair, $Pr(F=1)$ is $0.5$, and $Pr(Q=1)=4/6=0.666$
Random Variable and expectation (light version)

In many cases, we would like to know what is the **expected** value of a random var.

Example: If \( Y = 1 \) we earn a dollar. What is the expected amount we earn in one game. We denote it by \( \mathbb{E}(Y) \).

Good news. If \( Y \) is a Boolean var then \( \mathbb{E}(Y) \), the expected value of \( Y \), is just \( \Pr(Y=1) \).

What if we earn $17 if \( Y=1 \).

**Lemma**: for any constant \( \alpha \) it is always true that \( \mathbb{E}(\alpha Y) = \alpha \mathbb{E}(Y) \).

Random Variable and expectation (light version)

If a random variable could accept several values?

Say define a new random var \( Z=X+Q \).

So \( Z=0 \) if \( X=0 \) and \( Q=0 \);
\( Z=2 \) if \( X=1 \) and \( Y=1 \);
\( \ldots \)

Let’s assume that we earn $1 if \( X=1 \), and $1 if \( Q=1 \). The expected earning is

\[
\mathbb{E}(Z) = \sum_{j=0}^{\infty} j \cdot \Pr(Z=j)
\]

But in our setting \( \mathbb{E}(Z) = \Pr(Z=0) + 1 \cdot \Pr(Z=1) + 2 \Pr(Z=2) \)

**Lemma**: \( \mathbb{E}(X+Y) = \mathbb{E}(X) + \mathbb{E}(Y) \)

So in our case \( \mathbb{E}(Z) = 0.5 + 0.666 \)

**Lemma** for any constants: \( \alpha \) \( \beta \)

\[
\mathbb{E}(\alpha X + \beta Y) = \alpha \mathbb{E}(X) + \beta \mathbb{E}(Y)
\]

\[
\mathbb{E}(X+Y+Z) = \mathbb{E}(X) + \mathbb{E}(Y) + \mathbb{E}(Z)
\]

Expected number of keys in the SL

- Assume we are inserting \( n \) keys, \( k_1 \ldots k_n \), into an empty SkipList.
- Define a set of Random variables \( X_{ih} \) (\( i = 1 \ldots n \), \( h = 1 \ldots \infty \))
  - \( X_{ih} = 1 \) if the key \( k_i \) participates in level \( h \), and \( X_{ih} = 0 \) if not.
  - \( \Pr(X_{ih}=1) = 1/2^{h-1} \) So \( \mathbb{E}(X_{ih}) = 1/2^{h-1} \)
  - The number of keys in the SL is just

\[
\sum_{i=1}^{n} \sum_{h=1}^{\infty} X_{ih}
\]

Expected number of keys in the SL (cont)

\[
\mathbb{E} \left( \sum_{i=1}^{n} \sum_{h=1}^{\infty} X_{ih} \right) = \sum_{i=1}^{n} \sum_{h=1}^{\infty} \mathbb{E}(X_{ih}) = \sum_{h=1}^{\infty} \mathbb{E}(X_{ih}) = n \left( 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots \right) = 2n
\]