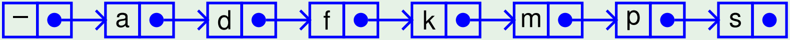
Contents

Section outline

1. **Skip lists**
   - Perfect skip lists
   - Searching in a SL
   - Improving over the perfect SL
   - Distribution of nodes in a (randomised) SL
   - Expected space usage
   - Maximum level in a SL
   - Idealised skipping for search
   - Skipping in the SL during search

Improving on the linked list

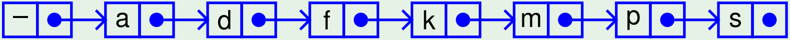
Example (The slow and simple linked list)



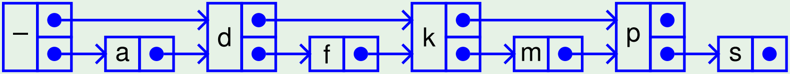
Steps to reach “s”: 7

Improving on the linked list

Example (The slow and simple linked list)



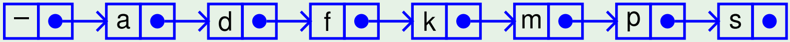
Steps to reach "s": 7



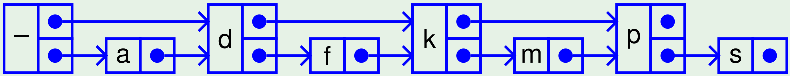
Steps to reach "s": 4

Improving on the linked list

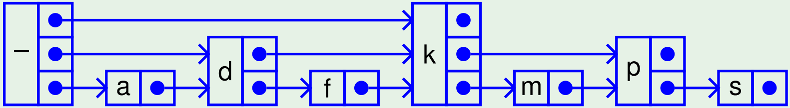
Example (The slow and simple linked list)



Steps to reach "s": 7



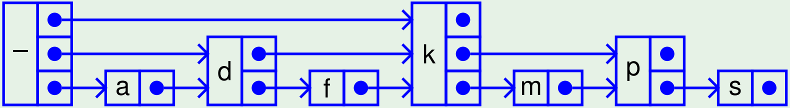
Steps to reach "s": 4



Steps to reach "s": 3

Perfect skip lists

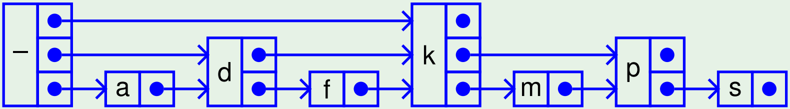
Example (Perfect SL)



- At level 0 all elements are connected as a simple linked list
- All alternate elements at level i are connected as a linked list at level $(i+1)$ – each higher level contains half the elements of the level below it
- Maximum number of levels is $\lceil \lg n \rceil$

Perfect skip lists

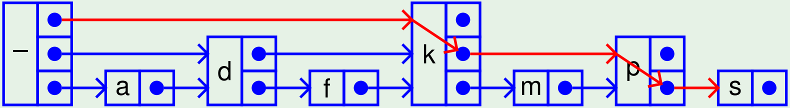
Example (Perfect SL)



- At level 0 all elements are connected as a simple linked list
- All alternate elements at level i are connected as a linked list at level $(i + 1)$ – each higher level contains half the elements of the level below it
- Maximum number of levels is $\lceil \lg n \rceil$
- Hard to maintain on insertion and deletion, entails $O(n)$ time for insertion and deletion
- Relax the criterion for higher level links from deterministic to probabilistic – proposed by William Pugh around 1989

Searching in a SL

Example (Searching for ‘s’)

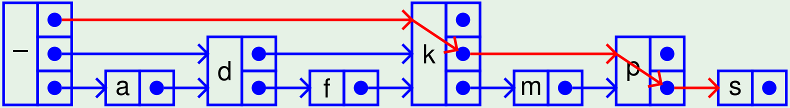


Searching for a key ky, starting at the highest level

1. If list is empty, then fail if at the lowest level
   otherwise continue searching after descending one level

Searching in a SL

Example (Searching for ‘s’)

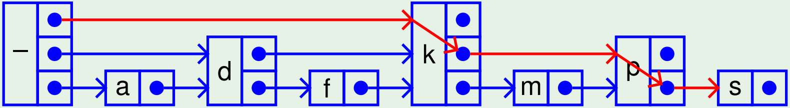


Searching for a key ky, starting at the highest level

1. If list is empty, then fail if at the lowest level
   otherwise continue searching after descending one level
2. Stored key at current level matches ky?
   If yes, then done

Searching in a SL

Example (Searching for 's')

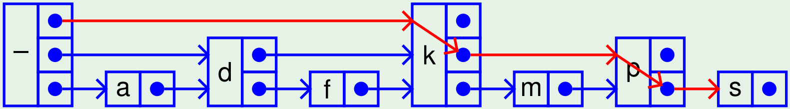


Searching for a key ky, starting at the highest level

1. If list is empty, then fail if at the lowest level
   otherwise continue searching after descending one level
2. Stored key at current level matches ky?
   If yes, then done
3. Stored key at current level $<$ ky ?
   If yes, redo search from one level less, if not at the lowest level
   otherwise fail

Searching in a SL

Example (Searching for ‘s’)

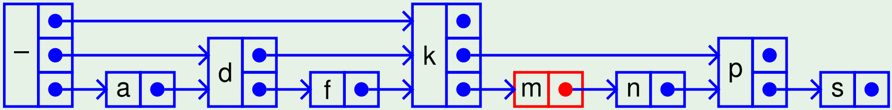


Searching for a key ky, starting at the highest level

1. If list is empty, then fail if at the lowest level
   otherwise continue searching after descending one level
2. Stored key at current level matches ky?
   If yes, then done
3. Stored key at current level $<$ ky ?
   If yes, redo search from one level less, if not at the lowest level
   otherwise fail
4. Now, Stored key at current level $>$ ky
   Continue search from this point

Improving over the perfect SL

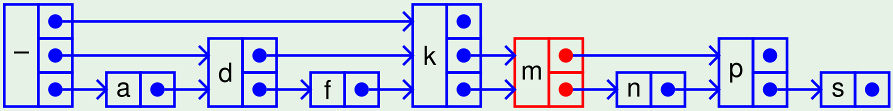
Example



- After inserting an element, keep adding levels with a probability of p, which is taken as $\frac{1}{2}$

Improving over the perfect SL

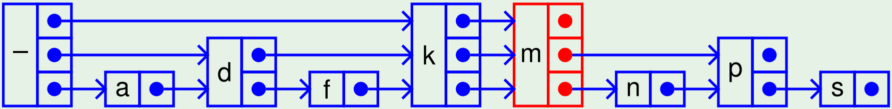
Example



- After inserting an element, keep adding levels with a probability of p, which is taken as $\frac{1}{2}$

Improving over the perfect SL

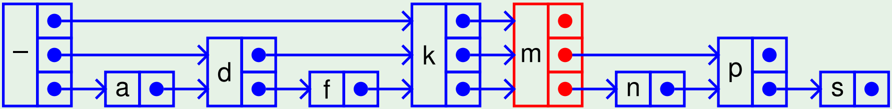
Example



- After inserting an element, keep adding levels with a probability of p, which is taken as $\frac{1}{2}$
- $\Pr[\ell = k] = (1-p)p^k$; $\Pr[\ell \geq k] = p^k$; $\ell \geq 0$
- Note that the number of levels at a key is independent of those of other nodes

Improving over the perfect SL

Example



- After inserting an element, keep adding levels with a probability of p, which is taken as $\frac{1}{2}$
- $\Pr[\ell = k] = (1-p)p^k$; $\Pr[\ell \geq k] = p^k$; $\ell \geq 0$
- Note that the number of levels at a key is independent of those of other nodes
- Searching is done as explained
- While deleting, a key is simply deleted with all its levels – does not affect the distribution of levels of other nodes!

Distribution of nodes in a (randomised) SL

- Let f_k be the fraction of nodes of an aribtrarily long SL *precisely* at level k (having *precisely* $(k+1)$ forward links)
- Let g_k be the fraction of nodes at level k (or higher) (having $(k+1)$ or more forward links)
- $f_k = (1-p)g_k$, as a node does not get promoted with probability $1-p$
- $g_0 = 1$, $f_0 = (1-p)g_0 = 1-p$
- $g_1 = pg_0$, $f_1 = (1-p)g_1 = p(1-p)g_0 = pf_0$
- $g_k = pg_{k-1}$, $f_k = (1-p)g_k = p(1-p)g_{k-1} = pf_{k-1}$
- $f_0 = 1-p = \frac{1}{2}$ for $p = \frac{1}{2}$
- So, 50% of the nodes are at level 0, if nodes are promoted to the next higher level with a probability of 50%

Expected space usage

- The probability that a node extends to a level k is $\frac{1}{2^k}$, $k \geq 0$
- Let the SL have n nodes
- Expected number of nodes at level k is $\frac{n}{2^k}$
- Let the maximum height of the SL be limited to h
- Expected number of nodes used by the SL is $\sum_{k=0}^{k=h} \frac{n}{2^k} < 2n$
- Thus the space requirement of a SL is $O(n)$

Maximum level in a SL

- Let M be the maximum level of any element in the skip list
- If an element tends to go beyond the maximum level, it will be restricted to M
- We do not want this to happen frequently, so it is important to determine M carefully

Maximum level in a SL

- Let M be the maximum level of any element in the skip list
- If an element tends to go beyond the maximum level, it will be restricted to M
- We do not want this to happen frequently, so it is important to determine M carefully
- Pr that a particular element reaches precisely level k is $(1-p)p^k$
- What is the expected number of elements present in levels beyond M?

Maximum level in a SL

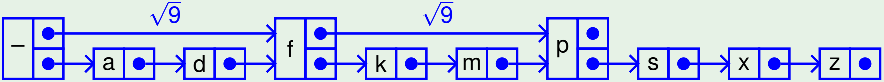
- Let M be the maximum level of any element in the skip list
- If an element tends to go beyond the maximum level, it will be restricted to M
- We do not want this to happen frequently, so it is important to determine M carefully
- Pr that a particular element reaches precisely level k is $(1-p)p^k$
- What is the expected number of elements present in levels beyond M?
- $\mathrm{E}[k > M] = \sum_{k=M+1}^{\infty} n(1-p)p^k = n(1-p)\sum_{k=M+1}^{\infty} p^k = \frac{n(1-p)p^{M+1}}{1-p} = np^{M+1}$
- Let $M = \log_{\frac{1}{p}} n = \log_r n$, where $p = \frac{1}{r}$
- Now, $\mathrm{E}[k > M] = \frac{n}{r^{M+1}} = \frac{n}{r^{\log_r n + \log_r r}} \frac{n}{r^{\log_r nr}} = \frac{1}{r} = p$

Maximum level in a SL

- Let M be the maximum level of any element in the skip list
- If an element tends to go beyond the maximum level, it will be restricted to M
- We do not want this to happen frequently, so it is important to determine M carefully
- Pr that a particular element reaches precisely level k is $(1-p)p^k$
- What is the expected number of elements present in levels beyond M?
- $\mathrm{E}[k > M] = \sum_{k=M+1}^{\infty} n(1-p)p^k = n(1-p)\sum_{k=M+1}^{\infty} p^k = \frac{n(1-p)p^{M+1}}{1-p} = np^{M+1}$
- Let $M = \log_{\frac{1}{p}} n = \log_r n$, where $p = \frac{1}{r}$
- Now, $\mathrm{E}[k > M] = \frac{n}{r^{M+1}} = \frac{n}{r^{\log_r n + \log_r r}} \frac{n}{r^{\log_r nr}} = \frac{1}{r} = p$
- So, $M = \log_{\frac{1}{p}} n$ is a good choice for the (dynamic) maximum level of the SL

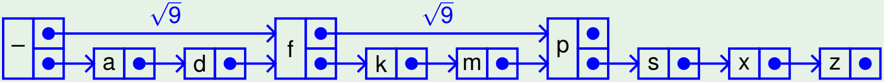
Idealised skipping for search

Example (Baby step, giant step)



Idealised skipping for search

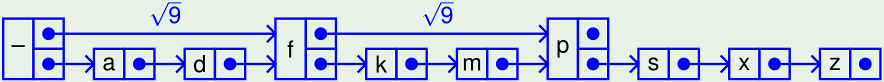
Example (Baby step, giant step)



- For a two level skip list structure, it is optimal to have a skip distance of $\sqrt{n}$
  Resulting search time is $2\sqrt{n}$

Idealised skipping for search

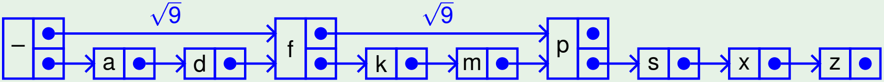
Example (Baby step, giant step)



- For a two level skip list structure, it is optimal to have a skip distance of $\sqrt{n}$
  Resulting search time is $2\sqrt{n}$
- For a three level skip list structure, optimal skip distance is $\sqrt[3]{n}$
  Search time $3\sqrt[3]{n}$

Idealised skipping for search

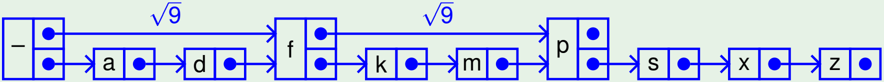
Example (Baby step, giant step)



- For a two level skip list structure, it is optimal to have a skip distance of $\sqrt{n}$
  Resulting search time is $2\sqrt{n}$
- For a three level skip list structure, optimal skip distance is $\sqrt[3]{n}$
  Search time $3\sqrt[3]{n}$
- For a k-level skip list structure, optimal skip distance is $\sqrt[k]{n}$
  Search time is $k\sqrt[k]{n}$

Idealised skipping for search

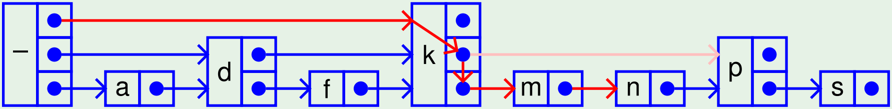
Example (Baby step, giant step)



- For a two level skip list structure, it is optimal to have a skip distance of $\sqrt{n}$
  Resulting search time is $2\sqrt{n}$
- For a three level skip list structure, optimal skip distance is $\sqrt[3]{n}$
  Search time $3\sqrt[3]{n}$
- For a k-level skip list structure, optimal skip distance is $\sqrt[k]{n}$
  Search time is $k\sqrt[k]{n}$
- If the number of levels is chosen as $\lg n$, the optimal skip distance becomes $n^{\frac{1}{\lg n}} = 2$
  Search time is $(\lg n)\left(n^{\frac{1}{\lg n}}\right) = 2\lg n$

Skipping in the SL during search

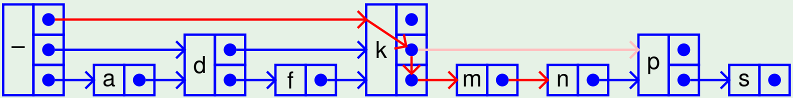
Example (Searching for ‘n’ in a SL)



- Necessary to estimate the number of steps to reach an entry
- Will try to workout in the backward direction
- Note that a level is introduced at a node with probablity of $p = \frac{1}{2}$
- Let $S(\jmath)$ denote the number of steps needed to walk back through $\jmath$ levels

Solving for $S(\jmath)$ – steps with $\jmath$-levels remaining

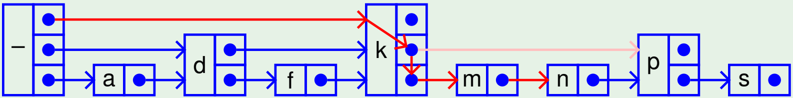
Example (Searching for 'n' in a SL)



- $S(\jmath) = 1 + pS(\jmath - 1) + (1 - p)S(\jmath)$
- $S(0) = 0$ [already at the top level]
  - 1 for traversing through the current node
  - $pS(\jmath - 1)$ for walking back through $(\jmath - 1)$ levels having ascended a level with probability p
  - $(p - 1)S(\jmath)$ for continuing to walking back through the current level with probability $(1 - p)$

Solving for $S(\jmath)$ – steps with $\jmath$-levels remaining

Example (Searching for ‘n’ in a SL)



- $S(\jmath) = 1 + pS(\jmath - 1) + (1 - p)S(\jmath)$
- $S(0) = 0$ [already at the top level]
  - 1 for traversing through the current node
  - $pS(\jmath - 1)$ for walking back through $(\jmath - 1)$ levels having ascended a level with probability p
  - $(p - 1)S(\jmath)$ for continuing to walking back through the current level with probability $(1 - p)$
- $S(\jmath) = \frac{1}{p} + S(\jmath - 1) = \frac{\jmath}{p}$
- For $p = \frac{1}{2}$, $S(\jmath) = 2\jmath$
- If $M = \log_{\frac{1}{p}} n$ is the maximum level, then $S(M) = 2 \lg n$, for $p = \frac{1}{2}$
- Search time is $O(\lg n)$