Föreläsning 10
Datastrukturer (DAT037)

Fredrik Lindblad[1]

29 november 2017

[1]Slides skapade av Nils Anders Danielsson har använts som utgångspunkt. Se
`http://www.cse.chalmers.se/edu/year/2015/course/DAT037`

Innehåll

- Splay-träd
- Prefixträd
- Skip-listor

Splay-träd

Splay-träd

- Sökträd, inget invariant för balansering. "självbalanserande"
- Ingen garanti att $h = O(\log n)$.
- Idé: För upp den åtkomna noden till roten på ett visst sätt.
- Detta ger $O(\log n)$ amorterat. Att visa detta är komplicerat.
- Dessutom går det snabbt att hitta nyligen åtkomna element igen.
- Enkel omstrukturering – kan vara bättre i praktiken än balanserade träd.

Splay-träd

- Vid sökning(!), insättning, urtag splayas djupast åtkomna noden upp till roten.
- Tre fall: *zig*, *zig-zig*, *zig-zag*. Utförs tills noden är rot.
- Zig: enkelrotation, görs om föräldern är roten.
- Zig-zig: två enkelrotationer åt samma håll, görs om noden och föräldern är barn på samma sida.
- Zig-zag: två enkelrotationer åt olika håll (på samma sätt som vänster-högerfallet för AVL-träd), görs om noden och föräldern är barn på olika sidor.
- Animerat: `https://www.cs.usfca.edu/~galles/visualization/SplayTree.html`

Prefixträd

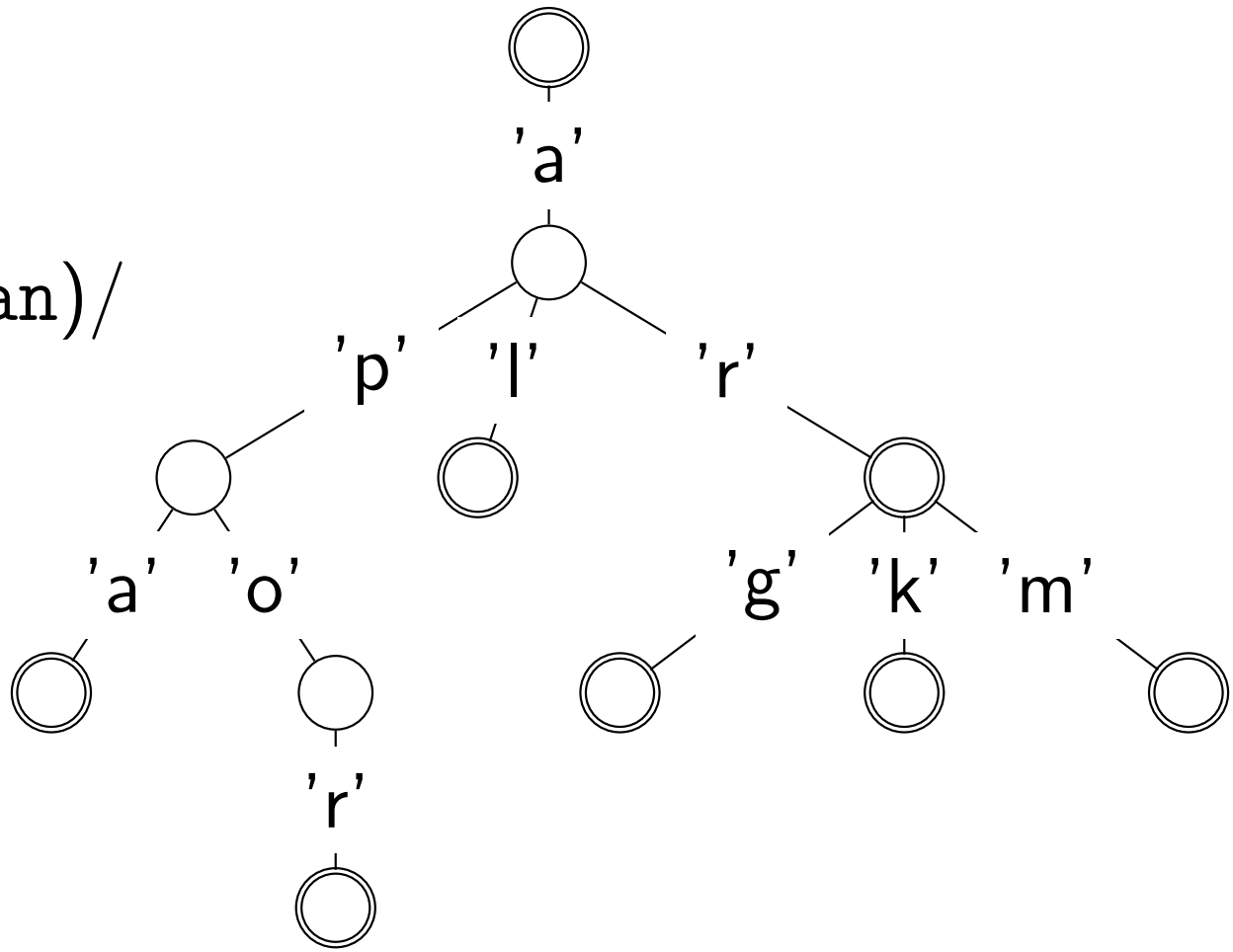
Prefixträd (tries)

- Kan implementera mängder och avbildningar.
- Nycklar: strängar.
- Varje nod kan ha ett barn för varje tecken i alfabetet.
- n:te nivån motsvarar n:te tecknet.

Prefixträd (tries)

- Träd med tecken på kanterna.
- Medlem(`boolean`)/ Värde(`V`)
- `Map` från alfabet till barn.
- Löv är `true`/har värde.

{"", "apa", "apor", "al", "ar", "arg", "ark", "arm"}:



Prefixträd (tries)

Några alternativ för `Map` i noderna:

	Array	Enkellänkad lista	AVL-träd
Storlek	$\Theta(\lvert\Sigma\rvert)$	$\Theta(c)$	$\Theta(c)$
Sökning	$\Theta(1)$	$O(c)$	$O(\log c)$
Insättning	$O(\lvert\Sigma\rvert)$	$O(c)$	$O(\log c)$

Måtten gäller per nod.

- Σ: Alfabetet $(0, 1, ..., \lvert\Sigma\rvert - 1)$.
- c: Antalet barn $(c \leq \lvert\Sigma\rvert)$.

Vad är värstafallstidskomplexiteten för att testa medlemskap av en sträng i en mängd som innehåller n strängar? Varje sträng har längd ℓ, prefixträdet använder arrayer, och hashfunktionen har tidskomplexiteten $\Theta(\ell)$.

- Hashtabell: $O(\ell)$ eller $O(\ell n)$.
- Prefixträd: $O(\ell)$ eller $O(\ell n)$.
- AVL-träd: $O(\log n)$ eller $O(\ell \log n)$.

Vad är värstafallstidskomplexiteten för att testa medlemskap av en sträng i en mängd som innehåller n strängar? Varje sträng har längd ℓ, prefixträdet använder arrayer, och hashfunktionen har tidskomplexiteten $\Theta(\ell)$.

- Hashtabell: $O(\ell)$ eller $O(\ell n)$.
- Prefixträd: $O(\ell)$ eller $O(\ell n)$.
- AVL-träd: $O(\log n)$ eller $O(\ell \log n)$.

Svar: Hashtabell: En hink: $O(\ell n)$. $O(\ell)$ med perfekt hashfunktion. Prefixträd: $O(\ell)$. AVL-träd: $O(\ell \log n)$.

Skipplistor

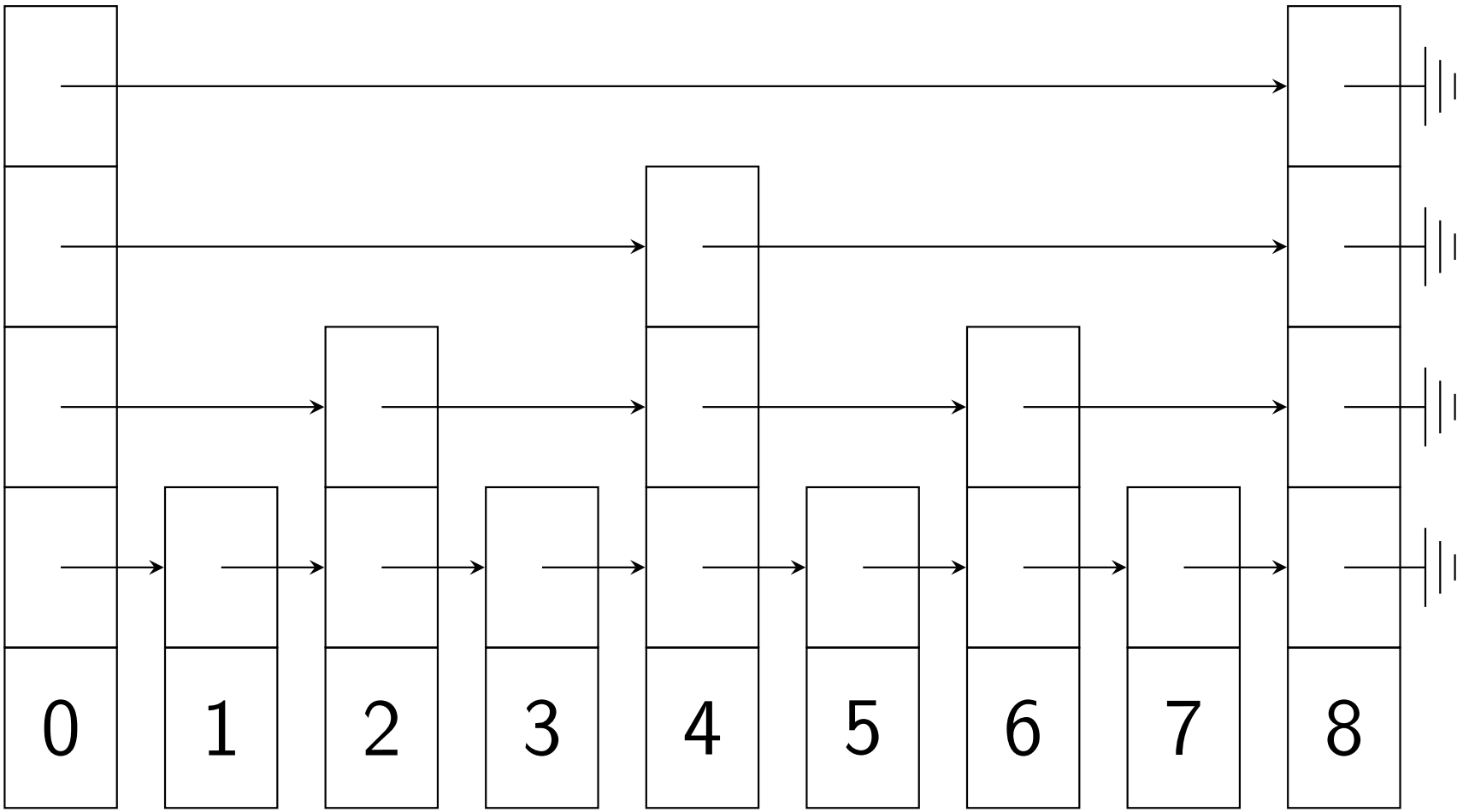
Skipplistor

- Alternativ till balanserade sökträd.
- Randomiserad datastruktur.

Skipplistor

- Sorterad länkad lista: långsam sökning.
- Kanske kan lägga till fler länkar.
- Om nod $i2^k$ har länk till nod $(i+1)2^k$:
  snabb sökning, långsam insättning.
  - Höjd (maximalt antal länkar från en nod): $\Theta(\log n)$.
  - `member`/`find-max`: $O(\log n)$.
  - `insert`/`delete-min`: $O(n)$. Inte bra.
    Måste i värsta fall ändra på höjden på alla noder.

Skipplistor



Skipplistor

Skipplistor:

- Samma idé, men slumpmässig struktur (samt vaktpost av maximal höjd i början).
- Insättning: Slumpmässig höjd på noden.
- Stanna på höjd 1 med sannolikhet $1 - p$ ($0 < p < 1$).
- Stanna på höjd 2 med sannolikhet $1 - p$.
- ...

Skipplistor

- Sannolikhet för höjd h: $P(h) = (1-p)p^{h-1}$.
- Förväntad höjd:

$$\sum_{h=1}^{\infty} hP(h) = \frac{1}{1-p}.$$

- Rekommenderade val av p: $1/4$, $1/2$.

I en skipplista med n noder, vad är det förväntade antalet noder med höjd $\geq h$ (exklusive vaktposten)?

- pn^{h-1}.
- $n(1-p)p^{h-1}$.
- np^{h-1}.
- n.
- $n(1-p)^{h-1}$.

I en skipplista med n noder, vad är det förväntade antalet noder med höjd $\geq h$ (exklusive vaktposten)?

- pn^{h-1}.
- $n(1-p)p^{h-1}$.
- np^{h-1}.
- n.
- $n(1-p)^{h-1}$.

Svar: np^{h-1}

Skipplistor

- *Förväntad* tidskomplexitet för `insert`, `member`, `delete`, för konstant p: $O(\log n)$.
- Gå igenom elementen i sorterad ordning: $\Theta(n)$.
- I praktiken: Maxhöjd $\log_{1/p} N$, där N är listans största tänkbara storlek.

Iterator för sökträd

Iterator för sökträd

Här kommer ett exempel till på en iterator, denna gång för inorder-genomlöpning av ett binärt träd.

Ex.: Iter. för BST som genomlöper sorterat

```
public class BinarySearchTree
    <A extends Comparable<? super A>> {

  private class Node {
      A    contents;
      Node left, right;
  }

  private Node root;

  public Iterator<A> orderedIterator() {
   return new OrderedIterator();
  }
```

```
...
private class OrderedIterator
   implements Iterator<A> {
 Stack<Node> stack;

 OrderedIterator() {
  pushMinStack(root);
 }
 private void pushMinStack(Node n) {
  while (n != null) {
   stack.push(n);
   n = n.left;
  }
 }
 ...
```

```
    ...
    public boolean hasNext() {
     return !stack.empty();
    }
    public A next() {
     if (stack.empty())
      throw new NoSuchElementException();
     Node node = stack.pop();
     pushMinStack(node.right);
     return node.contents;
    }
   }
}
```