Informatik II
Vorberechnung

G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

Definition

- Preconditioning (Vorbestimmung):
  Falls nacheinander mehrere Probleme zu lösen sind, die einen gemeinsamen Anteil haben, d.h., falls wir nacheinander Eingaben der Form

  $$< x_1; y > < x_2; y > < x_3; y > \ \ldots$$

  bekommen, so kann es sinnvoll sein, zunächst den gemeinsamen Anteil zu transformieren, also $\hat{y} := \mathrm{Tr}(y)$, und dann die Probleme

  $$< x_1; \hat{y} > < x_2; \hat{y} > < x_3; \hat{y} > \ \ldots$$

  zu lösen

- Beispiele
  - Auswertung eines Polynoms an viele Stellen, Eingabe ist

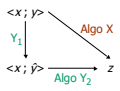
    $$< x_i; \underbrace{a_0, \ldots, a_n}_{\text{"konstant"}} >$$

  - Vorgänger-/Nachfolger-Beziehung in einem Baum testen, Eingabe ist

    $$< v_1, v_2; \underbrace{\text{Baum}}_{\text{„konstant“}} >$$

  - festen Text (z. B. Wörterbuch) nach einem Wort durchsuchen
  - Sortieren, Tries, …
  - Optimale Suchbäume

- Precomputation (Vorberechnung):
  auch wenn die Eingabe nur einmal vorkommt – aber dennoch aus 2 Teilen $<x; y>$ besteht – so kann es effizienter sein, zunächst einen Teil zu transformieren, $\hat{y} := \mathrm{Tr}(y)$, und dann die Lösung aus $<x; \hat{y}>$ zu berechnen



  - Beispiel: dynamischen Text nach einem Wort durchsuchen
- Achtung: im Umgangssprachgebrauch wird *"Precomputation"* oft für beide Arten verwendet
- wesentliche Frage bei Precomputation / Preconditioning ist immer: Wie hoch ist der Aufwand? – bzw.: macht er evtl. den anschließenden Zeitgewinn kaputt?

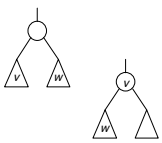
Beispiele bisher

- Closest Pairs
  - Vorsortieren der Punkte entlang x-Achse
- Huffman-Kodierung:
  - Vorsortieren der Zeichen gemäß Häufigkeit

Vorgänger im Baum

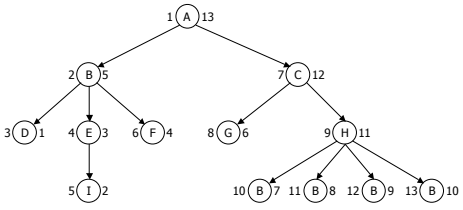
- Problem:
  - gegeben ein Baum
  - wir sollen immer wieder für ein Paar von Knoten (v, w) entscheiden, ob v Vorgängerin, d.h. direkte/indirekte Mutter von w ist
- Naïve Lösung:
  - traversiere Baum von w aus bis zur Wurzel und vergleiche mit v
  - Aufwand:
    - $\Omega(n)$ im Worst-Case,
    - untere Schranke für worst-case = $O(\log n)$
- Behauptung: wir können auf dem Baum Preconditioning durchführen in Zeit $\Theta(n)$, so daß danach das Problem in Zeit $O(1)$ gelöst werden kann

- Methode:
  1. Durchlaufe Baum in Preorder und nummeriere dabei die Knoten; bezeichne diese Nummer mit $N_{\text{pre}}(v)$
     - nummeriere also erst den Knoten selbst mit fortlaufender Nummer, dann nummeriere rekursiv linken, dann rechten Teilbaum
  2. Nummeriere analog die Knoten in Postorder; Bezeichnung $N_{\text{post}}(v)$
  3. Jetzt gilt für jedes Paar v, w von Knoten:
     1. $N_{\text{pre}}(v) \le N_{\text{pre}}(w) \Leftrightarrow$ v Vorgänger von w v v links von w
     2. $N_{\text{post}}(v) \ge N_{\text{post}}(w) \Leftrightarrow$ v Vorgänger von w v v rechts von w



  4. Zusammen:
     $N_{\text{pre}}(v) \le N_{\text{pre}}(w) \wedge N_{\text{post}}(v) \ge N_{\text{post}}(w) \Leftrightarrow$ v Vorgänger von w

Beispiel



Wiederholte Auswertung eines Polynoms

- Gegeben: Polynom $p(x) = a_n x^n + \cdots + a_1 x + a_0$
- Gesucht: $p(x_i)$ für viele $x_1, \ldots, x_n$
- Verboten ist:

```
p = a[0] + a[1]*x + a[2]*x*x + ...
```

- noch schlimmer:

```
p =   a[0] + a[1]*x + a[2]*pow(x,2.0)
    + a[3]*pow(x,3.0) ...
```

- Einfache Methode: transformiere p in Horner-Schema

$$p(x) = \left(\left(\cdots\left((a_n x + a_{n-1})x + a_{n-2}\right)\cdots\right)x + a_1\right)x + a_0$$

```
p = a[0]
for j in range( n-1,-1,-1 ):
    p = p*x + a[j]
```

  - Reicht für mittellange Polynome völlig aus
  - Aufwand: n Multiplikationen / n Additionen

Auswertung an äquidistanten Stellen

- Definition: Vorwärts-Differenz

$$\Delta^1 p(x) := p(x+h) - p(x)$$
$$\Delta^i p(x) := \Delta^{i-1} p(x+h) - \Delta^{i-1} p(x)$$
$$\Delta^0 p(x) := p(x)$$

- Bemerkung: $\Delta^i p(x)$ ist ein Polynom vom Grad $n-i$
- Beweis:

$$\begin{aligned}\Delta^1 p(x) &= p(x+h) - p(x)\\ &= a_0 + \cdots + a_n (x+h)^n\\ &\quad - a_0 - \cdots - a_n x^n\\ &= a_0 + \cdots + a_n \sum_{i=0}^{n} \binom{n}{i} x^i h^{n-i}\\ &\quad - a_0 - \cdots - a_n x^n\\ &= a'_0 + \cdots + a'_{n-1} x^{n-1}\end{aligned}$$

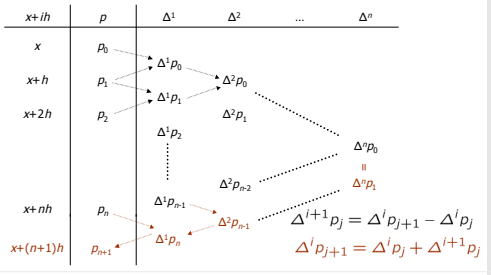
  - Rest per Induktion

- Korollar: $\Delta^n p(x) \equiv \text{const}$
  - Achtung: auf dieser Beobachtung basiert das ganze Verfahren!
- Bemerkung: $\Delta^i p(x)$ hängt ab von $p(x)$, $p(x+h)$, …, $p(x+ih)$
- Beispiel:

$$\begin{aligned}\Delta^2 p(x) &= \Delta^1 p(x+h) - \Delta^1 p(x)\\ &= \left(\Delta^0 p(x+h+h) - \Delta^0 p(x+h)\right)\\ &\quad - \left(\Delta^0 p(x+h) - \Delta^0 p(x)\right)\\ &= p(x+2h) - 2p(x+h) + p(x)\end{aligned}$$

Das Verfahren

- Vorwärtsdifferenzen-Pyramide:
  - vereinfachende Schreibweise $\Delta^i p(x + jh) =: \Delta^i p_j$
  - die Pyramide:



- Algo zur Berechnung von vielen äquidistanten Werten von p:
  1. initialisieren die Pyramide mit den Punkten $x, \ldots x + nh$
  2. berechne den nächsten Punkt an der Stelle $x + (n+1)h$ durch Fortsetzen der Pyramide um eine Zeile am unteren Ende
- Aufwand: für jeden neuen Punkt braucht man nur n Additionen (keine Multiplikation)
- Bemerkungen:
  - man muß eigentlich nur die unterste Zeile der Pyramide speichern
  - Rundungsfehler akkumulieren sich → Pyramide ab und zu neu aufsetzen
- Durch Umformung kann man ein Polynom vom Grad n mit $\frac{n}{2}$ Multiplikationen an beliebigen Stellen auswerten (s. Knuth)