Datenstrukturen und Algorithmen

Übung 7

FS 2018

Programm von heute

1. Feedback letzte Übung(en)
2. Wiederholung Theorie

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q$

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q \rightarrow$ nicht passend: $(k = 0) \mapsto 0$

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q \rightarrow$ nicht passend: $(k = 0) \mapsto 0$
- $s(j,k) = k^j \bmod p$

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q \rightarrow$ nicht passend: $(k = 0) \mapsto 0$
- $s(j,k) = k^j \bmod p \rightarrow$ nicht passend: $(k = 0) \mapsto 0, (k = 1) \mapsto 1$

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q \rightarrow$ nicht passend: $(k=0) \mapsto 0$
- $s(j,k) = k^j \bmod p \rightarrow$ nicht passend: $(k=0) \mapsto 0, (k=1) \mapsto 1$
- $s(j,k) = ((k \cdot j) \bmod q) + 1$

Nachbesprechung

Open hashing:

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q \rightarrow$ nicht passend: $(k = 0) \mapsto 0$
- $s(j,k) = k^j \bmod p \rightarrow$ nicht passend: $(k = 0) \mapsto 0, (k = 1) \mapsto 1$
- $s(j,k) = ((k \cdot j) \bmod q) + 1 \rightarrow$ nicht passend: 1 wenn k Vielfaches von q, und Bereich $p - q$ nicht abgedeckt.

Nachbesprechung

Coocoo hashing

- $h_1(k) = k \bmod 5$, $h_2(k) = \lfloor k/5 \rfloor \bmod 5$
- Hinzufügen von 27, 2, 32

```
T_1: __, __, 27, __, __          T_2: __, __, __, __, __

T_1: __, __,  2, __, __          T_2: 27, __, __, __, __

T_1: __, __, 27, __, __          T_2:  2, 32, __, __, __
```

Nachbesprechung

Coocoo hashing

- $h_1(k) = k \bmod 5$, $h_2(k) = \lfloor k/5 \rfloor \bmod 5$
- Hinzufügen von 7: Endlosschleife

```
     T_1: __, __, 27, __, __    T_2:  2, 32, __, __, __
 7:  T_1: __, __,  7, __, __    T_2: 27, 32, __, __, __
 2:  T_1: __, __,  2, __, __    T_2: 27,  7, __, __, __
32:  T_1: __, __, 32, __, __    T_2:  2,  7, __, __, __
27:  T_1: __, __, 27, __, __    T_2:  2, 32, __, __, __
 7:   ...
```

Nachbesprechung

Finden eines Sub-Arrays

```
// calculating hash_a, hash_b, c_to_k
It1 window_end = from;
for(It2 current = begin; current != end;
    ++current, ++window_end) {
  if(window_end == to) return to;
  hash_b = (C * hash_b % M + *current) % M;
  hash_a = (C * hash_a % M + *window_end) % M;
  c_to_k = c_to_k * C % M;
}
```

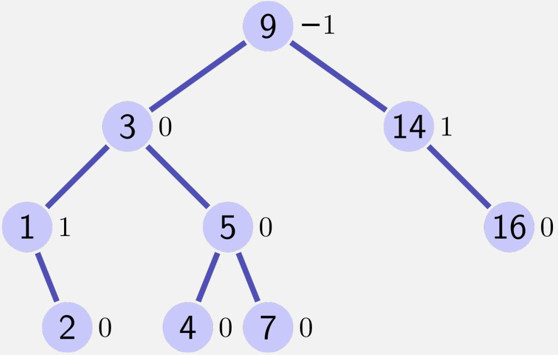
Nachbesprechung

Finden eines Sub-Arrays

```
// looking for b and updating hash_a
for(It1 window_begin = from;
        ; ++window_begin, ++window_end) {
  if(hash_a == hash_b)
    if(std::equal(window_begin, window_end, begin, end))
      return window_begin;
  if(window_end == to) return to;
  hash_a = (C * hash_a % M + *window_end
             + (M - c_to_k) * *window_begin % M) % M;
}
```

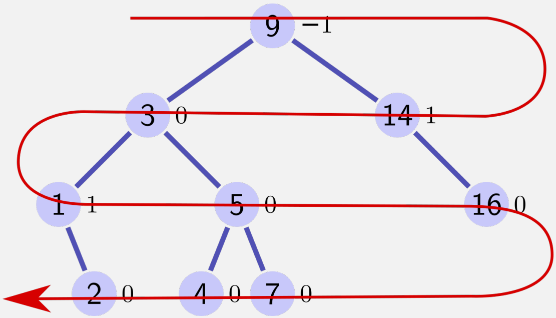
AVL Einfügesequenz

- Gegeben ein AVL Baum: Gibt es eine Einfügesequenz die den selben Baum erstellt und keine Rotation benötigt?



AVL Einfügesequenz

- Gegeben ein AVL Baum: Gibt es eine Einfügesequenz die den selben Baum erstellt und keine Rotation benötigt?



AVL Einfügesequenz - Beweisskizze

- Alle Sequenzen die die Höhenreihenfolge nicht ändern sind i.O.
- Beweis?
- Induktion über Baumhöhe

AVL Einfügesequenz - Beweisskizze

- Alle Sequenzen die die Höhenreihenfolge nicht ändern sind i.O.
- Beweis?
- Induktion über Baumhöhe
- Hypothese: Schlüssel an Höhe h und tiefer sind korrekt platziert und ihre Einfügeoperation verursacht keine Rotation.

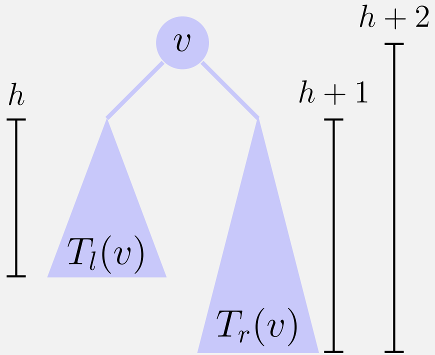
AVL Einfügesequenz - Beweisskizze

- Alle Sequenzen die die Höhenreihenfolge nicht ändern sind i.O.
- Beweis?
- Induktion über Baumhöhe
- Hypothese: Schlüssel an Höhe h und tiefer sind korrekt platziert und ihre Einfügeoperation verursacht keine Rotation.
- Schritt: Zeige dass Traversierung gleich ist wie im Originalbaum, ergibt gleiche Positionierung. Dann, benutze AVL Eigenschaft um zu zeigen dass nie einen Höhenunterschied grösser als 1 eintreten kann und deshalb keine Rotationen nötig sind.

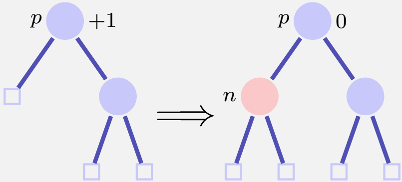
2. Wiederholung Theorie

AVL Bedingung

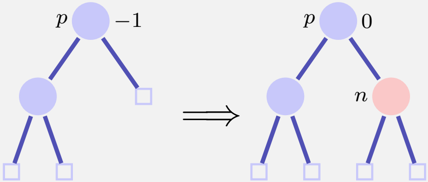
AVL Bedingung: für jeden Knoten v *eines Baumes gilt* $\mathrm{bal}(v) \in \{-1, 0, 1\}$



Balance am Einfügeort



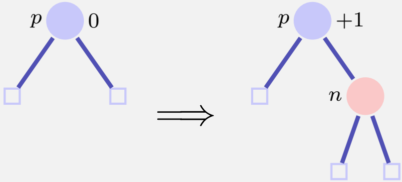
Fall 1: $\mathrm{bal}(p) = +1$



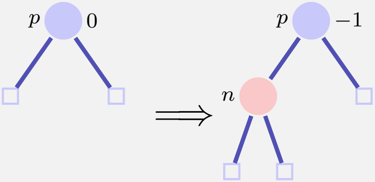
Fall 2: $\mathrm{bal}(p) = -1$

Fertig in beiden Fällen, denn der Teilbaum ist nicht gewachsen.

Balance am Einfügeort



Fall 3.1: $\mathrm{bal}(p) = 0$ rechts



Fall 3.2: $\mathrm{bal}(p) = 0$, links

In beiden Fällen noch nicht fertig. Aufruf von `upin(p)`.

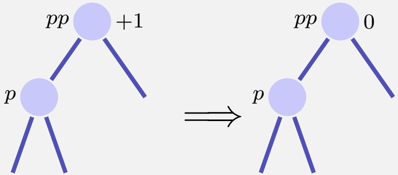
upin(p) - Invariante

Beim Aufruf von `upin(p)` gilt, dass

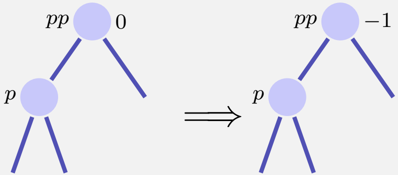
- der Teilbaum ab p gewachsen ist und
- $\mathrm{bal}(p) \in \{-1, +1\}$

upin(p)

Annahme: p ist linker Sohn von pp[1]



Fall 1: $\mathrm{bal}(pp) = +1$, fertig.



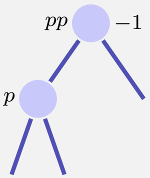
Fall 2: $\mathrm{bal}(pp) = 0$, **`upin(pp)`**

In beiden Fällen gilt nach der Operation die AVL-Bedingung für den Teilbaum ab pp

[1] Ist p rechter Sohn: symmetrische Fälle unter Vertauschung von $+1$ und -1

upin(p)

Annahme: p ist linker Sohn von pp



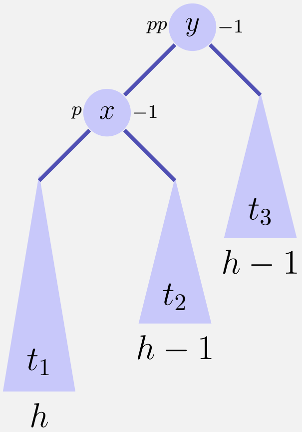
Fall 3: $\mathrm{bal}(pp) = -1$,

Dieser Fall ist problematisch: das Hinzufügen von n im Teilbaum ab pp hat die AVL-Bedingung verletzt. Rebalancieren!

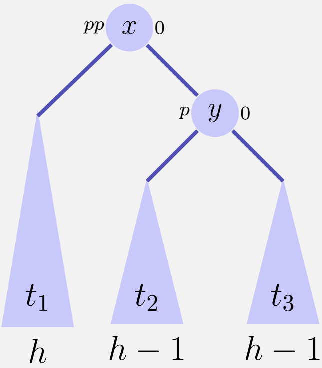
Zwei Fälle $\mathrm{bal}(p) = -1$, $\mathrm{bal}(p) = +1$

Rotationen

Fall 1.1 $\mathrm{bal}(p) = -1.$ [2]



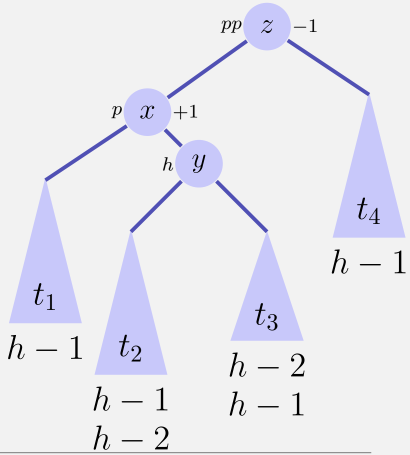
$\Longrightarrow$ Rotation nach rechts



[2] p rechter Sohn: $\mathrm{bal}(pp) = \mathrm{bal}(p) = +1$, Linksrotation

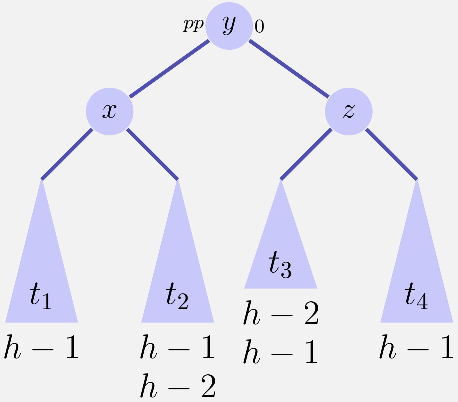
Rotationen

Fall 1.2 $\mathrm{bal}(p) = +1$. [3]



$\Longrightarrow$

Doppel-
rotation
links-
rechts



[3] p rechter Sohn: $\mathrm{bal}(pp) = +1$, $\mathrm{bal}(p) = -1$, Doppelrotation rechts links

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**:

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**: Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Dynamische Programmierung

Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte :

- **Definition der DP-Tabelle**: Welche Dimensionen hat die Tabelle? Was ist die Bedeutung jedes Eintrags?
- **Berechnung eines Eintrags**: Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
- **Berechnungsreihenfolge**: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?
- **Auslesen der Lösung**: Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Längste aufsteigende Sequenz in Matrix

Gegeben $n \times m$ Matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Längste aufsteigende Sequenz in Matrix

Gegeben $n \times m$ Matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Gesucht längste aufsteigende Sequenz:

$$4, 6, 28, 29, 47, 49$$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m(\times 2)$

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m (\times 2)$
- Was ist die Bedeutung jedes Eintrags?

Definition der DP-Tabelle

- Welche Dimensionen hat die Tabelle?
  - $n \times m (\times 2)$
- Was ist die Bedeutung jedes Eintrags?
  - In $T[x][y]$ steht Länge der längsten aufsteigenden Sequenz, die im Feld $A[x][y]$ endet
  - In $S[x][y]$ steht Koordinaten des Vorgängers von (x, y) in aufsteigender Sequenz (falls existänt)

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T
  - Aktuallisiere T und S. (S erhält Koordinaten vom ausgewählten Nachbar, T erhält Wert um eins erhöht vom ausgewählten Nachbar.)

Berechnung eines Eintrags

- Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?
  - Betrachte Nachbarn mit kleineren Eintrag in A.
  - Wähle von den kleineren Einträgen den mit dem grössten Eintrag in T
  - Aktuallisiere T und S. (S erhält Koordinaten vom ausgewählten Nachbar, T erhält Wert um eins erhöht vom ausgewählten Nachbar.)

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

- Beginne mit kleinstem Element in A und so weiter. (Bedeutet dass man A sortieren muss.)

Berechnungsreihenfolge

- In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

- Beginne mit kleinstem Element in A und so weiter. (Bedeutet dass man A sortieren muss.)

- Beliebige Reihenfolge, falls Eintrag schon berechnet überspringen sonst rekursiv von kleinern Nachbarn berechnen.

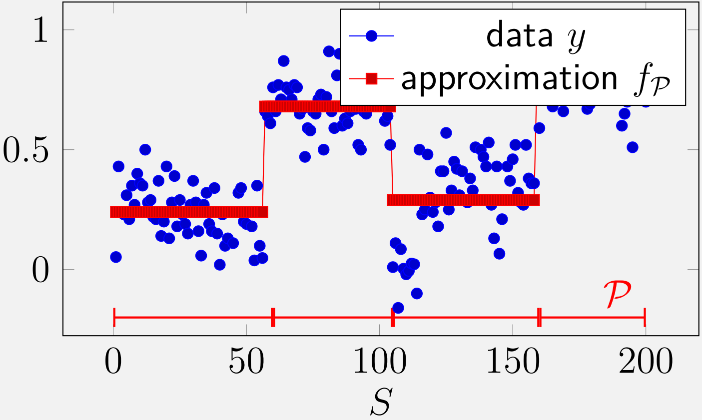
Auslesen der Lösung

- Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Auslesen der Lösung

- Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?
    - Betrachte alle Einträge um den Eintrag zu finden, in dem eine längste Sequenz endet. Von dort aus können wir die Lösung rekonstruieren, indem wir dem entsprechenden Vorgänger folgen.

Stückweise konstante Approximation



Stückweise konstante Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

Stückweise konstante Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: Partition von S (Menge von Intervallen I_i, so dass $\bigcup_i I_i = S$).
- **Ziel:** Finde die Partition $\hat{\mathcal{P}}$, so dass $H_{\gamma,y}(\hat{\mathcal{P}})$ minimal
- Nutze aus: effizientes Berechnen von Durchschnitten mit Präfixsummen (Übung 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$

Stückweise konstante Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: Partition von S (Menge von Intervallen I_i, so dass $\bigcup_i I_i = S$).
- **Ziel:** Finde die Partition $\hat{\mathcal{P}}$, so dass $H_{\gamma,y}(\hat{\mathcal{P}})$ minimal
- Nutze aus: effizientes Berechnen von Durchschnitten mit Präfixsummen (Übung 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$
- Nutze aus: effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$

Stückweise konstante Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: Partition von S (Menge von Intervallen I_i, so dass $\bigcup_i I_i = S$).
- **Ziel:** Finde die Partition $\hat{\mathcal{P}}$, so dass $H_{\gamma,y}(\hat{\mathcal{P}})$ minimal
- Nutze aus: effizientes Berechnen von Durchschnitten mit Präfixsummen (Übung 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$
- Nutze aus: effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$

Stückweise konstante Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- **Ziel:** Finde die Partition $\hat{\mathcal{P}}$, so dass $H_{\gamma,y}(\hat{\mathcal{P}})$ minimal
- **Dynamische Programmierung**: Definition der Tabelle, Berechnung eines Eintrags, Berechnungsreihenfolge, Auslesen der Lösung
- Nutze aus[§]: $H_{\gamma,y}(\mathcal{P} \cup \{[l,r)\}) = H_{\gamma,y}(\mathcal{P}) + \gamma + e_{[l,r)}$

[§]Vorraussetzung: $\mathcal{P} \cup \{[l,r)\}$ ist eine Partition

Fragen?