Datenstrukturen und Algorithmen

Übung 8

FS 2018

Programm von heute

1. Quiz
2. Feedback letzte Übung

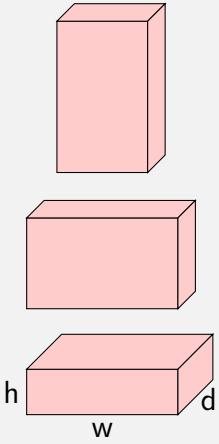
1. Quiz

Quiz: Boxen Stapeln

- Gegeben: n Boxen mit Grössen $w_i \times d_i \times h_i$
- Gesucht: maximale Höhe eines erlaubten Stapels
- Erlaubter Stapel: Grundfläche gestapelter Boxen muss in beiden Richtungen (Breite und Tiefe) strikt kleiner werden

Boxen Stapeln

Wir gehen davon aus, dass es genügend Boxen jeder Sorte gibt, so dass jede Box in jeder Orientierung verfügbar ist (Abbildung rechts).

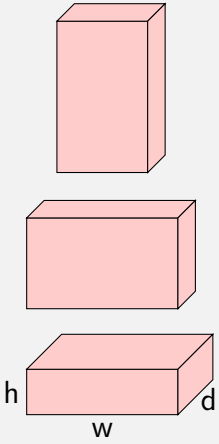


Box	1	2	3	4	5	6
$[w \times d \times h]$	$[1 \times 2 \times 3]$	$[1 \times 3 \times 2]$	$[2 \times 3 \times 1]$	$[3 \times 4 \times 5]$	$[3 \times 5 \times 4]$	$[4 \times 5 \times 3]$

Boxen Stapeln

Wir gehen davon aus, dass es genügend Boxen jeder Sorte gibt, so dass jede Box in jeder Orientierung verfügbar ist (Abbildung rechts).

Erstellen Sie einen DP Algorithmus zum Finden der maximalen Höhe eines erlaubten Stapels



Box	1	2	3	4	5	6
$[w \times d \times h]$	$[1 \times 2 \times 3]$	$[1 \times 3 \times 2]$	$[2 \times 3 \times 1]$	$[3 \times 4 \times 5]$	$[3 \times 5 \times 4]$	$[4 \times 5 \times 3]$

Lösungsidee

- $n \times n$ Tabelle
- Eintrag in Zeile i, Spalte j: Höhe eines höchsten Turmes mit maximal i Boxen und Basisbox j.

$[w \times d]$	$[1 \times 2]$	$[1 \times 3]$	$[2 \times 3]$	$[3 \times 4]$	$[3 \times 5]$	$[4 \times 5]$
h	3	2	1	5	4	3
1	<u>3</u>	2	1	5	4	3
2	3	2	<u>4</u>	8	8	8
3	3	2	4	<u>9</u>	8	11
4	3	2	4	9	8	<u>12</u>

Bestimmung der Tabelle: $\Theta(n^3)$, für jeden Eintrag müssen alle Einträge der vorherigen Zeile durchlaufen werden.

Berechnung der optimalen Lösung durch Rückverfolgung im schlechtesten Fall $\Theta(n^2)$.

Alternative Lösungsidee

- $1 \times n$ Tabelle, topologisch sortiert[1] nach Halbordnung Stapelbarkeit
- Eintrag an Position j: Höhe eines höchsten Turmes mit Basisbox j.

$[w \times d]$	$[1 \times 2]$	$[1 \times 3]$	$[2 \times 3]$	$[3 \times 4]$	$[3 \times 5]$	$[4 \times 5]$
h	3	2	1	5	4	3
	3	2	4	9	8	12

Topologisches Sortieren in $\Theta(n^2)$. Durchlaufen von rechts nach links in $\Theta(n)$, insgesamt $\Theta(n^2)$. Rückverfolgung auch $\Theta(n^2)$

[1] Erklärung folgt

2. Feedback letzte Übung

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

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten: $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$

Stückweise Konstante Approximation

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten: $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i \Rightarrow$ prefixsum ✓

Stückweise Konstante Approximation

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten: $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i \Rightarrow$ prefixsum ✓
- Effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$

Stückweise Konstante Approximation

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten:
  $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i \Rightarrow$ prefixsum ✓
- Effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$
  $\Rightarrow e_{[l,r)} = \sum_{i=l}^{r-1} y_i^2 - \frac{1}{r-l} \left(\sum_{i=l}^{r-1} y_i \right)^2$ ✓

Stückweise Konstante Approximation

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten: $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i \Rightarrow$ prefixsum ✓
- Effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$
  $\Rightarrow e_{[l,r)} = \sum_{i=l}^{r-1} y_i^2 - \frac{1}{r-l} \left(\sum_{i=l}^{r-1} y_i \right)^2$ ✓
- **Dynamische Programmierung**: Definition der Tabelle, Berechnung eines Eintrags, Berechnungsreihenfolge, Auslesen der Lösung

Stückweise Konstante Approximation

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

- Wie in Übung 1 effizientes berechnen von Durchschnitten: $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i \Rightarrow$ prefixsum ✓
- Effizientes Berechnen von $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$
  $\Rightarrow e_{[l,r)} = \sum_{i=l}^{r-1} y_i^2 - \frac{1}{r-l} \left(\sum_{i=l}^{r-1} y_i \right)^2$ ✓
- **Dynamische Programmierung**: Definition der Tabelle, Berechnung eines Eintrags, Berechnungsreihenfolge, Auslesen der Lösung $\Rightarrow$ **?**

Dynamische Programmierung

- **Definition der DP-Tabelle**: zwei Tabellen: B und V mit jeweils $n+1 \times 1$ Einträgen, $B[k]$ beinhaltet Zeiger zum Besten vorherigen Intervall, $V[k]$ beinhaltet entsprechendes Minimum von H_γ.
- **Berechnung eines Eintrags**: um neuen Eintrag in $B[k+1]$ zu berechnen berechne alle H_γ für alle Partitionen von 0 bis $k+1$.
- **Berechnungsreihenfolge**: von links nach rechts
- **Auslesen der Lösung**: konstruiere Intervalle mit $B[n]$ von rechts nach links, Minimum ist gegeben durch $V[n]$

Summen

Gegeben Datenvektor y der Länge $n \in \mathbb{N}$: $(y_i)_{i=1\ldots n} \in \mathbb{R}^n$

Summe $m_n := \sum_{i=1}^{n} y_i \Rightarrow \mu_n = m_n/n$

Summe Quadrate $s_n := \sum_{i=1}^{n} y_i^2$

$$
\begin{aligned}
e_n &:= \sum_{i=1}^{n} (y_i - \mu_n)^2 = \sum_{i=1}^{n} y_i^2 - 2\mu_n y_i + \mu_n^2 \\
&= s_n - 2\mu_n \left(\sum_{i=1}^{n} y_i \right) + n \cdot \mu_n^2 = s_n - 2\mu_n \cdot n\mu_n + n \cdot \mu_n^2 \\
&= s_n - n \cdot \mu_n^2 = s_n - m_n^2/n
\end{aligned}
$$

Statistik

```
// post: return mean of data[from,to)
double mean(unsigned int from, unsigned int to) const{
    assert(from < to && to <= n);
    return getsum(vsum,from,to) / (to-from);
}

// post: return err of constant approximation in interval [from,to)
double err(unsigned int from, unsigned int to) const{
    assert(from < to && to <= n);
    double m = getsum(vsum,from,to);
    return getsum(vssq,from,to) - m*m / (to-from);
}
```

DP – Setup and Base Case

```
double MinimizeH(double gamma,const Statistics& s,
                 std::vector<double>& result){
    int n = s.size();
    // B[k] contains the pointer to the end of the best previous interval
    // i.e. best possible approximation is given by
    // best possible approximation of [0,B[k]), [B[k],k)
    std::vector<int> B(n+1);
    // V(k) contains the corresponsing attainable minimum of H_gamma
    std::vector<double> V(n+1);
    // base case: empty interval
    B[0] = 0;
    V[0] = 0;
```

DP – Construct Table

```
// now consider all combinations of Partition ([0, left )) + [left , right )
for (int right=1; right <= n; ++right){
    // interval [0, right )
    int best = 0;
    double min = gamma + s.err(0,right);
    // intervals [ left , right ), left > 0
    for (int left = 1; left < right ; ++left){
        double h = V[left] + gamma + s.err(left,right);
        if (h < min){
            min = h; best = left ;
        }
    }
    B[right] = best;
    V[right] = min;
}
```

DP – Reconstruct Solution

```
    // reconstruct solution
    unsigned int right=n;
    while (right != 0){
        unsigned int left  = B[right];
         fill (result,s.mean(left,right), left, right);
        right  = left;
    }
    return V[n];

}
```

Fragen?