

Assignment 11

Advanced Algorithms & Data Structures PS

Christian Müller 1123410
Daniel Kocher, 0926293

June 22, 2016

Aufgabe 21

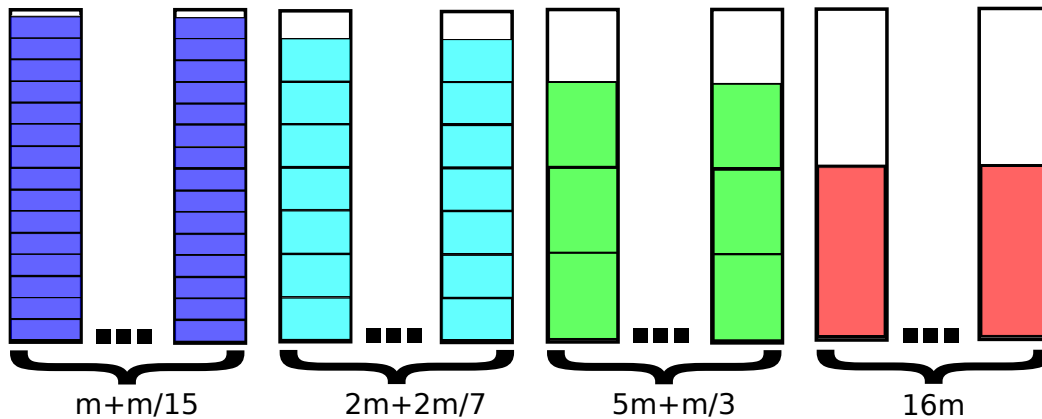
Gegeben sei folgende Sequenz I von Objekten

$$\frac{1}{16} + \epsilon, \dots, \frac{1}{16} + \epsilon, \frac{1}{8} + \epsilon, \dots, \frac{1}{8} + \epsilon, \frac{1}{4} + \epsilon, \dots, \frac{1}{4} + \epsilon, \frac{1}{2} + \epsilon, \dots, \frac{1}{2} + \epsilon \quad (1)$$

wobei jede Teilsequenz (Objekte der gleichen Größe) die Länge $16m$ hat.

Es gilt $m \in \mathbb{N} : 0 \equiv m \pmod{15} \wedge 0 \equiv m \pmod{7} \wedge 0 \equiv m \pmod{3}$ und $\epsilon < 10^{-6}$.

- Geben Sie $FF(I)$ sowie die zugehörige Packung an.
- Wenden Sie First-Fit-Decreasing auf I an. Geben Sie die zugehörige Packung sowie $FFD(I)$ an.



Aufteilung der $16m$ Elemente der Größe $\frac{1}{16} + \epsilon$.

Eine Runde sei das Einfügen von 16 Elementen. Also ist nach m Runden das Einfügen von $16m$ Elementen beendet.

Runde 1: $B_1 : 15 \cdot (\frac{1}{16} + \epsilon), B_2 : (\frac{1}{16} + \epsilon)$

Runde 2: $B_1 : 15 \cdot (\frac{1}{16} + \epsilon), B_2 : 15 \cdot (\frac{1}{16} + \epsilon), B_3 : 2 \cdot (\frac{1}{16} + \epsilon)$

Runde 3: $B_1 : 15 \cdot (\frac{1}{16} + \epsilon), B_2 : 15 \cdot (\frac{1}{16} + \epsilon), B_3 : 15 \cdot (\frac{1}{16} + \epsilon), B_4 : 3 \cdot (\frac{1}{16} + \epsilon)$

...

Runde 15: $B_1 : 15 \cdot (\frac{1}{16} + \epsilon), B_2 : 15 \cdot (\frac{1}{16} + \epsilon), B_3 : 15 \cdot (\frac{1}{16} + \epsilon), B_4 : 15 \cdot (\frac{1}{16} + \epsilon), \dots, B_{16} : 15 \cdot (\frac{1}{16} + \epsilon)$

wegen $0 \equiv m \pmod{15}$ gilt:

Runde m : $B_1 : 15 \cdot (\frac{1}{16} + \epsilon), B_2 : 15 \cdot (\frac{1}{16} + \epsilon), B_3 : 15 \cdot (\frac{1}{16} + \epsilon), B_4 : 15 \cdot (\frac{1}{16} + \epsilon), \dots, B_{m+\frac{m}{15}} : 15 \cdot (\frac{1}{16} + \epsilon)$

Alle $m + \frac{m}{15}$ Blöcke sind nach m Runden vollständig gefüllt. Nach m Runden gilt für alle Blöcke B_j mit $1 \leq j \leq m + \frac{m}{15}$ also:

$$B_j + \left(\frac{1}{16} + \epsilon\right) > 1 \quad (2)$$

Aufgrund der gegebenen Einfügereihenfolge bedeutet das also, dass diesen Blöcken kein weiteres Element der Restsequenz mehr hinzugefügt werden kann, da diese alle eine Größe haben, die $(\frac{1}{16} + \epsilon)$ übersteigt.

Aufteilung der $16m$ Elemente der Größe $\frac{1}{8} + \epsilon$.

B_k	Runde 1	Runde 2	...	Runde 7	...	Runde m
$(m + \frac{m}{15}) + 1$	$7 \cdot (\frac{1}{8} + \epsilon)$	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 2$	$7 \cdot (\frac{1}{8} + \epsilon)$	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 3$	$2 \cdot (\frac{1}{8} + \epsilon)$	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 4$	/	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 5$	/	$4 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 6$	/	/	...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
...
$(m + \frac{m}{15}) + 16$	/	/	...	$7 \cdot (\frac{1}{8} + \epsilon)$...	$7 \cdot (\frac{1}{8} + \epsilon)$
$(m + \frac{m}{15}) + 17$	/	/	...	/	...	$7 \cdot (\frac{1}{8} + \epsilon)$
...
$(m + \frac{m}{15}) + (2m + \frac{2m}{7})$	/	/	...	/	...	$7 \cdot (\frac{1}{8} + \epsilon)$

16 Objekte der Größe $(\frac{1}{8} + \epsilon)$ füllen (d.h. sodass kein weiteres Objekt dieser Größe Platz hat) 2 Blöcke und es bleiben 2 Objekte übrig. Im Rahmen von k Runden wächst dieser Rest auf $2k$ Objekte an. Um diese $2k$ zusätzlichen Objekte unterzubringen, werden $\lceil \frac{2k}{7} \rceil$ zusätzliche Blöcke benötigt.

Analog für $\frac{1}{4} + \epsilon$ und $\frac{1}{2} + \epsilon$

$$\begin{aligned}
FF(I) &= m + \frac{m}{15} + 2m + \frac{2m}{7} + 5m + \frac{m}{3} + 16m \\
&= 24m + \frac{m}{15} + \frac{2m}{7} + \frac{m}{3} \\
&= 24m + \frac{7m + 30m + 35m}{105} \\
&= 24m + \frac{72m}{105} = 24m + \frac{24m}{35}
\end{aligned}$$

Für das Einfügen der Sequenz I ($16m \cdot 4 = 64m$ Elemente) werden $24m + \lceil \frac{24m}{35} \rceil$ Container benötigt.

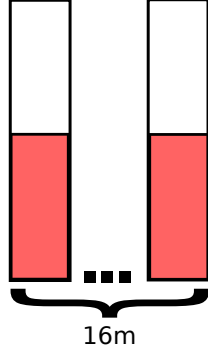
Wenden Sie First-Fit-Decreasing auf I an. Geben Sie die zugehörige Packung sowie $FFD(I)$ an. Sortiere Sequenz I , sodass die Elemente in absteigender Reihenfolge vorliegen:

$$\frac{1}{2} + \epsilon, \dots, \frac{1}{2} + \epsilon, \frac{1}{4} + \epsilon, \dots, \frac{1}{4} + \epsilon, \frac{1}{8} + \epsilon, \dots, \frac{1}{8} + \epsilon, \frac{1}{16} + \epsilon, \dots, \frac{1}{16} + \epsilon \quad (3)$$

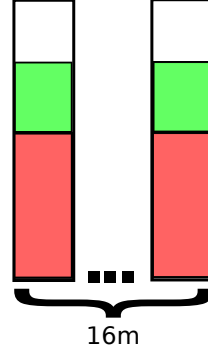
wobei jede Teilsequenz (Objekte der gleichen Größe) die Länge $16m$ hat.

Es gilt $m \in \mathbb{N} : 0 \equiv m \pmod{15} \wedge 0 \equiv m \pmod{7} \wedge 0 \equiv m \pmod{3}$ und $\epsilon < 10^{-6}$.

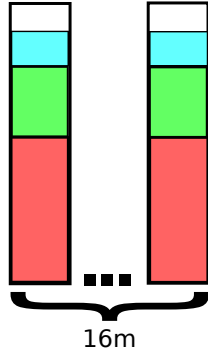
Nach Einfügen der $16m \frac{1}{2} + \epsilon$ Objekte.



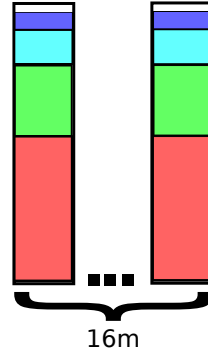
Nach Einfügen der $16m \frac{1}{4} + \epsilon$ Objekte.



Nach Einfügen der $16m \frac{1}{8} + \epsilon$ Objekte.



Nach Einfügen der $16m \frac{1}{16} + \epsilon$ Objekte.



Keine zwei (gleich großen) Elemente einer $16m$ Objekte langen Teilsequenz stehen im selben Block.

$$\begin{aligned} & \left(\frac{1}{2} + \epsilon\right) + \left(\frac{1}{2} + \epsilon\right) > 1 \\ \implies & \left(\frac{1}{2} + \epsilon\right) + 2 \cdot \left(\frac{1}{4} + \epsilon\right) \\ = & \left(\frac{1}{2} + \epsilon\right) + \left(\frac{1}{4} + \epsilon\right) + 2 \cdot \left(\frac{1}{8} + \epsilon\right) \\ = & \left(\frac{1}{2} + \epsilon\right) + \left(\frac{1}{4} + \epsilon\right) + \left(\frac{1}{8} + \epsilon\right) + 2 \cdot \left(\frac{1}{16} + \epsilon\right) > 1 \end{aligned}$$

Bei der angegebenen Belegung wird die Kapazität keines Blocks überschritten:

$$\begin{aligned}
 & \left(\frac{1}{2} + \epsilon\right) + \left(\frac{1}{4} + \epsilon\right) + \left(\frac{1}{8} + \epsilon\right) + \left(\frac{1}{16} + \epsilon\right) \\
 = & \left(\frac{8}{16} + \epsilon\right) + \left(\frac{4}{16} + \epsilon\right) + \left(\frac{2}{16} + \epsilon\right) + \left(\frac{1}{16} + \epsilon\right) \\
 & = \left(\frac{15}{16} + 4\epsilon\right) < 1
 \end{aligned}$$

da:

$$\epsilon < \frac{1}{10^6} \implies 4\epsilon < \frac{1}{16} \tag{4}$$

$$FFD(I) = OPT(I) = 16m$$

Aufgabe 22

Geben Sie ein dynamisches Programm für die Berechnung von $\binom{n}{i}$ an.

Hinweis: In der Vorlesung haben wir gezeigt, dass die Anzahl der Knoten mit Tiefe i in einem Binomialbaum B_n genau $\binom{n}{i}$ entspricht. Benutzen Sie die Argumente aus dem Beweis dieser Aussage.

Aus dem Beweis in der VO zu Binomial Queues bzw. Binomialbäumen (*Es gibt genau $\binom{n}{i}$ Knoten mit Tiefe i in B_n*) entnehmen wir folgende Argumente:

$$\binom{n}{0} = 1 \text{ und } \binom{n}{n} = 1 \quad (5)$$

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1} \quad (6)$$

Diese beiden Gleichungen sind bereits eine rekursive Darstellung des Binomialkoeffizienten $\binom{n}{i}$. Für das dynamische Programm definieren wir nun analog dazu:

```
binom(n, 0) = 1
```

```
binom(n, n) = 1
```

```
binom(n, i) = binom(n - 1, i) + binom(n - 1, i - 1)
```

Weiters benötigen wir eine $(n+1) \times (i+1)$ -Matrix `coeff` [], in der wir bereits berechnete Ergebnisse speichern (Memorization-Array). +1, da i bzw. n auch den Wert 0 annehmen können.

Mit diesen Bestandteilen lässt sich das dynamische Programm formulieren. Das dynamische Programm hat quadratische Laufzeit (in n) - im Gegensatz zu einer naiven Lösung, welche exponentielle Laufzeit (in n) hätte.

<pre> Input : n: integer, i: integer Output: $\binom{n}{i}$: integer 1 Function <i>binom</i>(n, i) 2 for $k \leftarrow 0$ to n do 3 for $l \leftarrow 0$ to i do 4 $\text{coeff}[k][l] \leftarrow \infty$; 5 end 6 end 7 for $k \leftarrow 0$ to n do 8 $\text{coeff}[k][0] \leftarrow 1$; 9 end 10 for $l \leftarrow 1$ to i do 11 $\text{coeff}[0][l] \leftarrow 0$; 12 end 13 return <i>lookupBinom</i>(n, i); </pre>	<pre> Input : n: integer, i: integer Output: $\binom{n}{i}$: integer 1 Function <i>lookupBinom</i>(n, i) 2 if $\text{coeff}[n][i] = \infty$ then 3 $\text{coeff}[n][i] \leftarrow \text{lookupBinom}(n-1, i) + \text{lookupBinom}(n-1, i-1)$; 4 end 5 return $\text{coeff}[n][i]$; </pre>
--	--

Eine weitere Lösung welche mit weniger Speicher auskommt ist durch folgenden Algorithmus gegeben.
Sei `coeff` nun ein eindimensionales Array der Länge n .

```

Input  :  $n$ : integer,  $i$ : integer
Output:  $\binom{n}{i}$ : integer
1 Function binom( $n$ ,  $i$ )
2   if  $i > n$  then
3     return 0;
4   end
5   coeff[0]  $\leftarrow$  1;
6   for  $k \leftarrow 1$  to  $n$  do
7     coeff[ $k$ ]  $\leftarrow$  1;
8     for  $j \leftarrow k - 1$  to 1 do
9       coeff[ $j$ ]  $\leftarrow$  coeff[ $j$ ] + coeff[ $j - 1$ ];
10    end
11  end
12  return coeff[ $i$ ];

```

Belegung von `coeff` nach der k -ten Iteration der äußeren Schleife:

$k = 1$:

1	1	0	0	0	0	0	0	...	0
---	---	---	---	---	---	---	---	-----	---

$k = 2$:

1	2	1	0	0	0	0	0	...	0
---	---	---	---	---	---	---	---	-----	---

$k = 3$:

1	3	3	1	0	0	0	0	...	0
---	---	---	---	---	---	---	---	-----	---

$k = 4$:

1	4	6	4	1	0	0	0	...	0
---	---	---	---	---	---	---	---	-----	---

..
..
..

Die Belegung von `coeff` nach dem k -ten Durchlauf entspricht also der k -ten Zeile des Pascalschen Dreiecks. Das i -te Element des nach der n -ten Iteration erhaltenen Koeffizientenarrays entspricht so der Lösung von $\binom{n}{i}$.