Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2 – Übung #1

Ramin Kosfeld und Chek-Manh Loi

17.04.2024

Orga - Kleine Übungen

Übungsgruppen

Die kleinen Übungen finden im Informatikzentrum in Raum IZ 161 und IZ 305 statt.

Gruppe	Termin (siehe Semesterplan)	Raum	Tutor*in
01	Dienstag, 13:15 – 14:45	IZ 305	Finn Senft
02	Mittwoch, 13:15 – 14:45	IZ 305	Henrik Heitmann
03	Mittwoch, 16:45 – 18:15	IZ 305	Finn Senft
04	Mittwoch, 16:45 – 18:15	IZ 161	Lisa Glowczewski
05	Donnerstag, 13:15 – 14:45*	IZ 161	Lisa Glowczewski
06	Freitag, 11:30 – 13:00	IZ 305	Abdelrahman Anbar
07	Freitag, 13:15 – 14:45	IZ 305	Henrik Heitmann

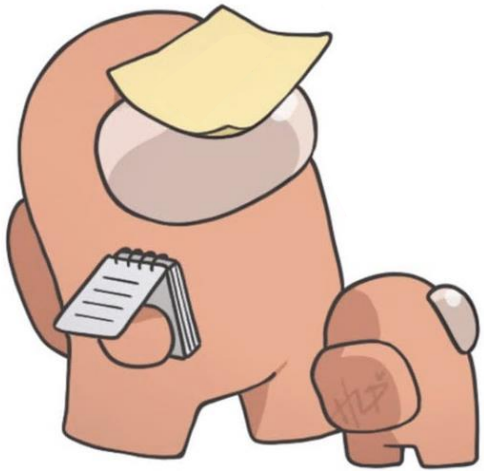
Heute:

Wir bauen klassische Algorithmen für “einfache” Probleme
(die man damit tatsächlich optimal gelöst kriegt.)

Thema: Hörsaalvergabe

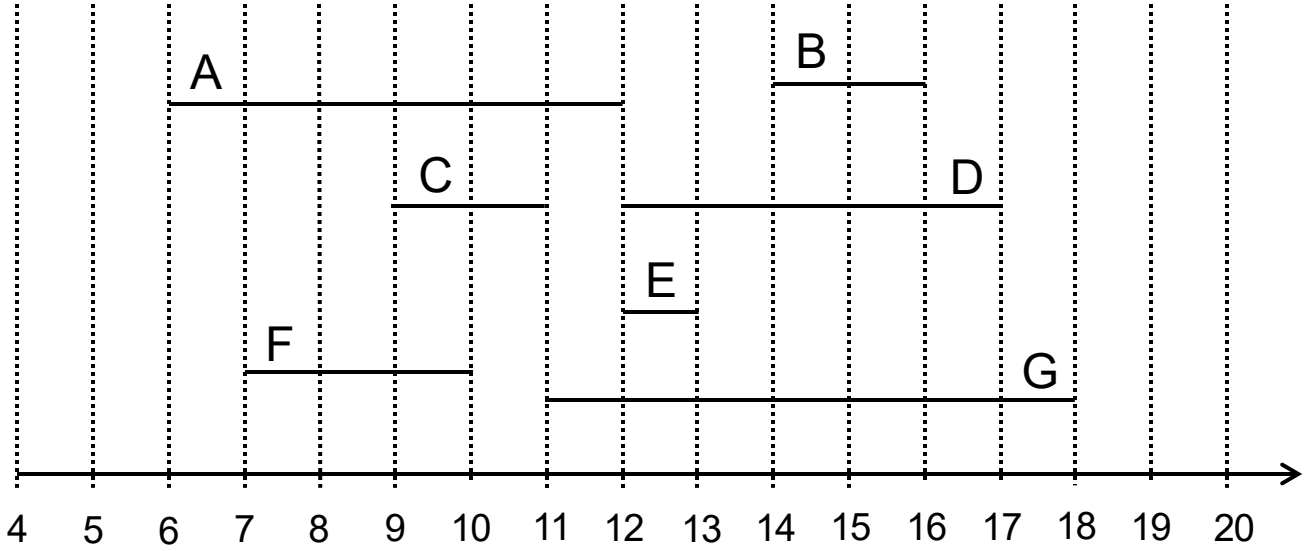
Raum

Oh man. Wir haben doch nur diesen einen Hörsaal …



Dann versuchen wir, möglichst viele Veranstaltungen unterzukriegen!

Hörsaal-Belegung



Hörsaal-Belegung – Das Problem

Gegeben

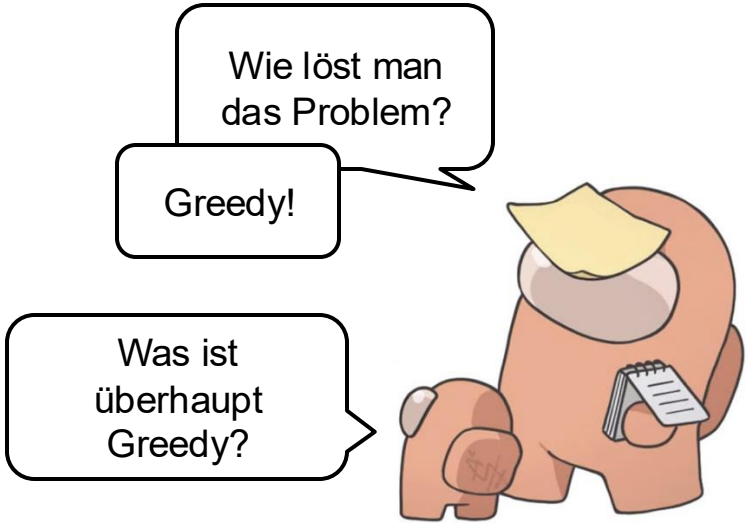
Menge von Intervallen $\mathcal{I} = \{I_1 = [s_1, e_1), \dots, I_n = [s_n, e_n)\}$

Gesucht

Teilmenge $\mathcal{I}' \subseteq \mathcal{I}$ mit den folgenden Eigenschaften

1. $\forall I_i, I_j \in \mathcal{I}' : I_i \cap I_j = \emptyset$
2. $\mathcal{I}'$ ist größtmöglich

Die ausgewählten Intervalle müssen paarweise disjunkt sein.



Eingabe: $z_i, ..., z_n, Z, p_i, ..., p_n$

Ausgabe: $x_i, ..., x_n \in [0, 1]$

mit

$$\sum_{i=1}^{n} z_i x_i \leq Z$$

und

$$\sum_{i=1}^{n} p_i x_i = Maximal$$

1: *Sortiere {1,...,n} nach* $\frac{z_i}{p_i}$ *aufsteigend;*
Dies ergibt die Permutation $\pi(1), ..., \pi(n)$.
Setze $j = 1$.

2: ***while*** $(\sum_{i=1}^{j} z_{\pi(i)} \leq Z)$ ***do***

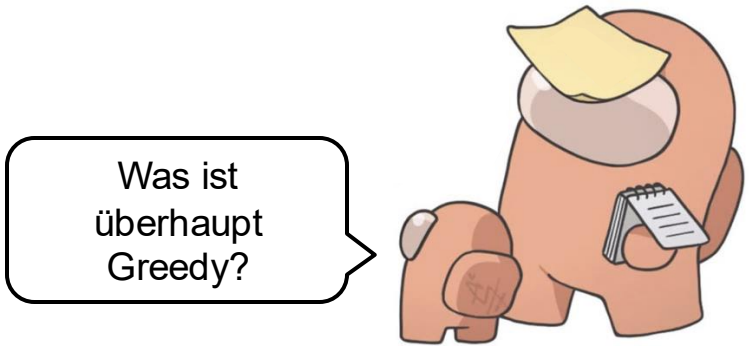
3: $\quad x_{\pi(j)} := 1$

4: $\quad j := j + 1$

5: *Setze* $x_{\pi(j)} := \dfrac{Z - \sum_{i=1}^{j-1} z_{\pi(i)}}{z_{\pi(j)}}$

6: ***return***

- Greedy-Algorithmen sind “gierig” bei ihren Entscheidungen:
- “Nimm das beste, was gerade geht”
- “Und gib es nicht wieder her” – Getroffene Entscheidungen werden nicht mehr rückgängig gemacht.



Hier:

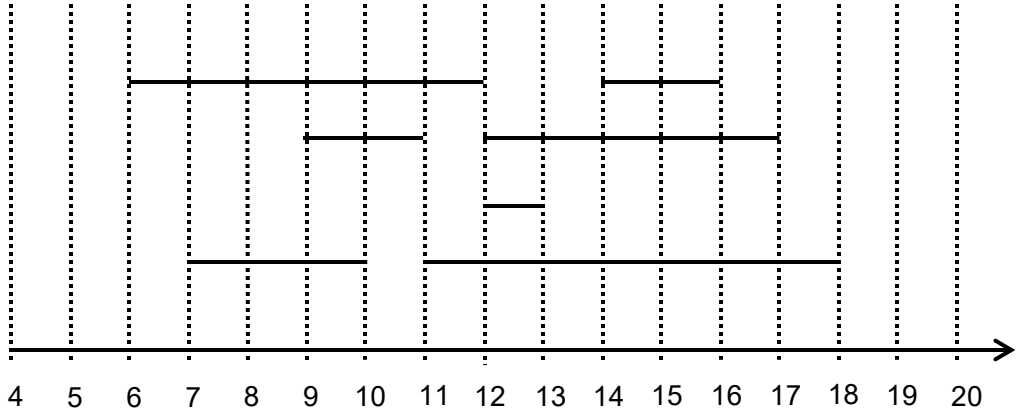
- Sortiere alle Intervalle nach einem Kriterium
- Nimm in dieser Reihenfolge jedes Intervall mit, was reinpasst.

→ Welches Kriterium?

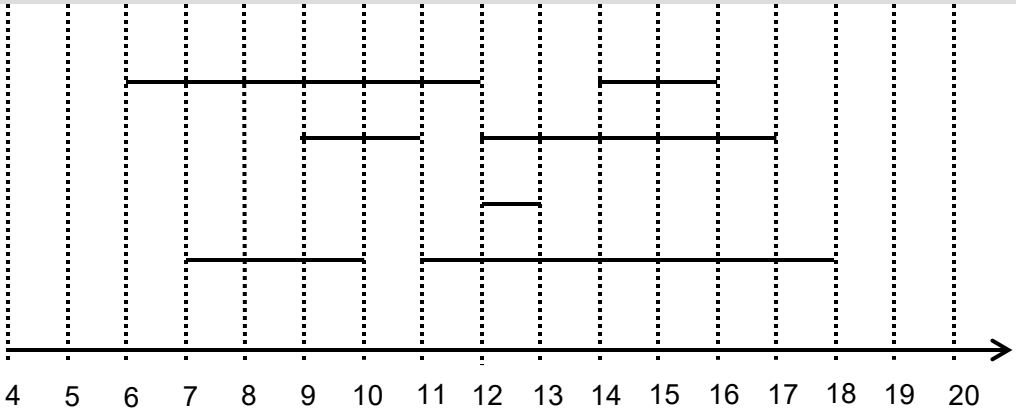
Hier:

- Sortiere alle Intervalle nach einem Kriterium
- Nimm in dieser Reihenfolge jedes Intervall mit, was reinpasst.

→ Welches Kriterium?



Hörsaal-Belegung – Strategien



Welche Greedyfunktion?

Kürzeste zuerst?

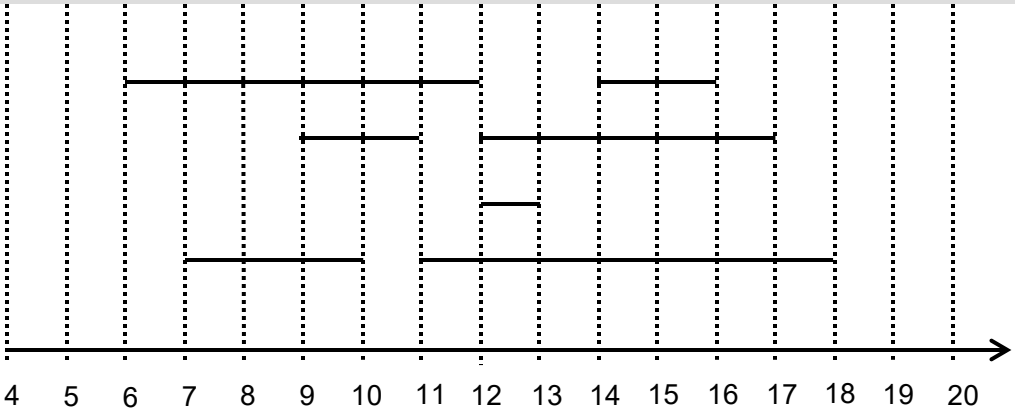
Längste zuerst?

Frühester Start?

Frühestes Ende?

Wenig Überlappungen?

Hörsaal-Belegung – Strategien



Welche Greedyfunktion?

Kürzeste zuerst?

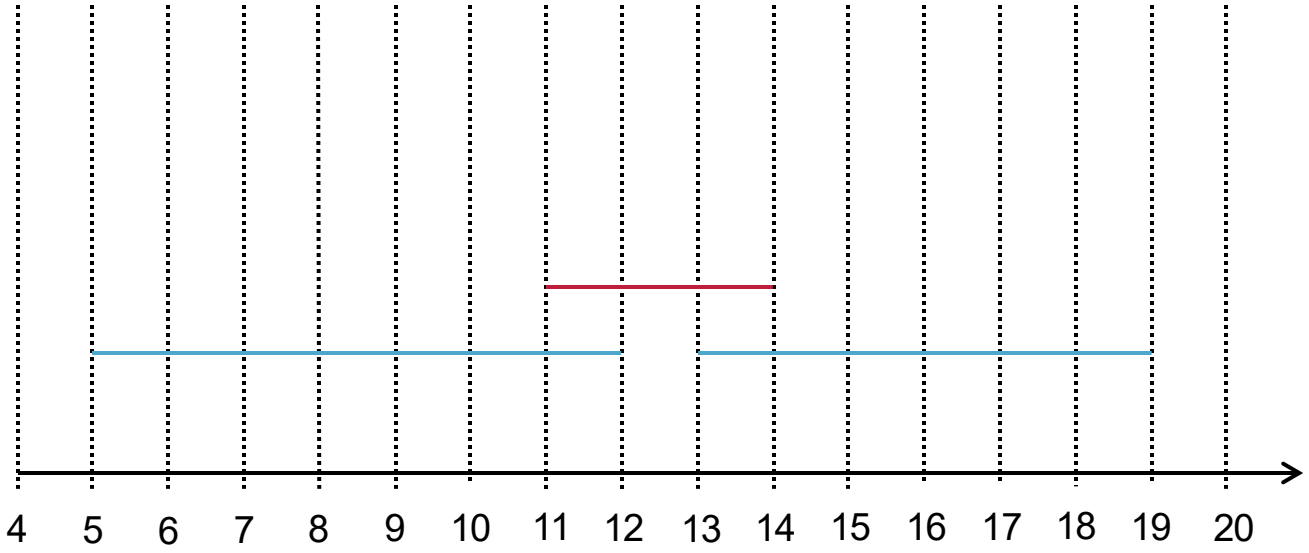
Längste zuerst?

Frühester Start?

Frühestes Ende?

Wenig Überlappungen?

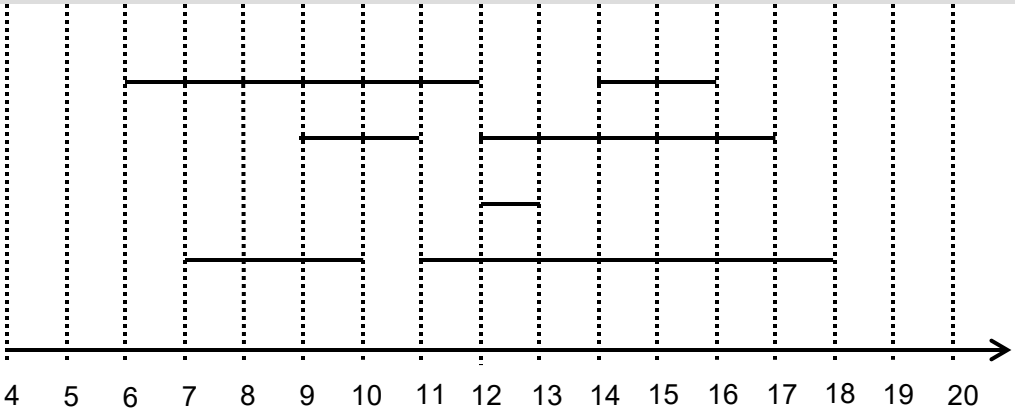
Hörsaal-Belegung – Kürzeste zuerst



ALG = 1

OPT = 2

Hörsaal-Belegung – Strategien



Welche Greedyfunktion?

Kürzeste zuerst?

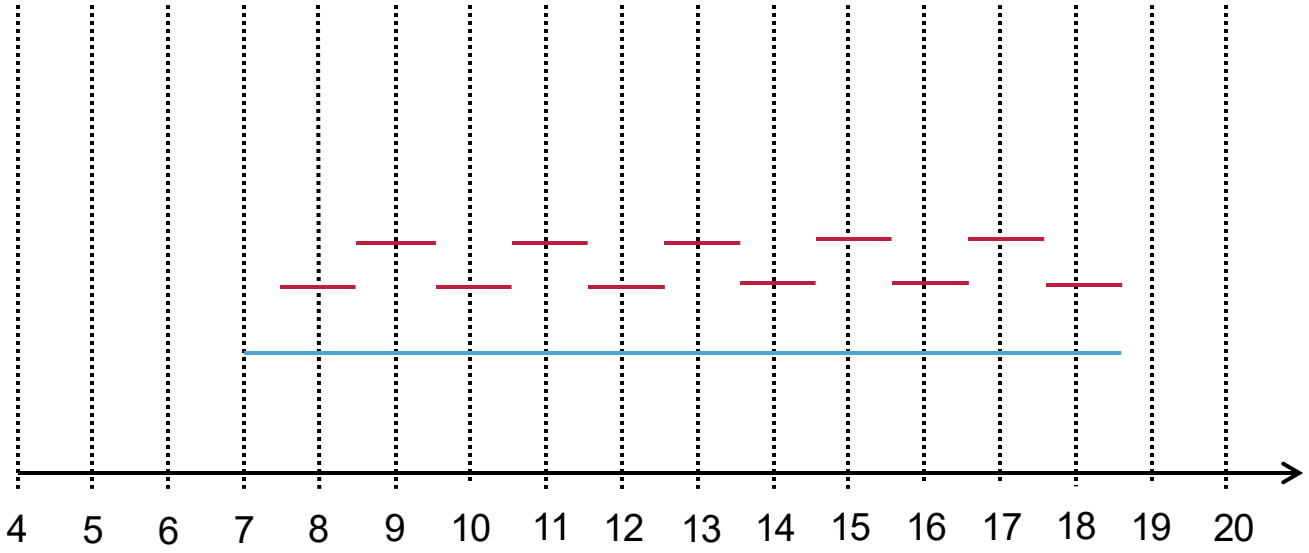
Längste zuerst?

Frühester Start?

Frühestes Ende?

Wenig Überlappungen?

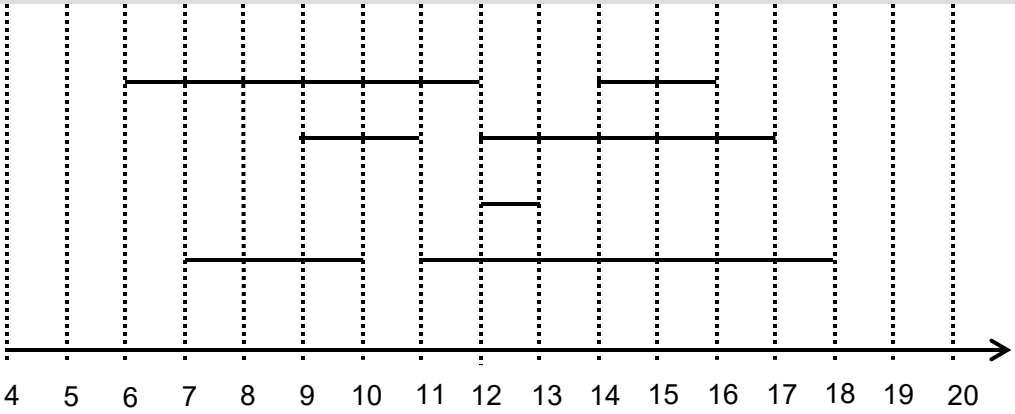
Hörsaal-Belegung – Frühester Start/Längstes Intervall



ALG = 1

OPT = 11

Hörsaal-Belegung – Strategien



Welche Greedyfunktion?

Kürzeste zuerst?

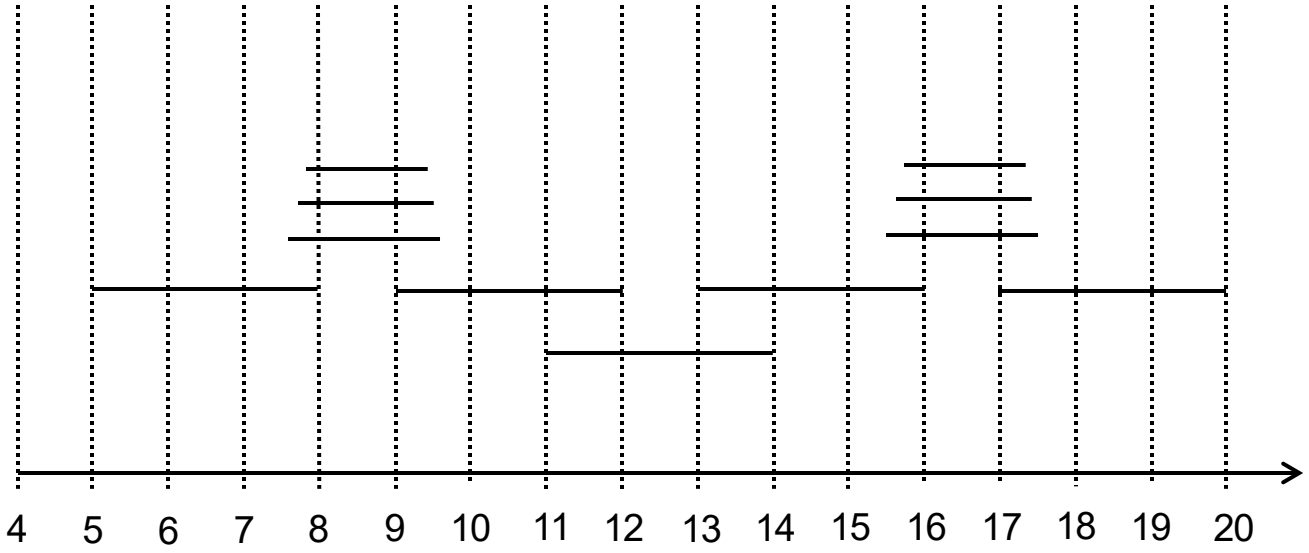
Längste zuerst?

Frühester Start?

Frühestes Ende?

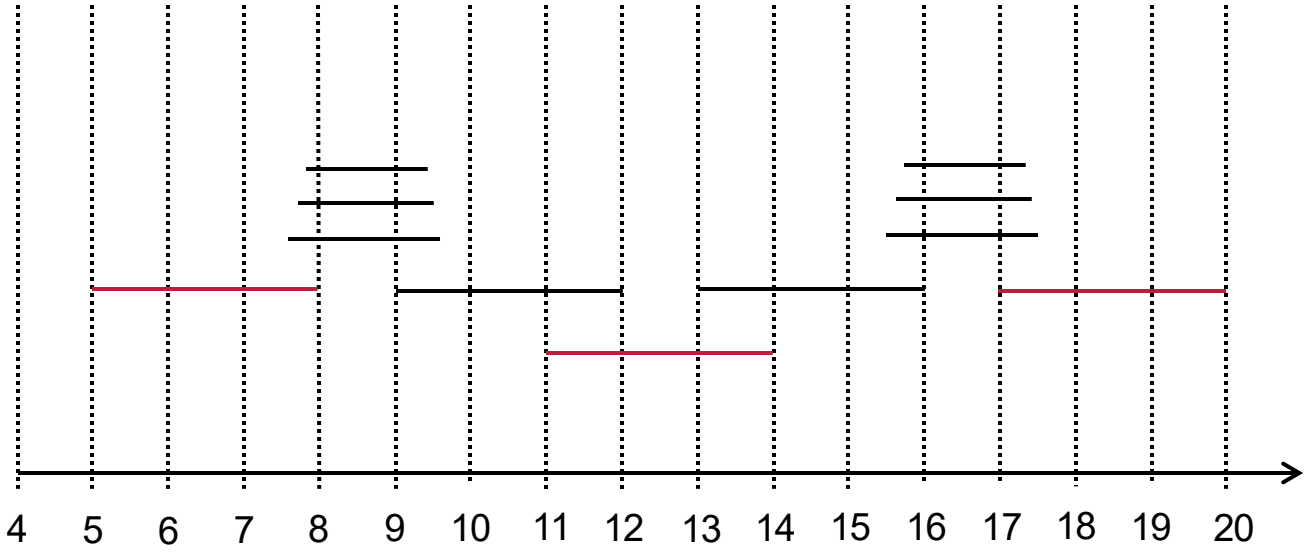
Wenig Überlappungen?

Hörsaal-Belegung – Wenig Überlappungen



ALG = 3

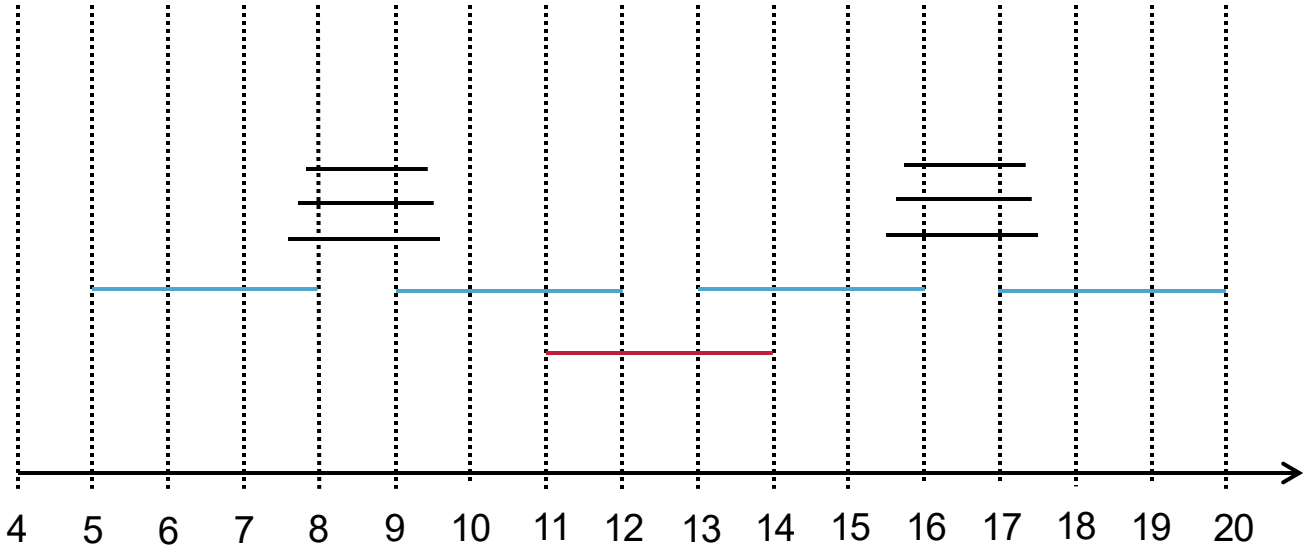
Hörsaal-Belegung – Wenig Überlappungen



ALG = 3

OPT = 4

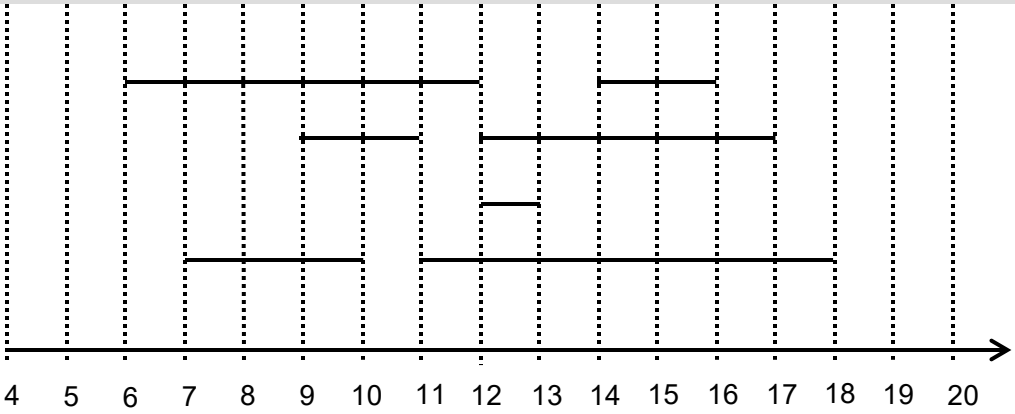
Hörsaal-Belegung – Wenig Überlappungen



ALG = 3

OPT = 4

Hörsaal-Belegung – Strategien



Welche Greedyfunktion?

Kürzeste zuerst?

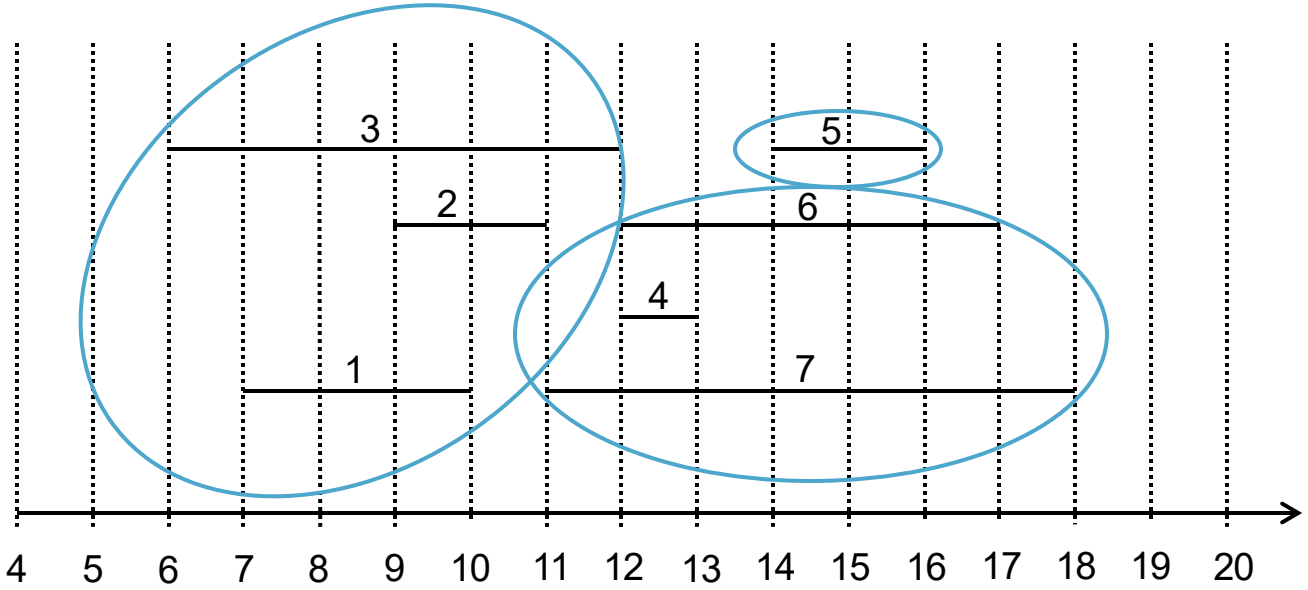
Längste zuerst?

Frühester Start?

Frühestes Ende?

Wenig Überlappungen?

Hörsaal-Belegung – Frühestes Ende



ALG = 3

OPT = 3?

Hörsaal-Belegung – Frühestes Ende

Greedy-Algorithmus: Frühestes Ende

1. Sortiere $\{1, \dots, n\}$ nach e_i aufsteigend; erhalte Permutation $\pi(1), \dots, \pi(n)$
2. $S \coloneqq \{\pi(1)\}$
3. $e \coloneqq e_{\pi(1)}$
4. **for** $k = 2$ **to** n **do**
5. **if** $(s_{\pi(k)} \geq e)$ **then**
6. $S \coloneqq S \cup \{\pi(k)\}$
7. $e \coloneqq e_{\pi(k)}$
8. **return** S

Theorem
Der Algorithmus löst das Hörsaal-Problem optimal in Zeit $O(n \log n)$

Beweisidee

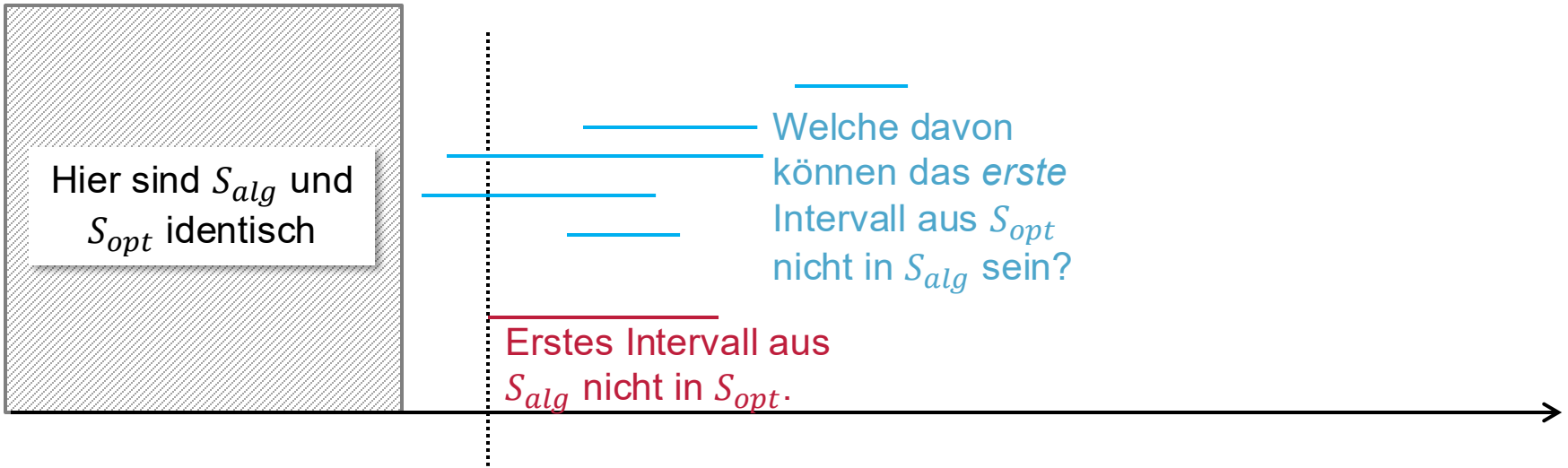
(Ein Klassiker):

- Vergleiche eine optimale Lösung mit der des Algorithmus.
- Wenn sie nicht gleich sind, dann gibt es einen erste Zeitpunkt (ein frühestes Intervall), ab dem die Lösungen sich unterscheiden.

→ Untersuche diesen Schritt genauer!

Beweis

Sei S_{alg} die Lösung von Algorithmus 1 und S_{opt} eine optimale Lösung.

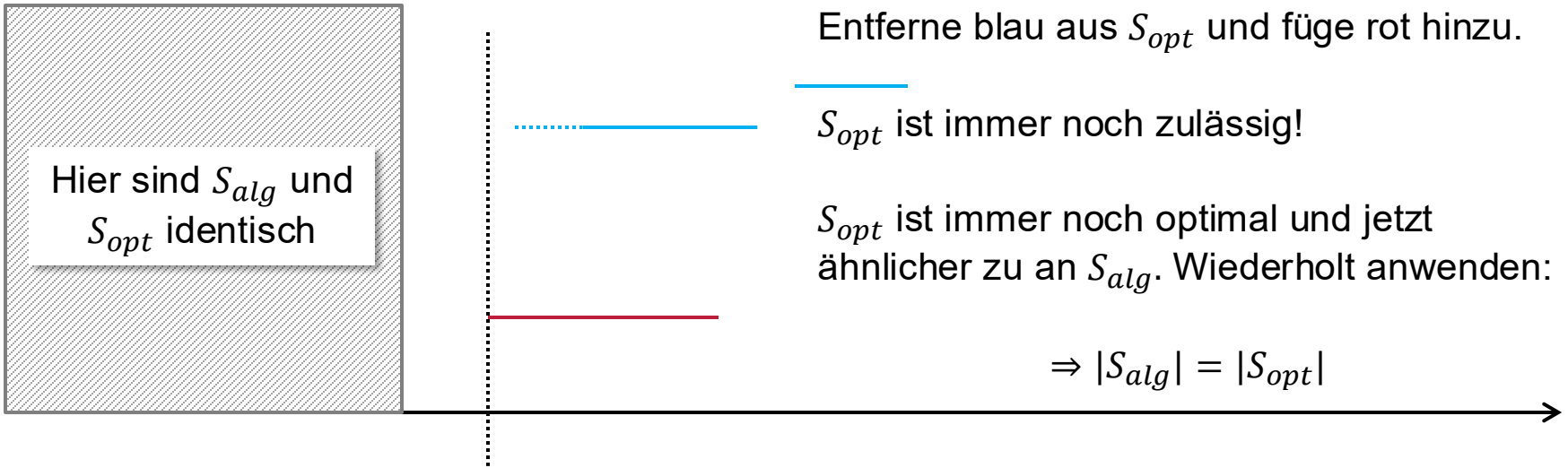


Beweis

Sei S_{alg} die Lösung von Algorithmus 1 und S_{opt} eine optimale Lösung.

Beweis

Sei S_{alg} die Lösung von Algorithmus 1 und S_{opt} eine optimale Lösung.

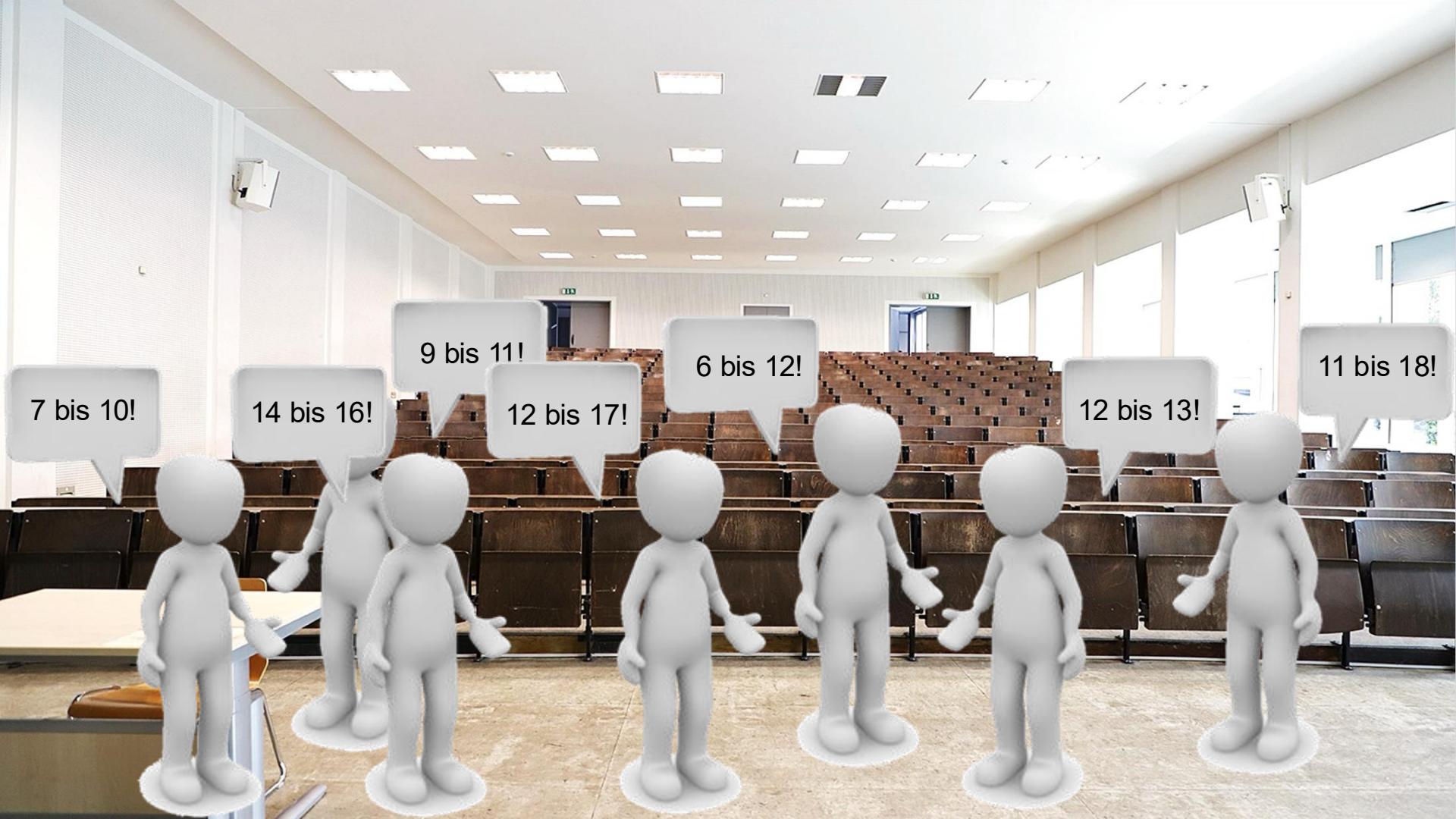


Entferne blau aus S_{opt} und füge rot hinzu.

S_{opt} ist immer noch zulässig!

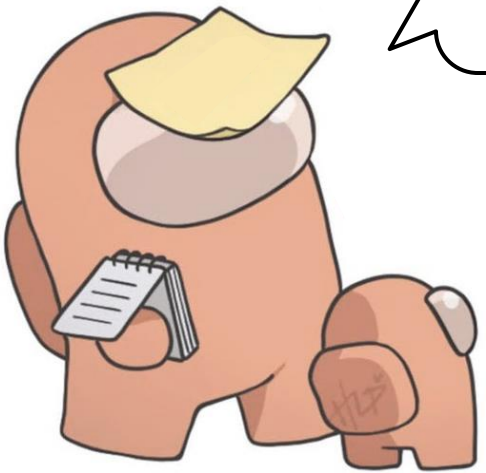
S_{opt} ist immer noch optimal und jetzt ähnlicher zu an S_{alg}. Wiederholt anwenden:

$$\Rightarrow |S_{alg}| = |S_{opt}|$$


7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!

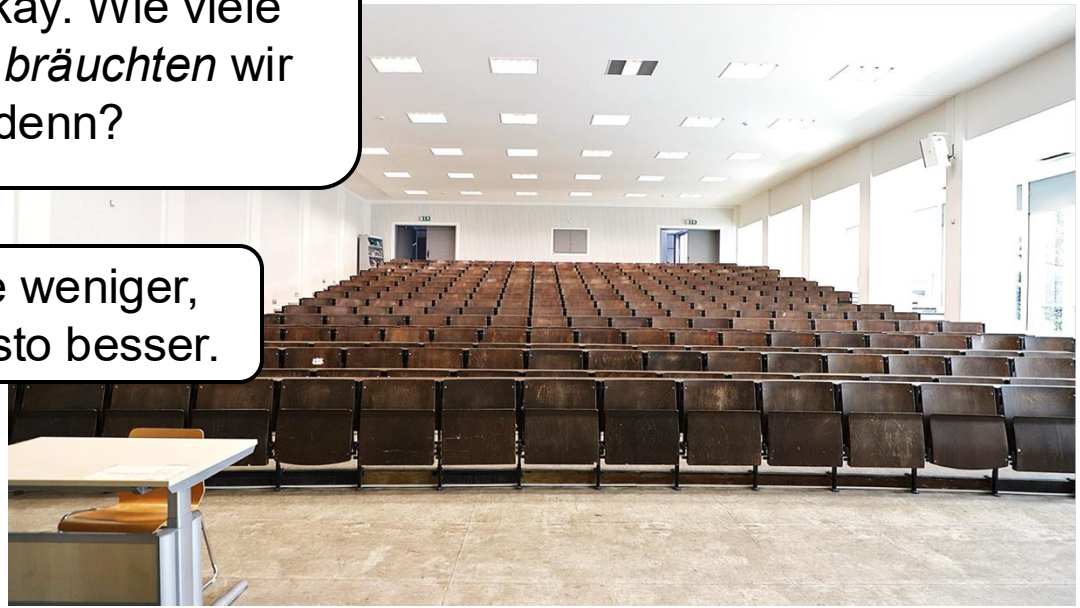

…und was ist mit uns?
9 bis 11!
6 bis 12!
11 bis 18!
7 bis 10!
14 bis 16!
12 bis 17!
Problem gelöst!

Raumplanung



Okay okay. Wie viele Hörsäle *bräuchten* wir denn?

Je weniger, desto besser.



Hörsaal-Belegung – Eine Variante

Gegeben

Menge von Intervallen $\mathcal{I} = \{I_1 = [s_1, e_1), \dots, I_n = [s_n, e_n)\}$

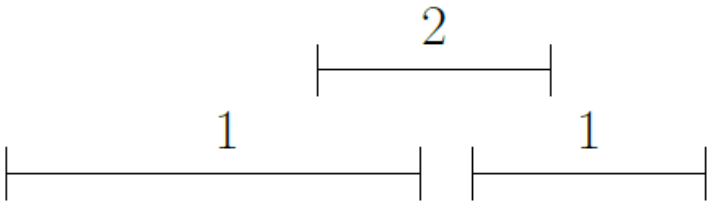
Können jedem Intervall (Veranstaltung) k verschiedene Werte (Räume) zuordnen mit f

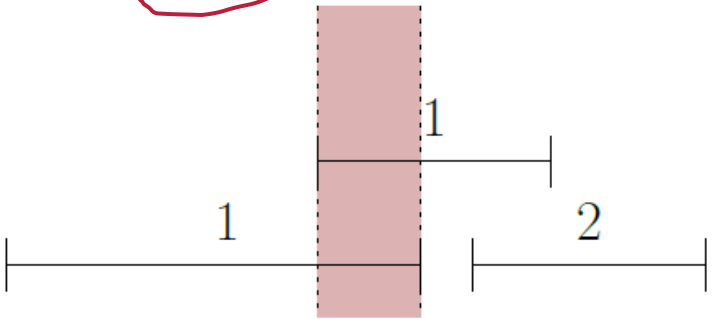
Minimiere k!

Gesucht

Die kleinste Zahl $k \in \mathbb{N}$, sodass eine Funktion $f: I \to \{1, \dots, k\}$ existiert und für je zwei Intervalle I und I' gilt $I \cap I' \neq \emptyset \Rightarrow f(I) \neq f(I')$

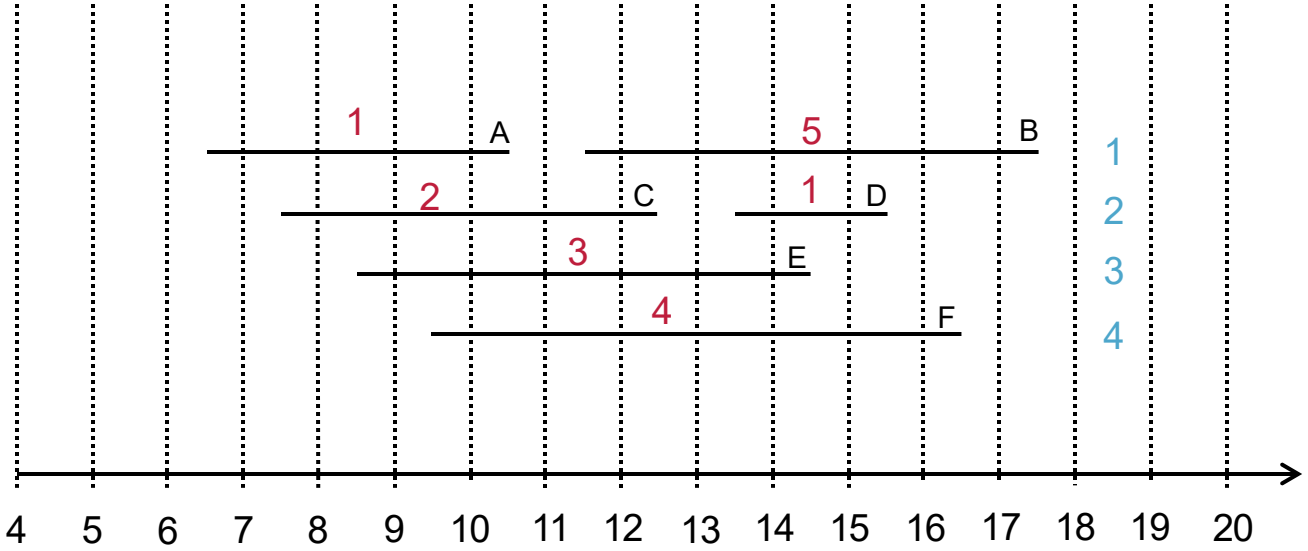
Wenn zwei Intervalle überlappen, haben sie verschiedenen Wert in f (anderen Raum)





Ideen?

Hörsaal-Belegung – Eine Variante

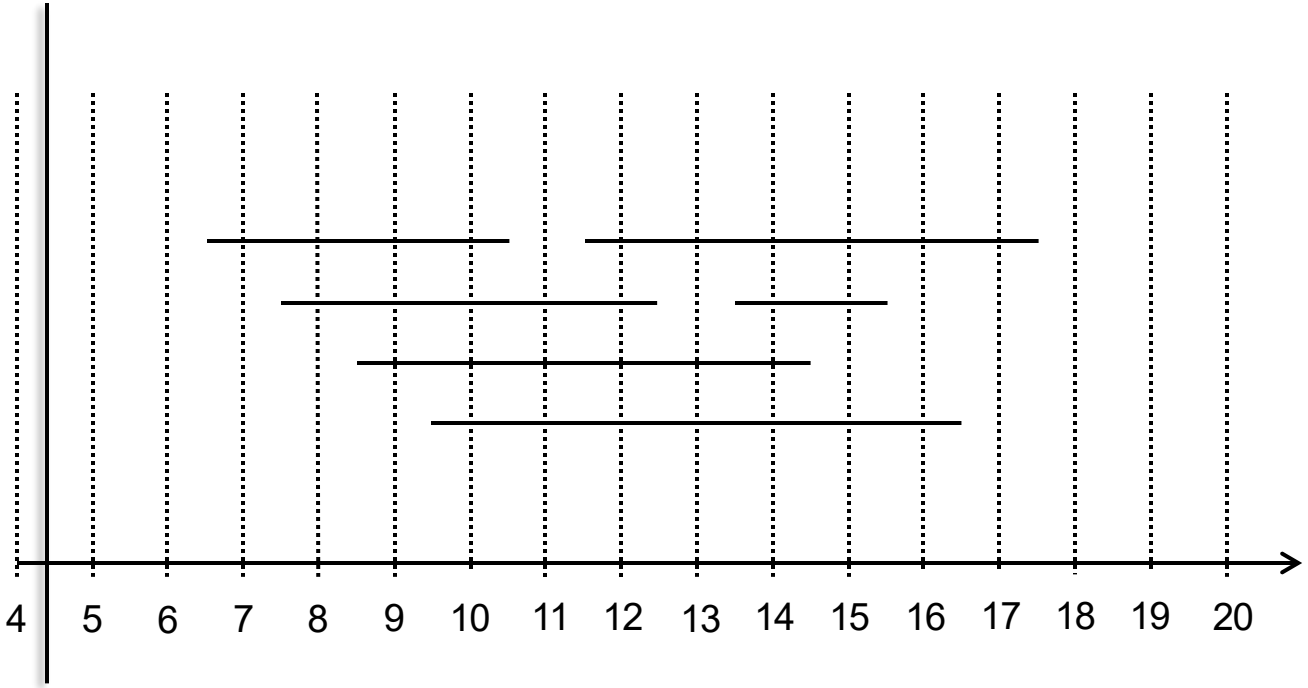


ALG = 5

OPT = 4

Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

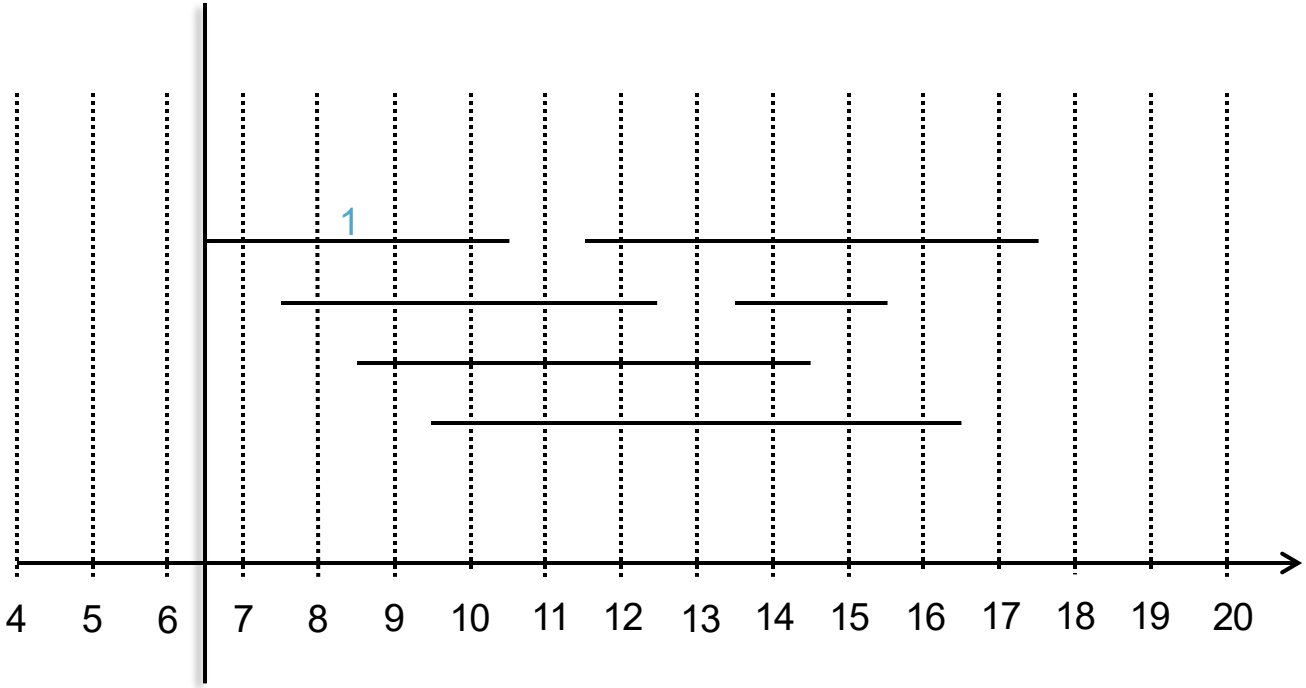
Hörsaal-Belegung – Eine Variante



Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Idee:
Bewege eine Linie von links nach rechts

Hörsaal-Belegung – Eine Variante

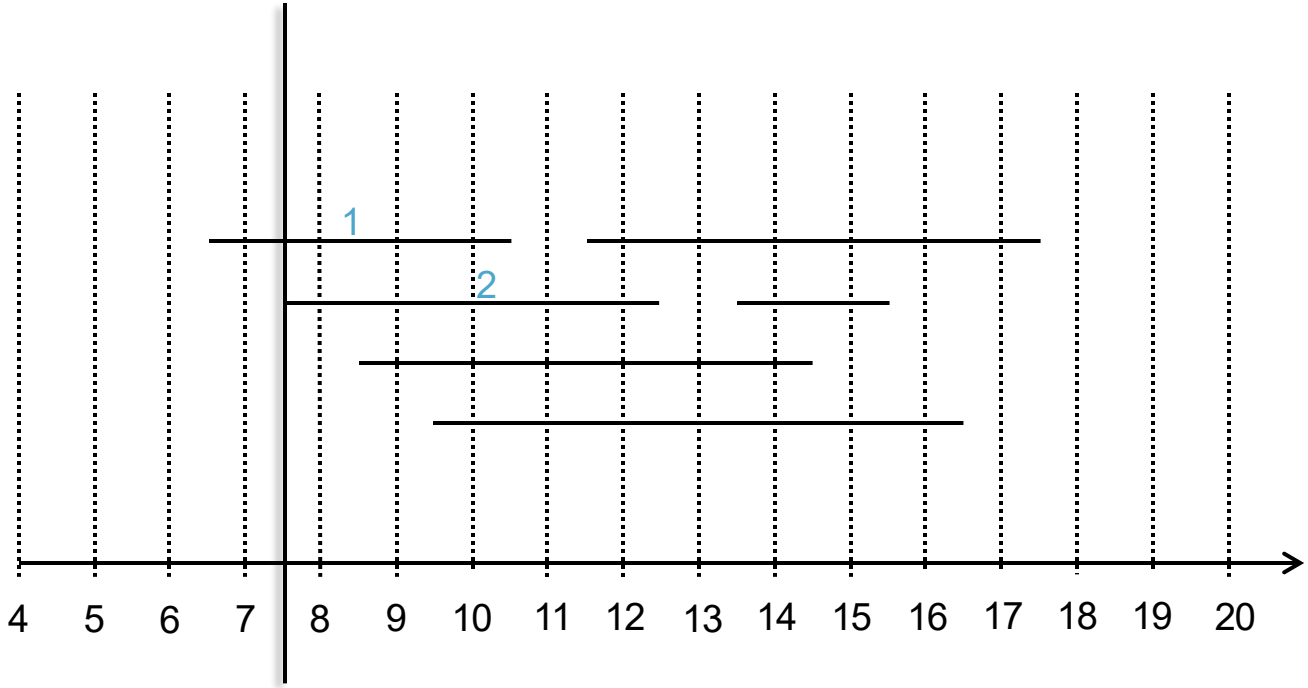


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

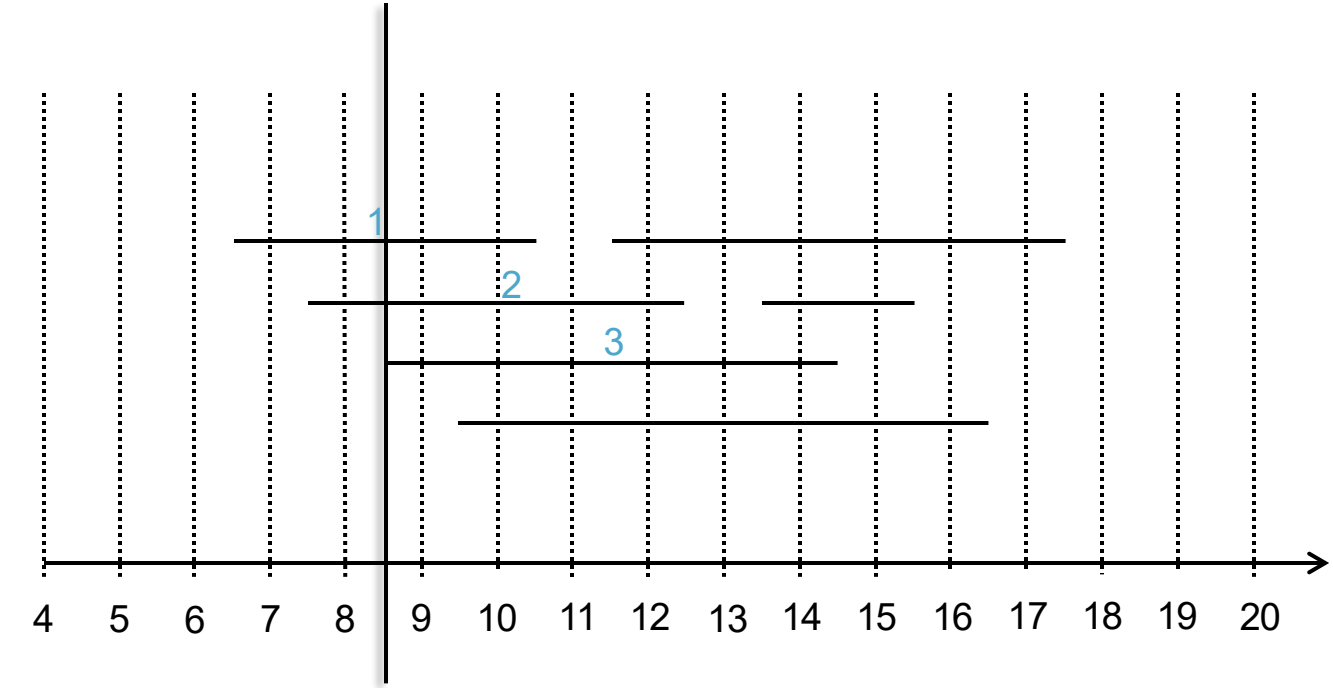


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

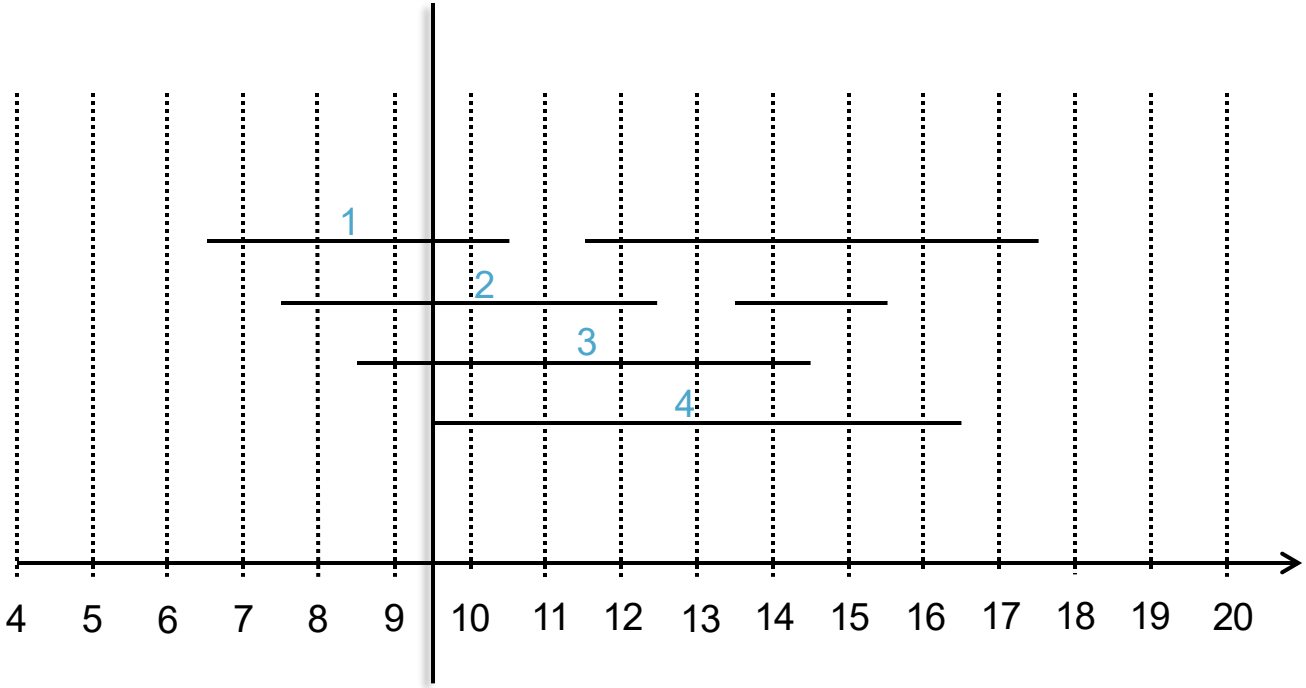


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

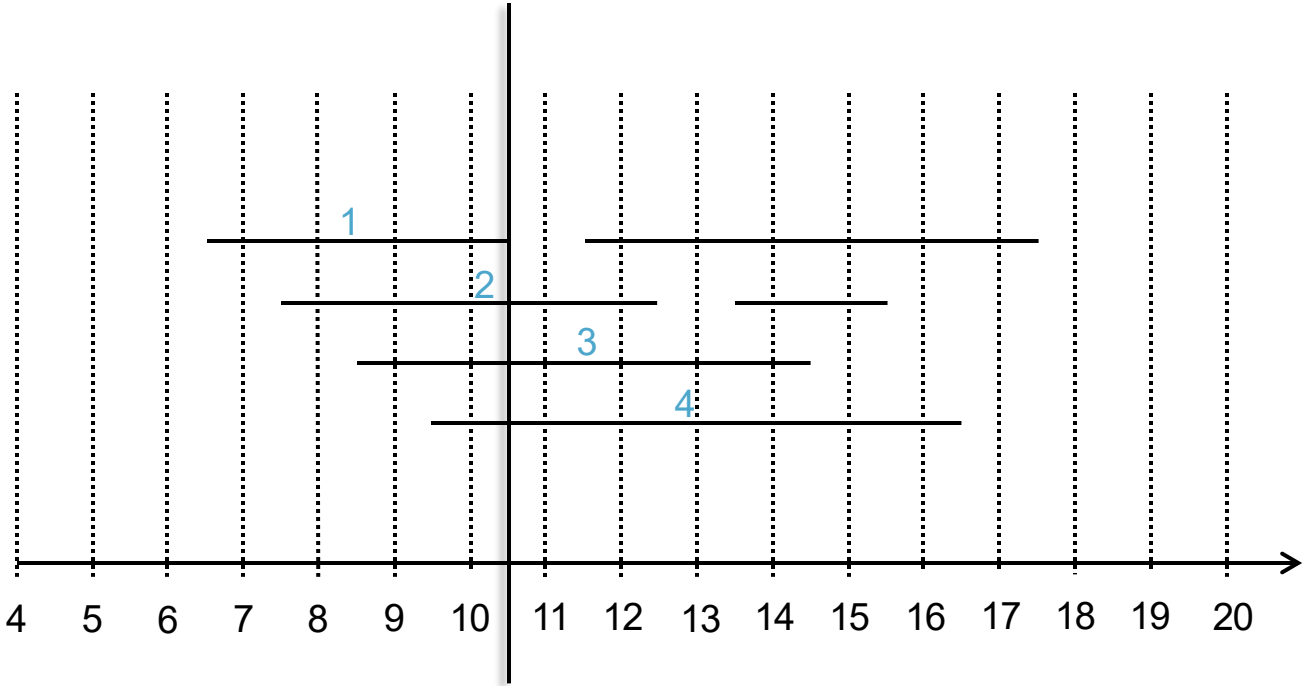


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

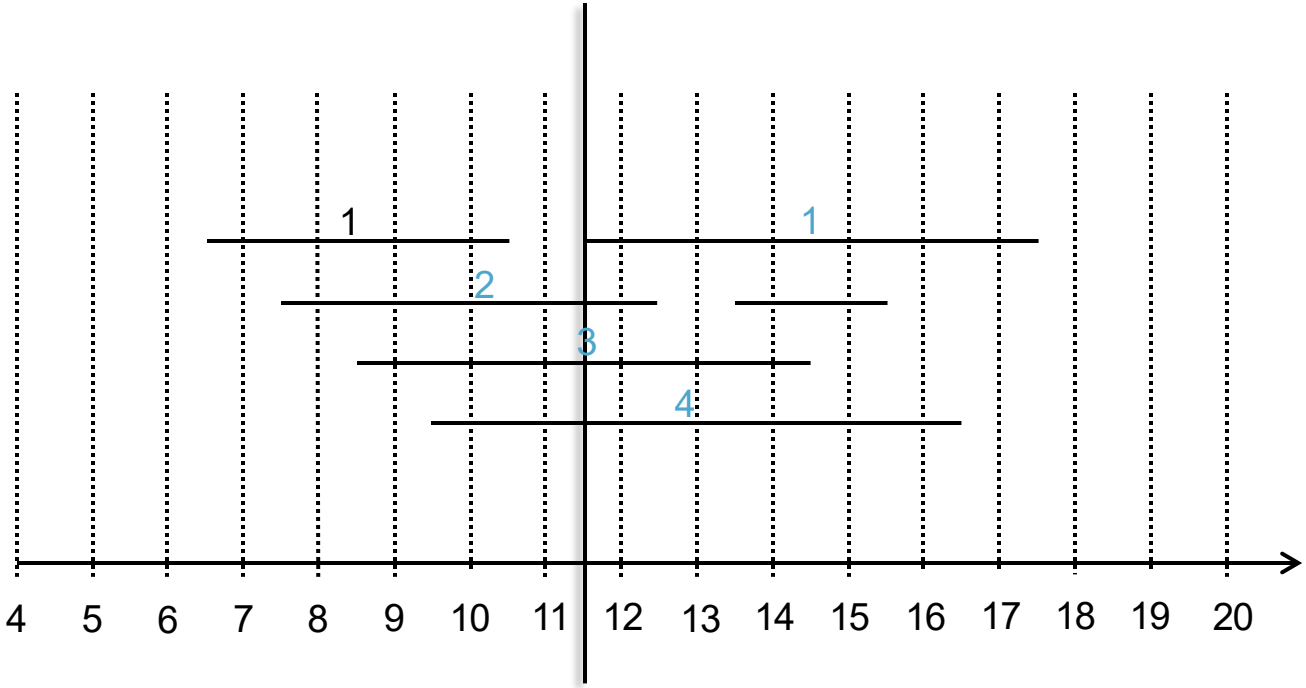


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

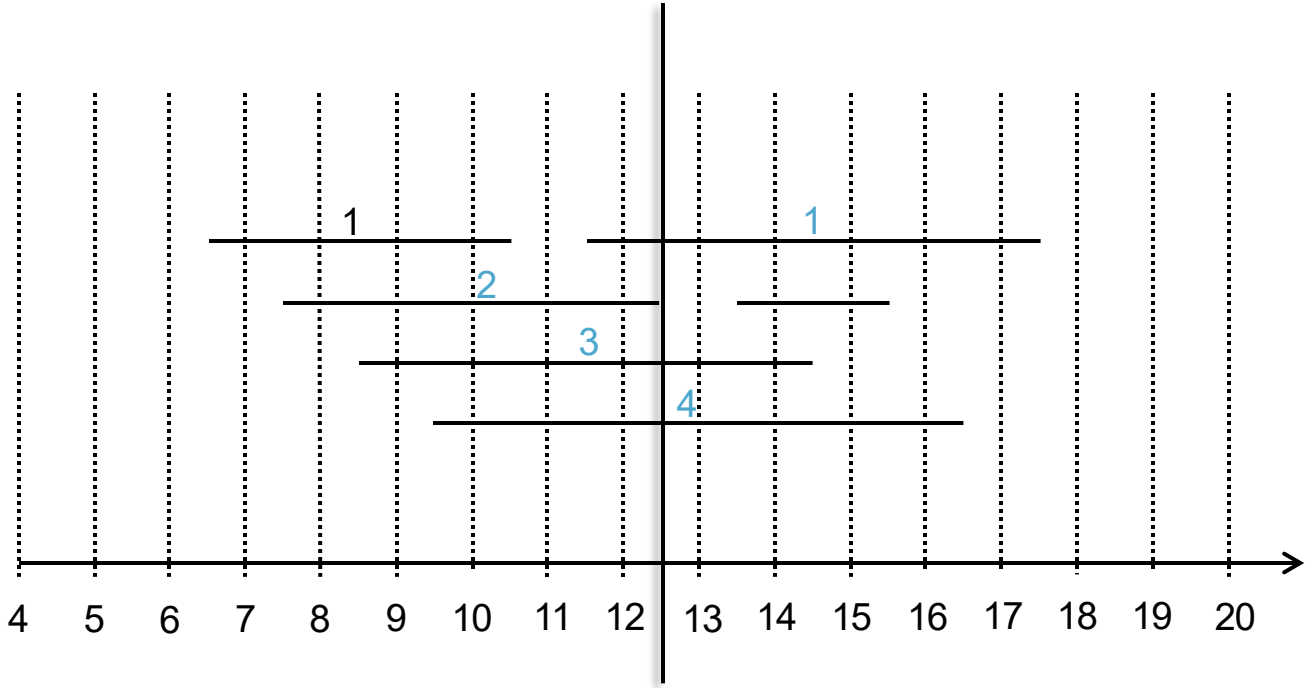


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

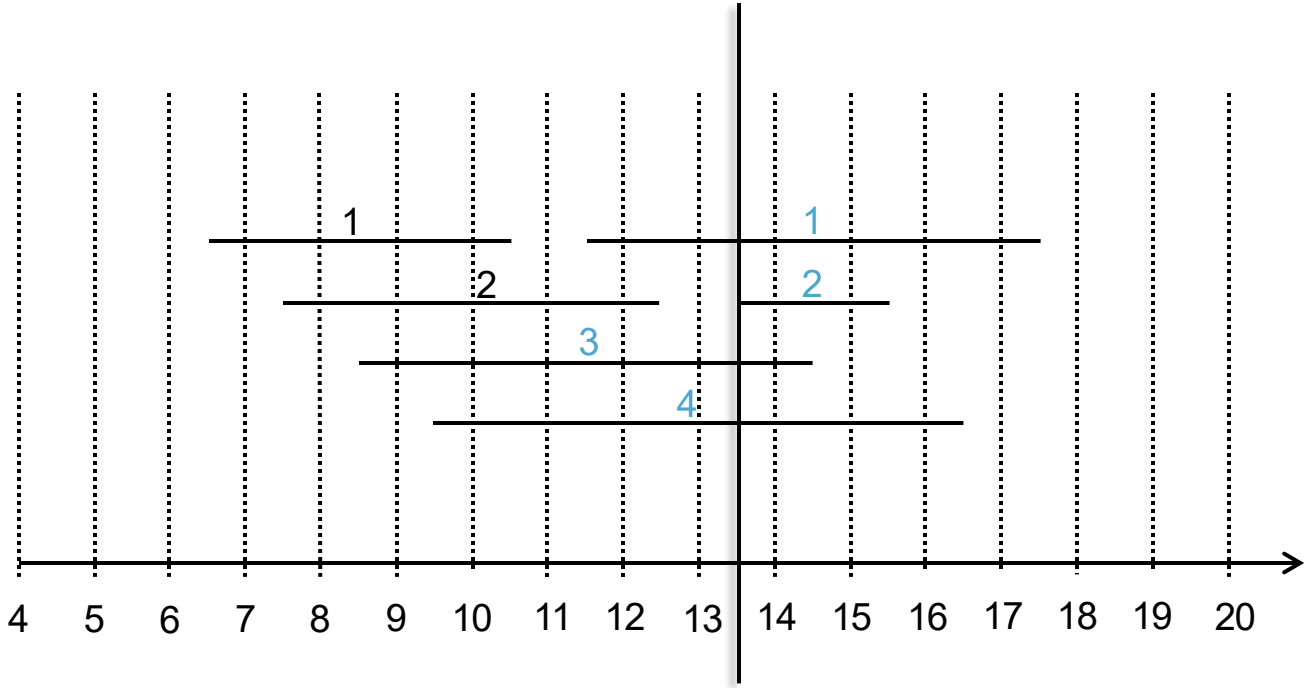


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

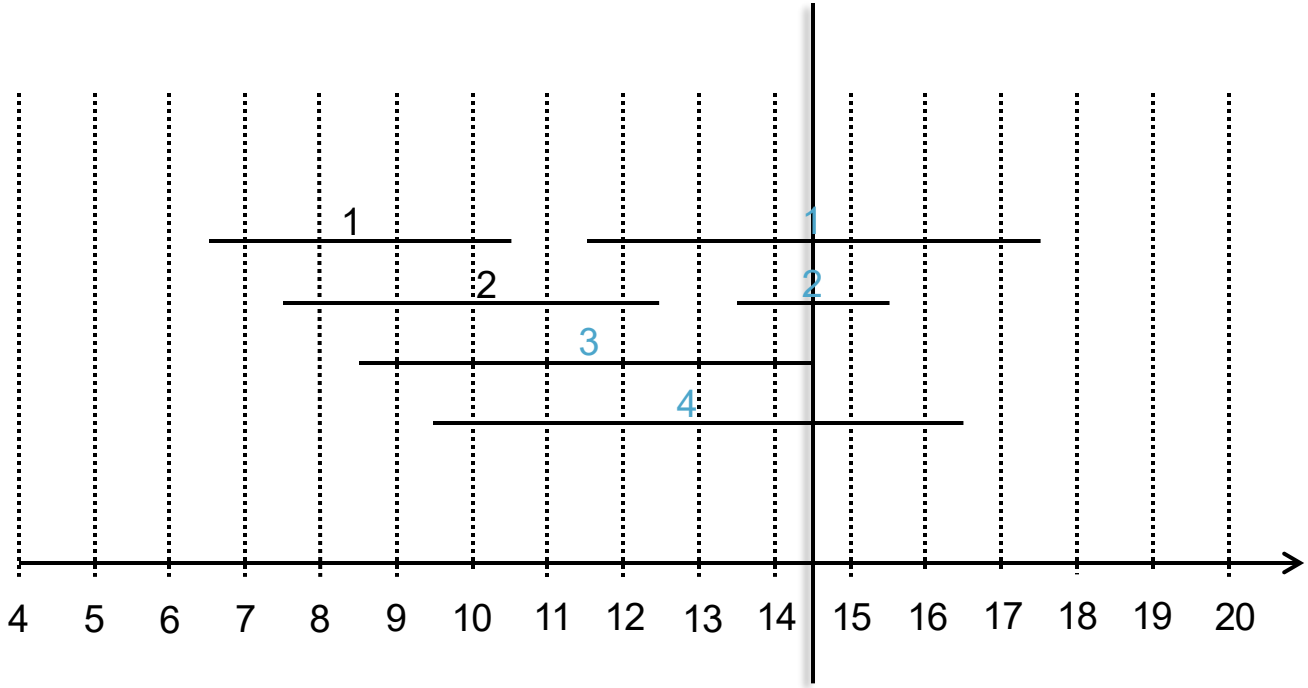


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

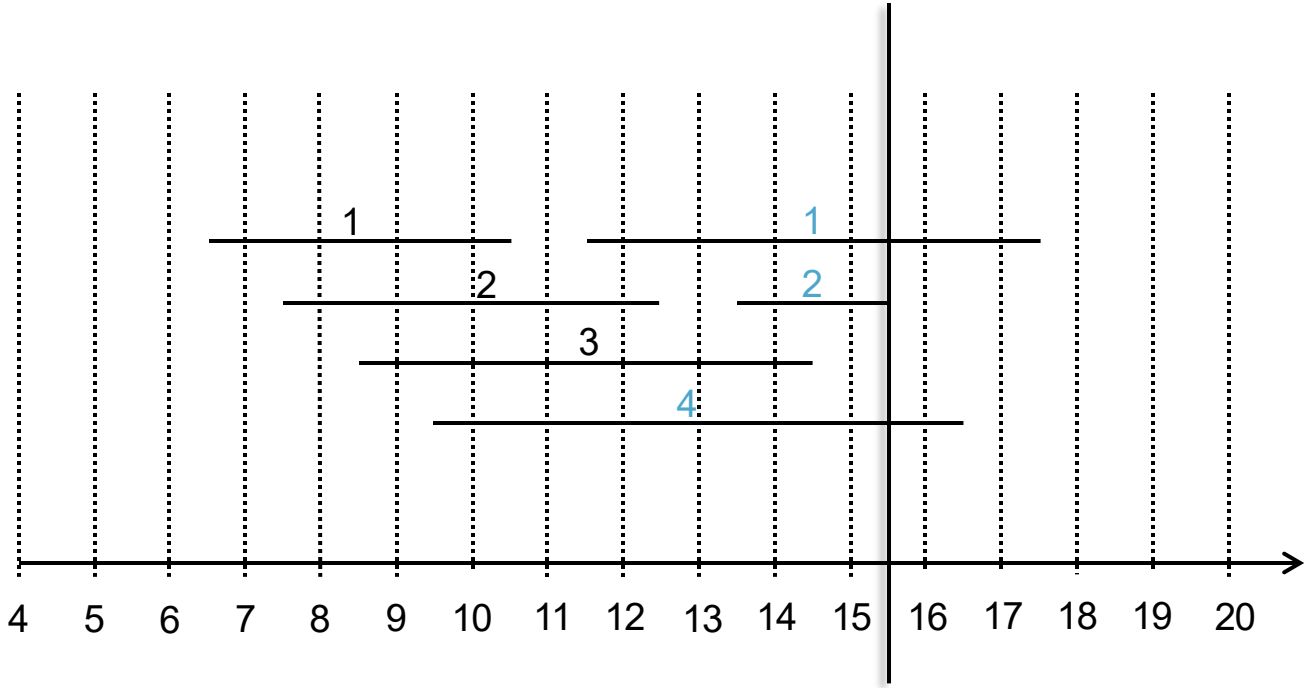


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

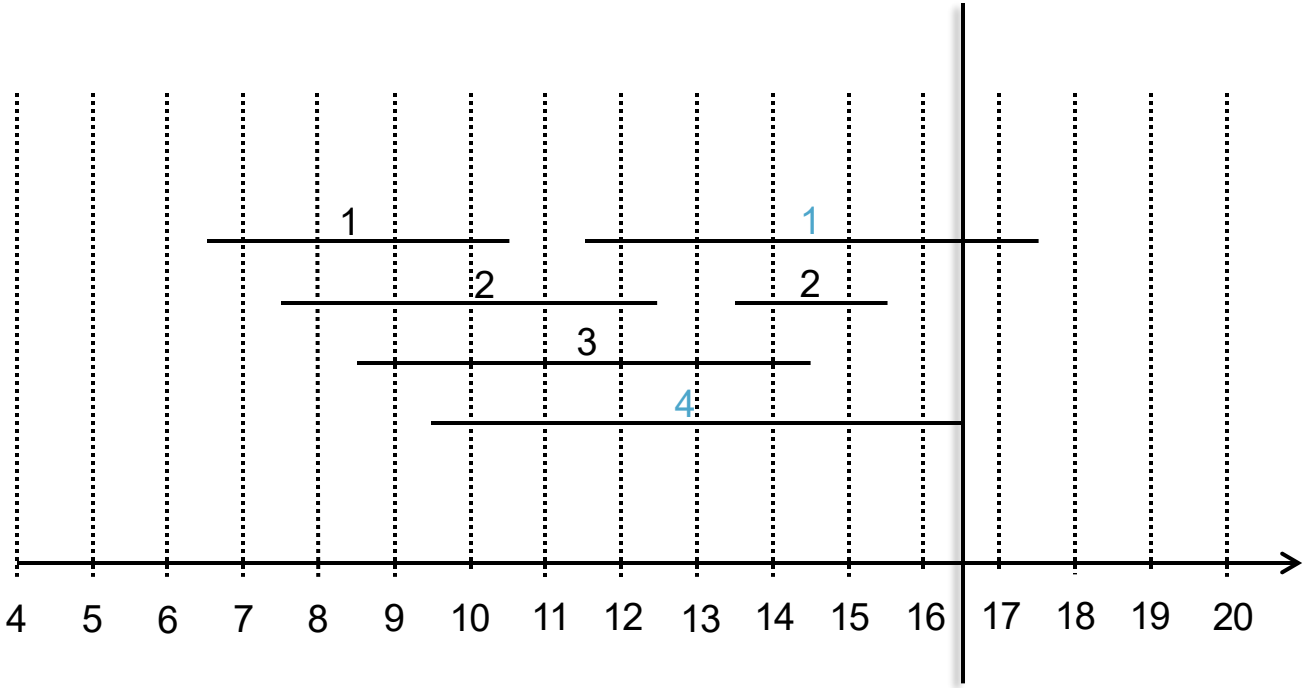


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

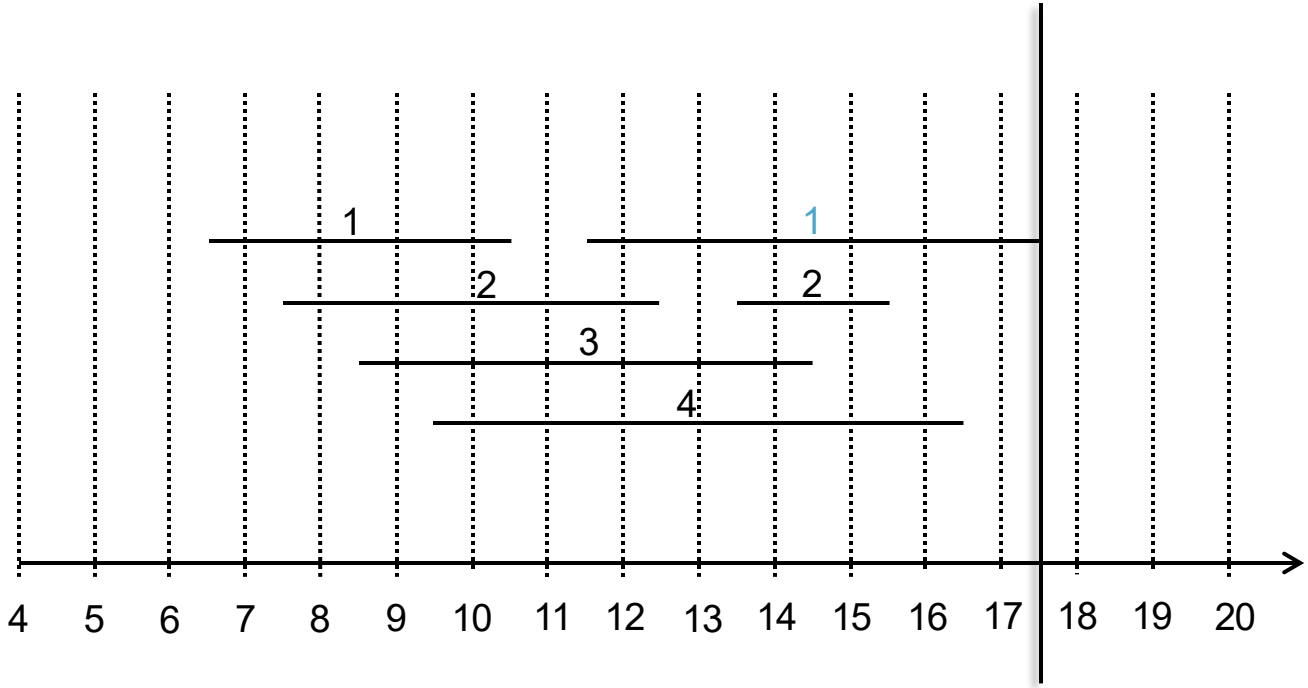


Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante



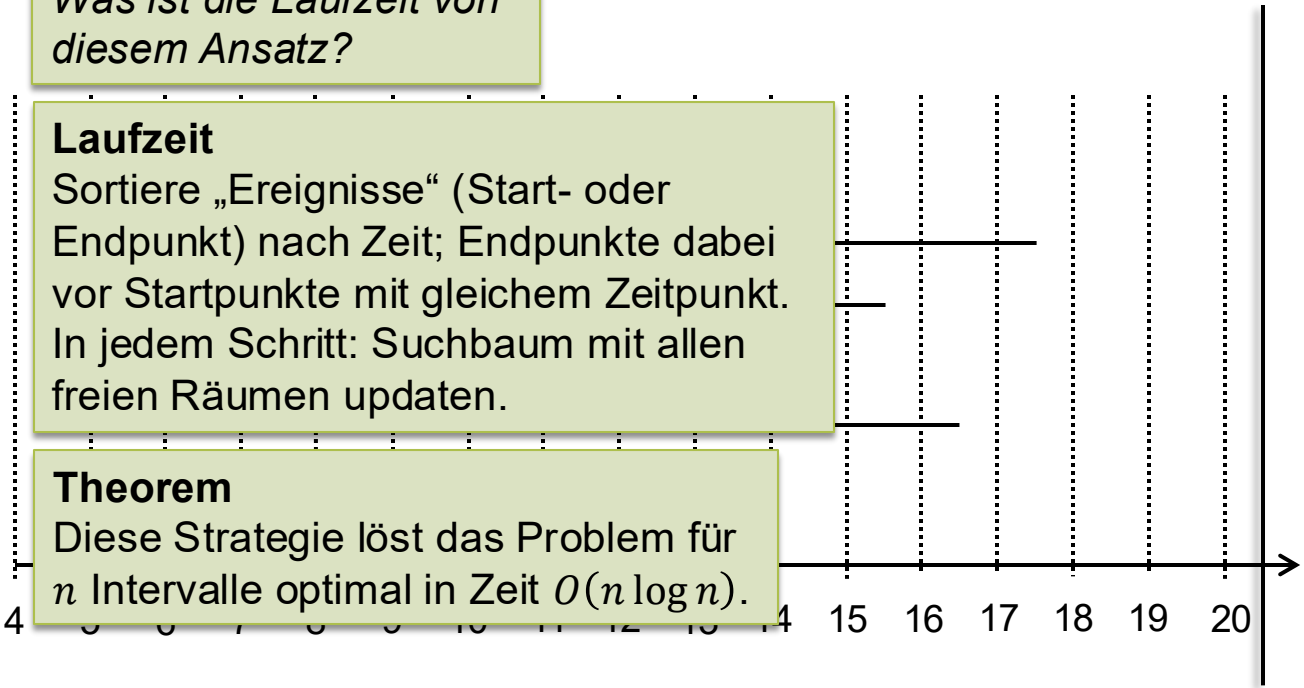
Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines

- **Startpunktes:**
  Weise kleinste Raumnummer zu
- **Endpunktes:**
  Gib Raum frei

Hörsaal-Belegung – Eine Variante

Was ist die Laufzeit von diesem Ansatz?



Laufzeit
Sortiere „Ereignisse“ (Start- oder Endpunkt) nach Zeit; Endpunkte dabei vor Startpunkte mit gleichem Zeitpunkt. In jedem Schritt: Suchbaum mit allen freien Räumen updaten.

Theorem
Diese Strategie löst das Problem für n Intervalle optimal in Zeit $O(n \log n)$.

Wichtig ist nicht nur das Ende, sondern auch der Start eines Intervalls!

Bei Erreichen eines
- **Startpunktes:** Weise kleinste Raumnummer zu
- **Endpunktes:** Gib Raum frei

Beweis

Wie viele Räume braucht man mindestens?

Sei

- $\chi \coloneqq \max_i(|\{I \in \mathcal{I} : i \in I\}|)$,
- opt der Wert einer optimalen Lösung,
- alg der Wert der Lösung unserer Strategie

Maximale Anzahl an Intervallen, die sich gleichzeitig überlappen.

Beobachtung: $opt \geq \chi$

$\Rightarrow alg = \chi$

Wir zeigen: $alg \leq \chi$

Beweis

Angenommen, wir brauchen k Räume, d.h. $alg = k$.

Betrachte den ersten Zeitpunkt, zu dem wir Raum k benutzen.

Das passiert nur beim Startpunkt eines neuen Intervalls I.

Was muss noch gelten?

- Alle anderen Räume $1, \dots, k-1$ sind in Gebrauch!
- D.h. I überlappt $k-1$ andere Intervalle an seinem Startpunkt.
- Endpunkte vor Startpunkten zum gleichen Zeitpunkt bearbeitet: echte Überlappung!

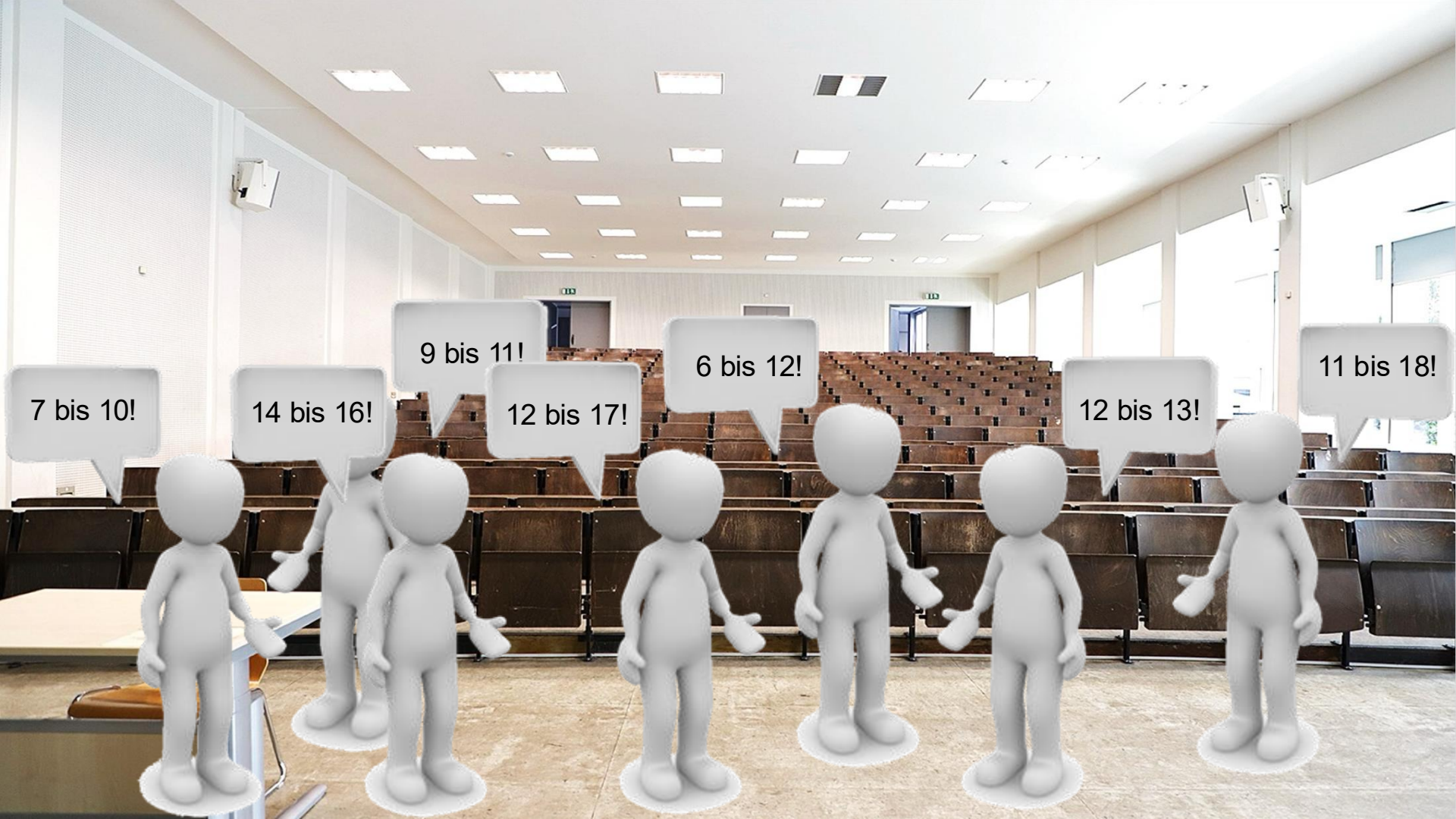
→ Wenn $alg = k$, dann überlappen auch (mindestens) k Intervalle.

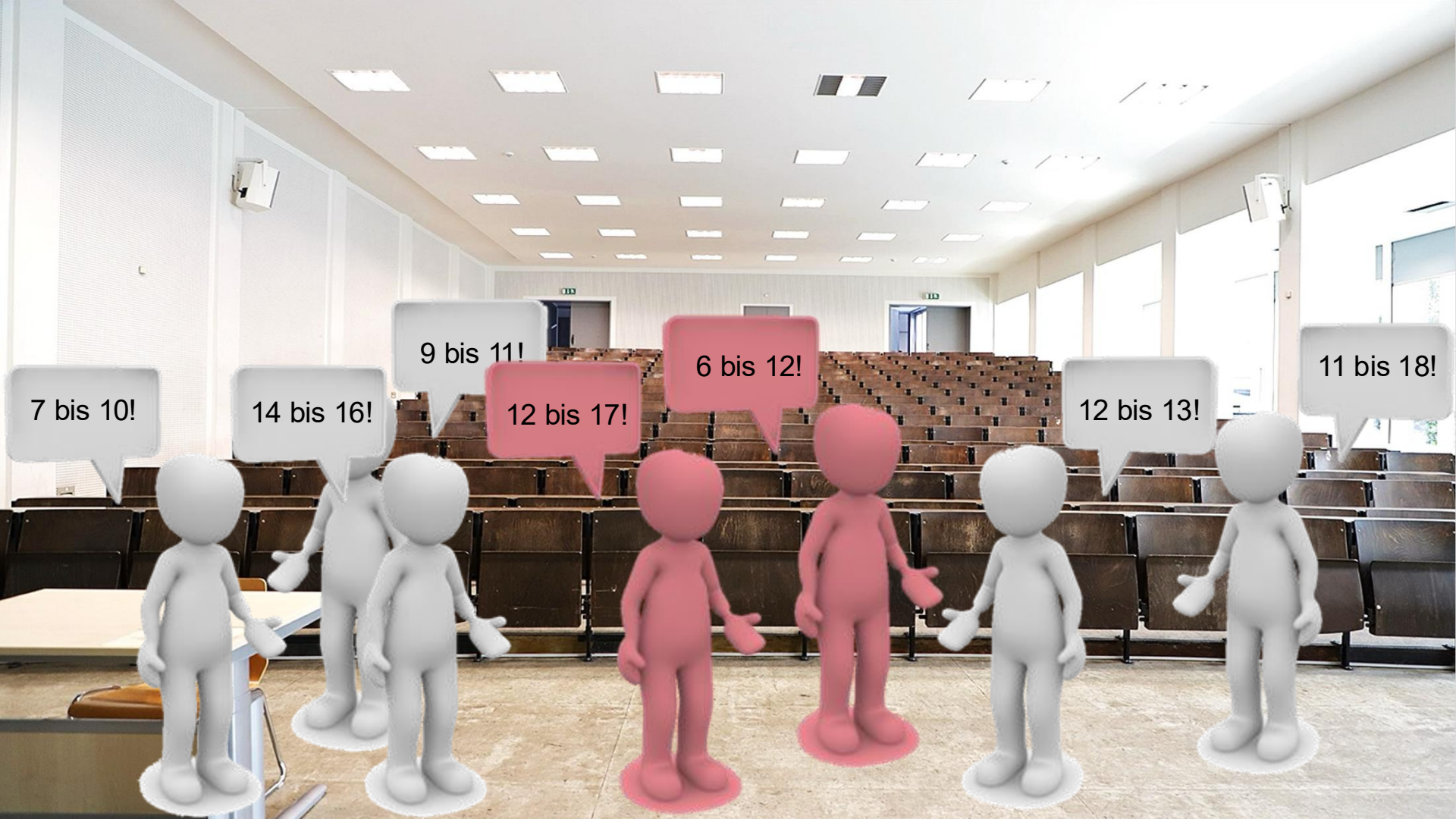
→ $alg \leq \chi$

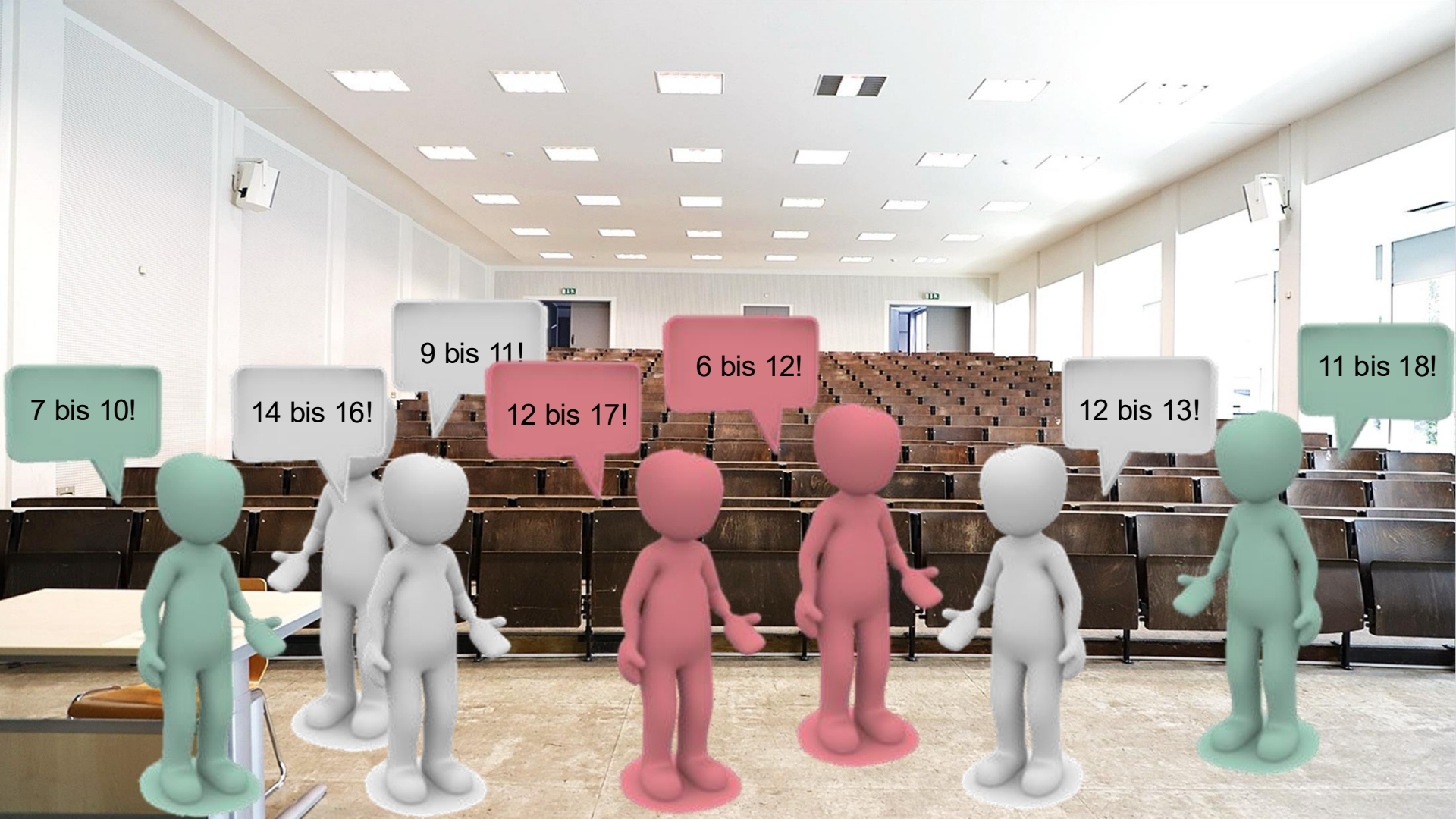
… Sweep-Line Algorithmen

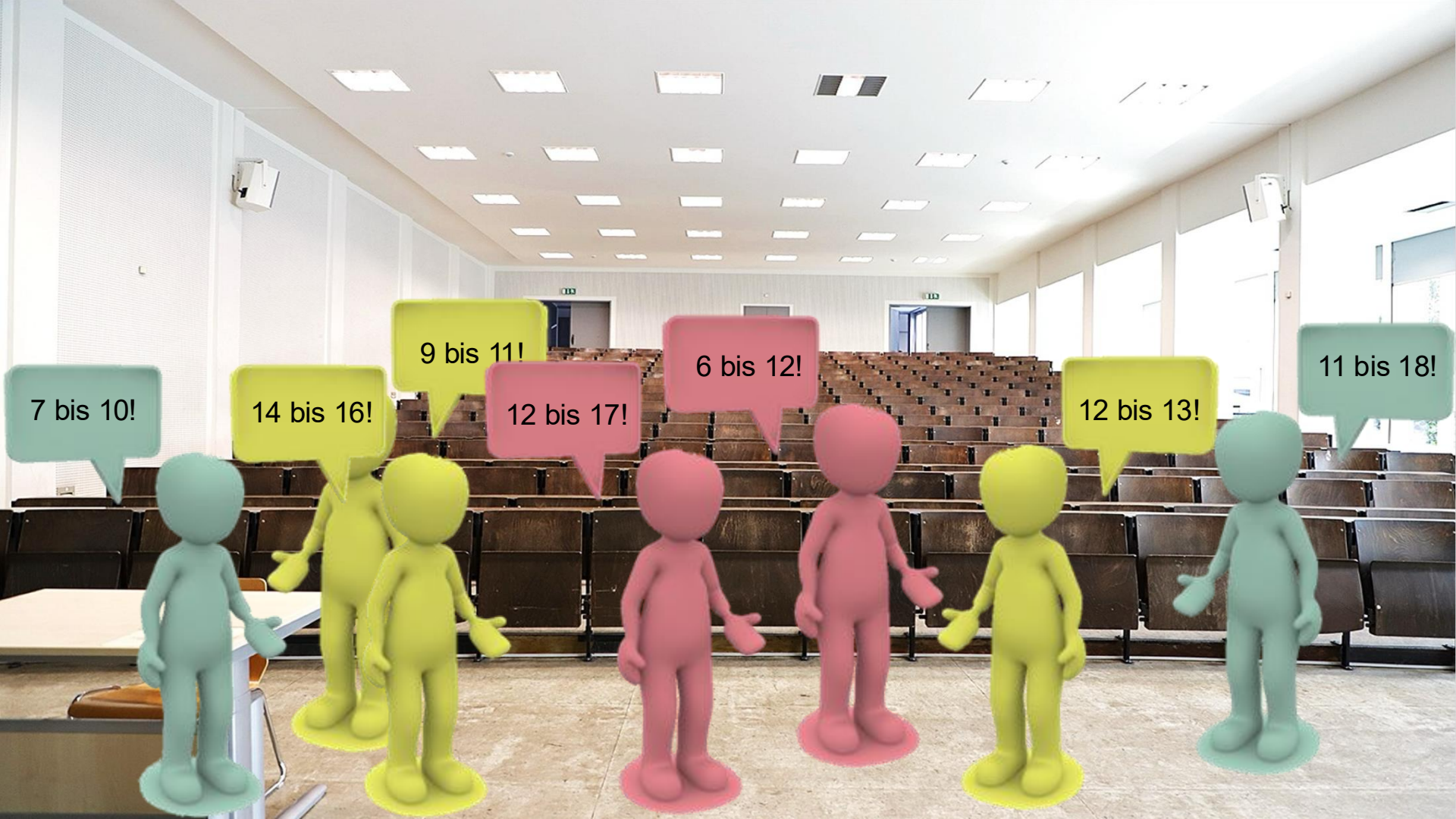
… ein weitere, typische algorithmische Idee:

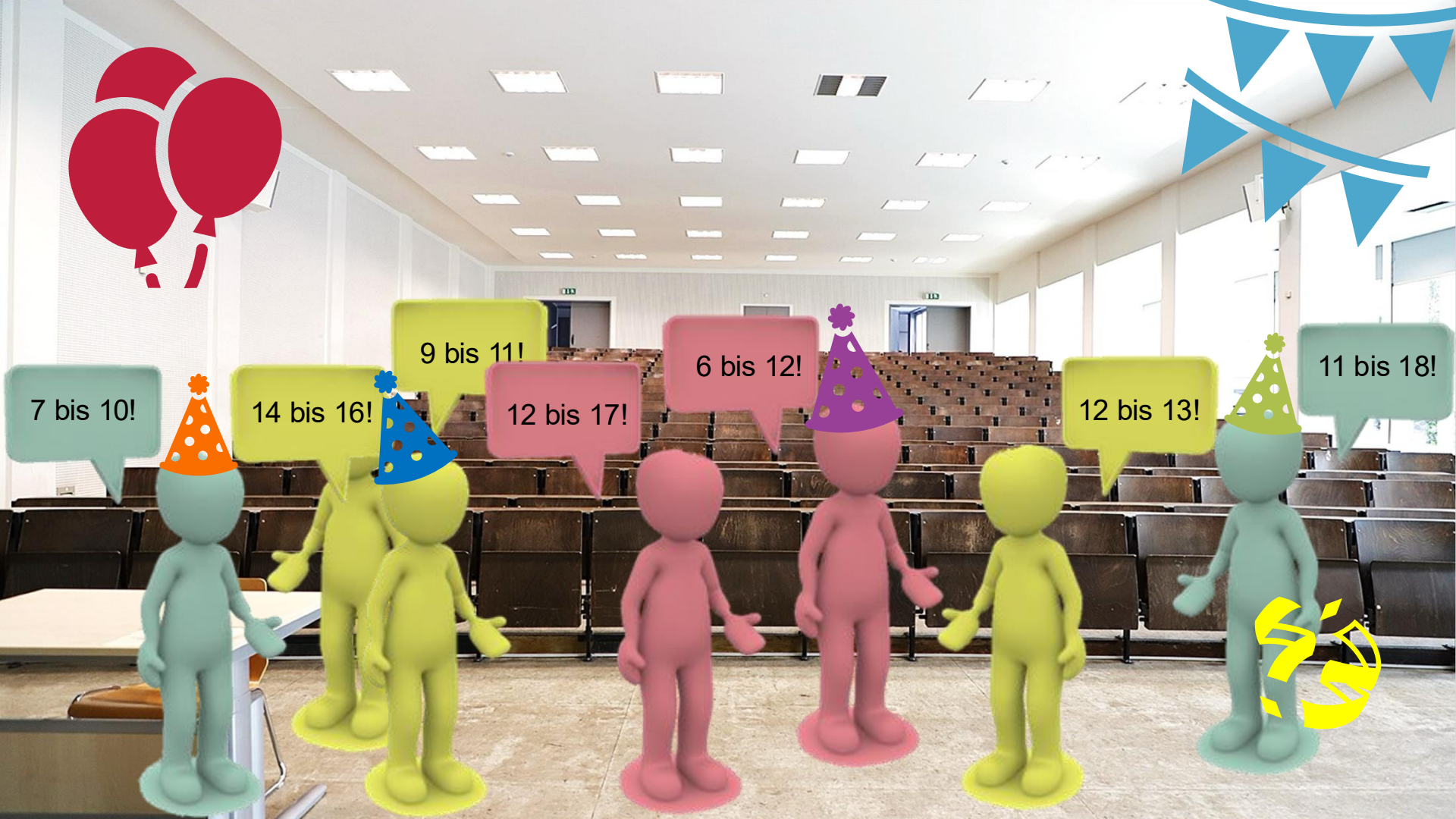
- Gehe sortierte Eventpunkte ab
  (also lasse gedanklich eine Linie über die Instanz fahren)
  - triff darauf basierend Entscheidungen.
  - Manchmal entstehen auch noch dynamisch neue Eventpunkte oder welche verschwinden ...


7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!

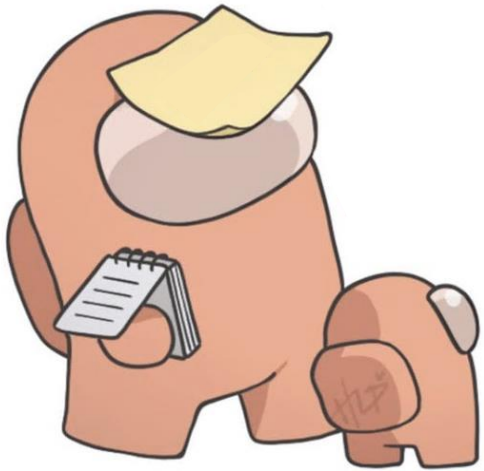

7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!

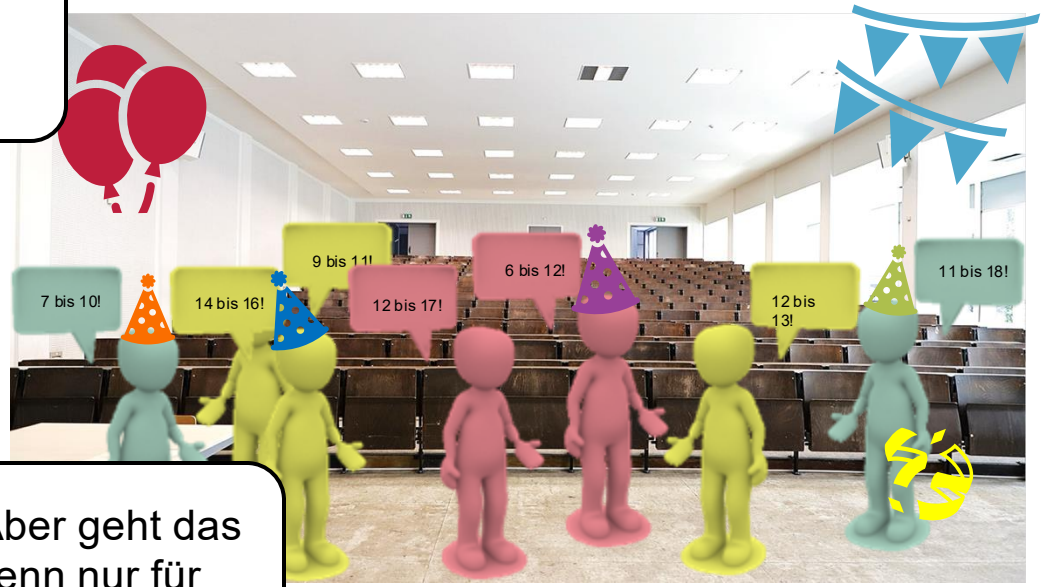

7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!


7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!


7 bis 10!
14 bis 16!
9 bis 11!
12 bis 17!
6 bis 12!
12 bis 13!
11 bis 18!

Endlich Feierabend!



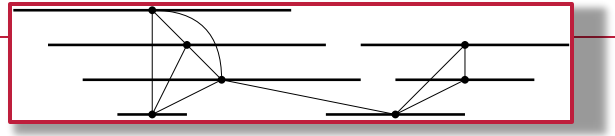


... Aber geht das denn nur für Intervalle?

Verallgemeinerung

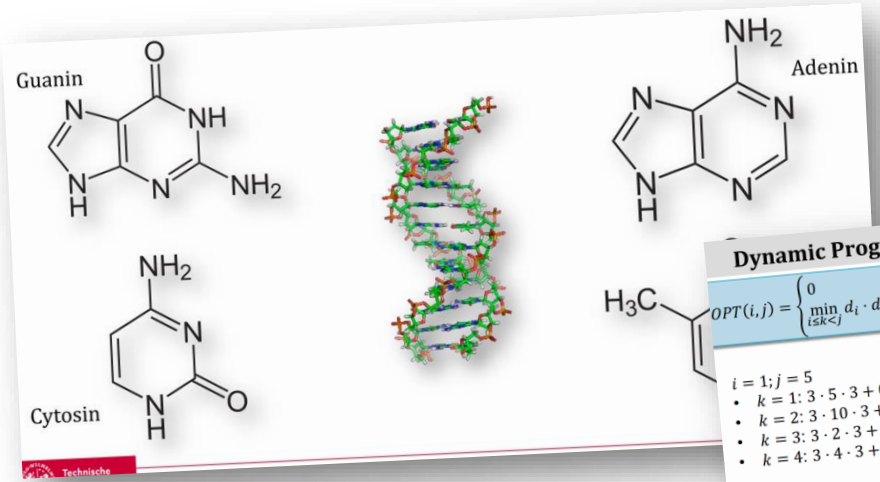
Hier vorgestellte Probleme sind Spezialfälle von berühmten Graphproblemen:

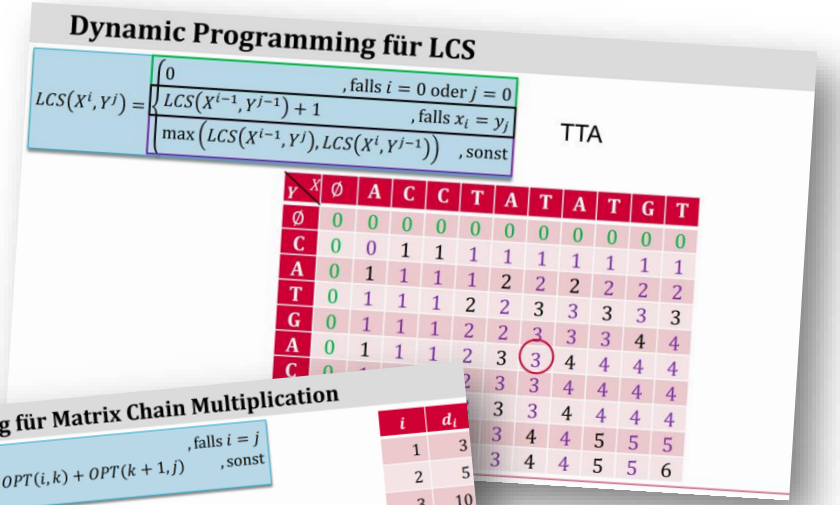
- Independent Set (unabhängige Menge bzw. stabile Menge)
  - **Gegeben:** ungerichteter Graph $G = (V, E)$
  - **Gesucht:** möglichst großes $S \subseteq V$ mit $\forall \{v, w\} \subseteq S: vw \notin E$
  - D.h. keine Kanten in S
- Graphfärbung
  - **Gegeben:** ungerichteter Graph $G = (V, E)$
  - **Gesucht:** Färbung $c: V \rightarrow \{1, \ldots, k\}$ mit $c(v) = c(w) \Rightarrow vw \notin E$
    - D.h. keine einfarbigen Kanten
    - Möglichst wenige Farben (möglichst kleines k)
  - Äquivalent: Möglichst wenige Independent Sets $S_i, 1 \leq i \leq k$ mit $\bigcup_{i=1}^{k} S_i = V$.
- Hier betrachtet auf so genannten Intervallgraphen
- Auf allgemeinen Graphen: Wohl kein effizienter Algorithmus für optimale Lösungen…

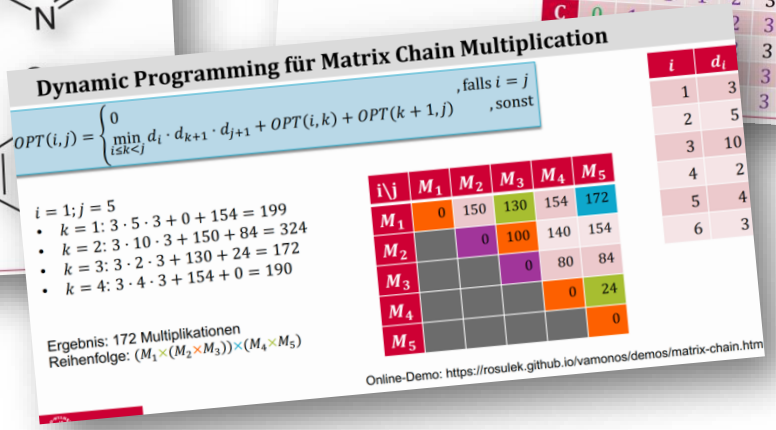


… und nächstes Mal?

Dynamic Programming!







Schon nächste Woche, am 07.05.2025!