FLIPPING OUT

Stefan Finckh, Biko Makinya

PROBLEMBESCHREIBUNG

- Es existieren Erfinder und Räume
- Raumverteilung:
  - Jeder Erfinder schreibt zwei Raumnummern auf die beiden Seiten einer fairen Münze (d.h. eine Nummer pro Seite)
  - Alle Münzen werden gleichzeitig geworfen
  - Falls ein Raum mehrfach verteilt wurde werden alle Münzen erneut geworfen
- Der Host des Events ist selbst ein Erfinder und will einen Vorteil bei der Verteilung

SIE SIND GEFRAGT!

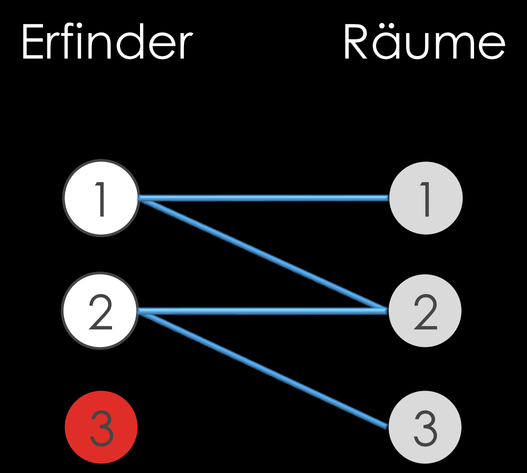
- Host will Räume bewerten, Programm gibt zwei Raumnummern aus
- Programm erhält Münzen aller anderen Erfinder als Input
- Keine Raumverteilung möglich ⇒ „impossible“ ausgeben
- Niemals zwei Raumnummern vorschlagen, die die Raumverteilung unmöglich machen

INPUT

- Erste Zeile: $2 \leq n < 50\,000$: Anzahl Erfinder = Anzahl Räume
- Folgende $n - 1$ Zeilen: Münzen der (n – 1) Erfinder (d.h. ohne den Host)
  - Zeile i: "a b" mit ($1 \leq a < b \leq n$) und a und b sind die Raumwünsche von Erfinder i
- Letzte Zeile: n Integer $v_1,\ldots,v_n$ mit v_i = Bewertung des Raums i
- Bsp.:

Input

3
1 2
2 3
100 40 70



OUTPUT

- Eine Zeile mit den Raumnummern a und b die vom Host ausgewählt werden sollten um die erwartete Bewertung zu maximieren.
- Falls es mehrere optimale Möglichkeiten gibt, soll das Programm die beiden kleinsten Raumnummern in aufsteigender Reihenfolge auswählen.

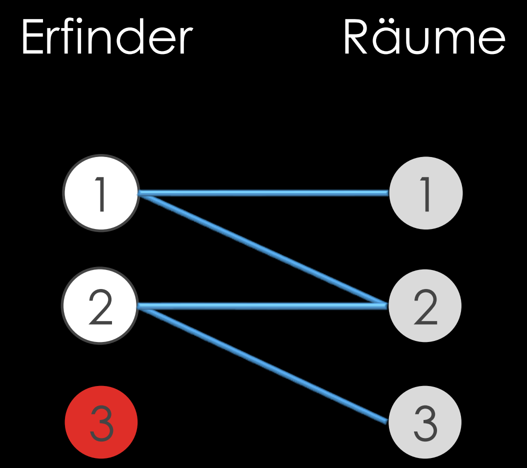
- Bsp.:

Input

3
1 2
2 3
100 40 70

Output:

1 3



AUSGABE

Verfügbare Räume ermitteln:

- Impossible
- ein verfügbarer Raum i
  - Output: i <kleinster unmöglicher Raum> bzw. Umgekehrt
- Alle Räume verfügbar
  - Output: i j mit i < j und i, j haben die höchsten Bewertungen von allen Räumen

- 1 < |verfügbare Räume| < n
  - Seien i j die verfügbaren Räume mit den beiden höchsten Bewertungen. Und sei k die niedrigste Raumnr. eines nicht verfügbaren Raumes
  - Falls Bewertung(i) = Bewertung(j):

    Output: die beiden niedrigsten Elemente aus {i, j, k} in aufsteigender Reihenfolge
  - Falls Bewertung(i) ≠ Bewertung(j):

    Output: <am höchsten bewerteter verfügbarer Raum> k bzw. umgekehrt

IMPOSSIBLE

Input

4

1 2

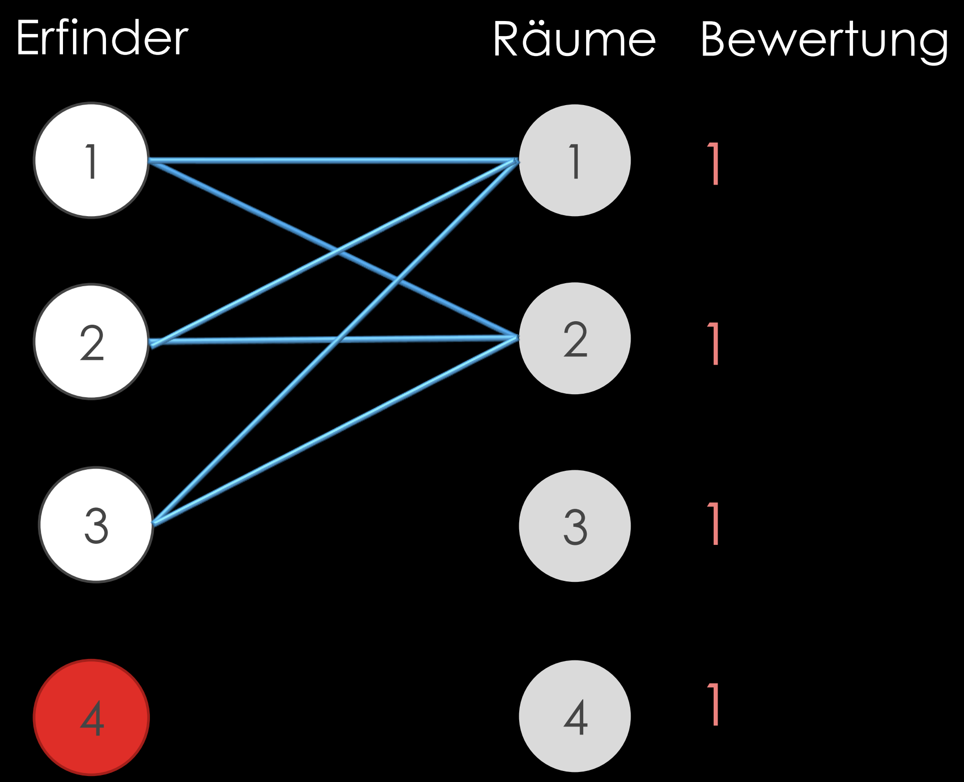
1 2

1 2

1 1 1 1

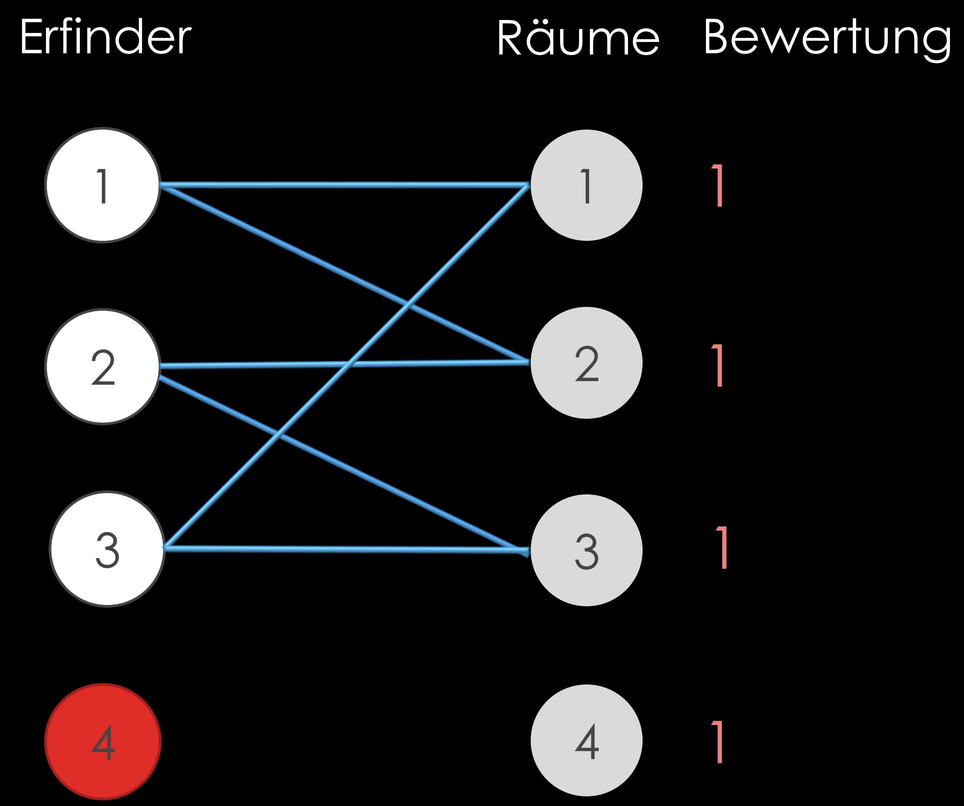
Output

impossible



EIN VERFÜGBARER RAUM

Input	Output
4	1 4
1 2	
2 3	
1 3	
2 3 4 1	



ALLE RÄUME VERFÜGBAR

Input

4

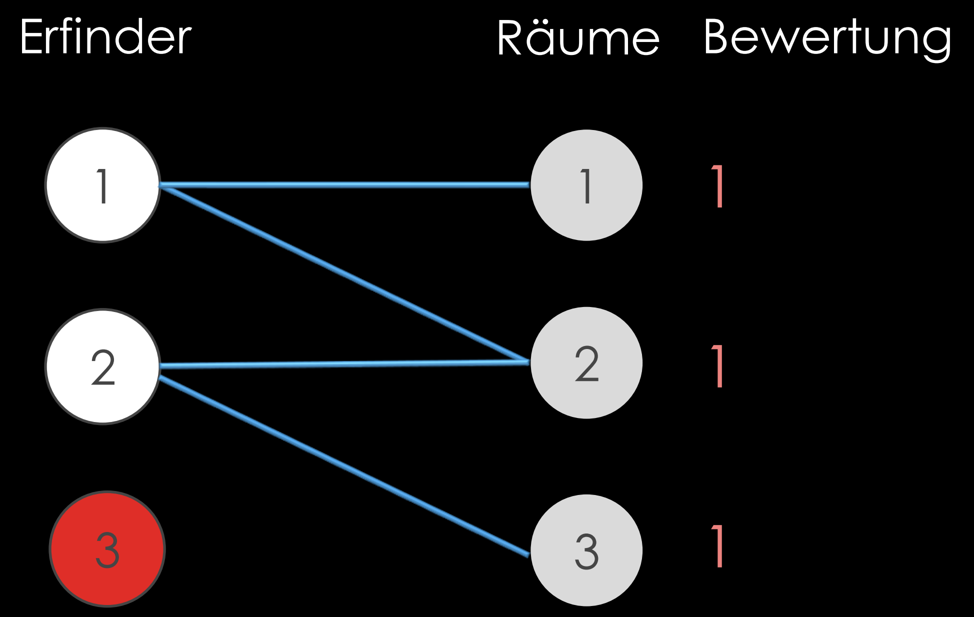
1 2

2 3

100 40 70

Output

1 3



BEWERTUNG(I) = BEWERTUNG(J)

Input

4

1 2

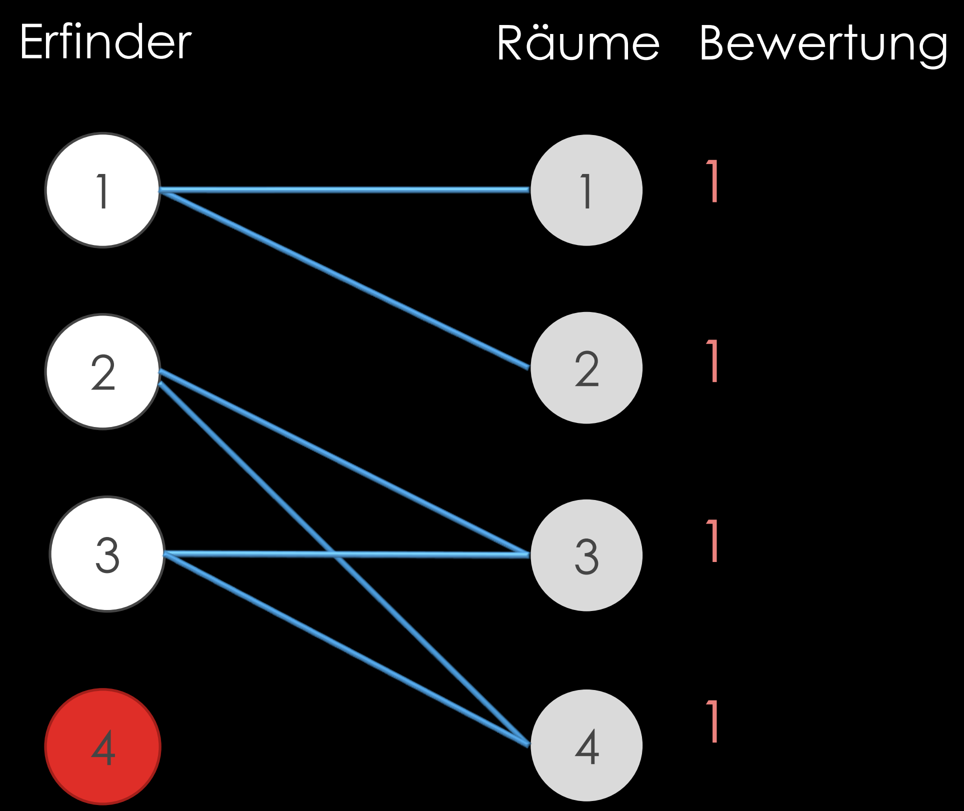
3 4

3 4

1 1 1 1

Output

1 **2**



BEWERTUNG(I) ≠ BEWERTUNG(J)

Input

4

1 2

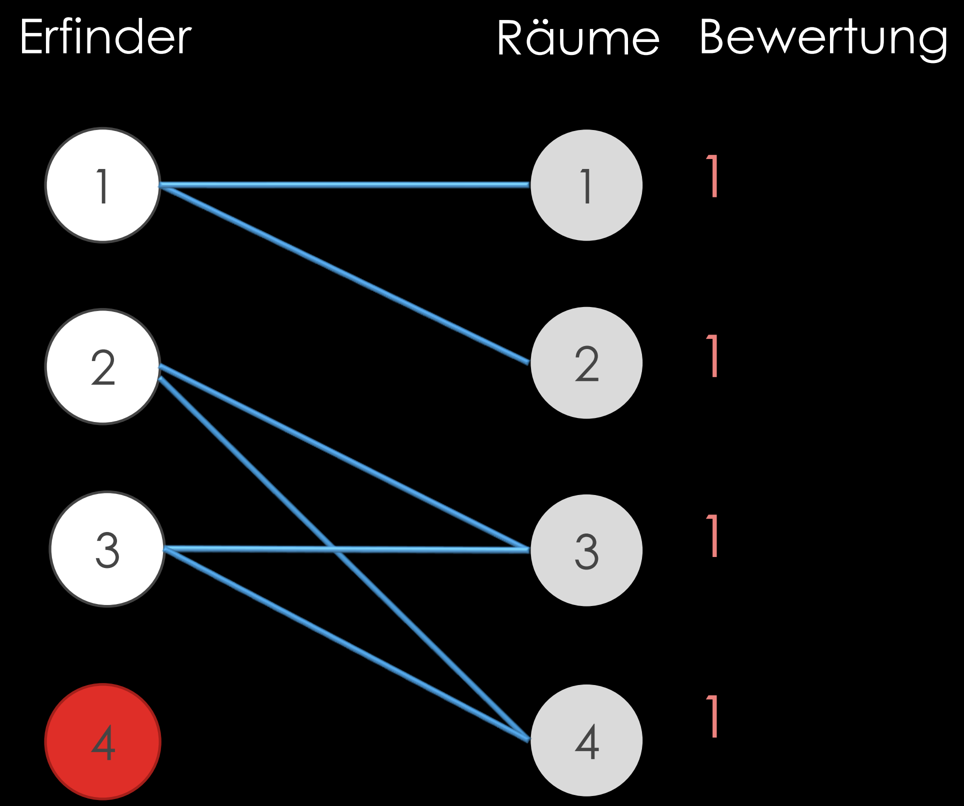
3 4

3 4

2 1 1 1

Output

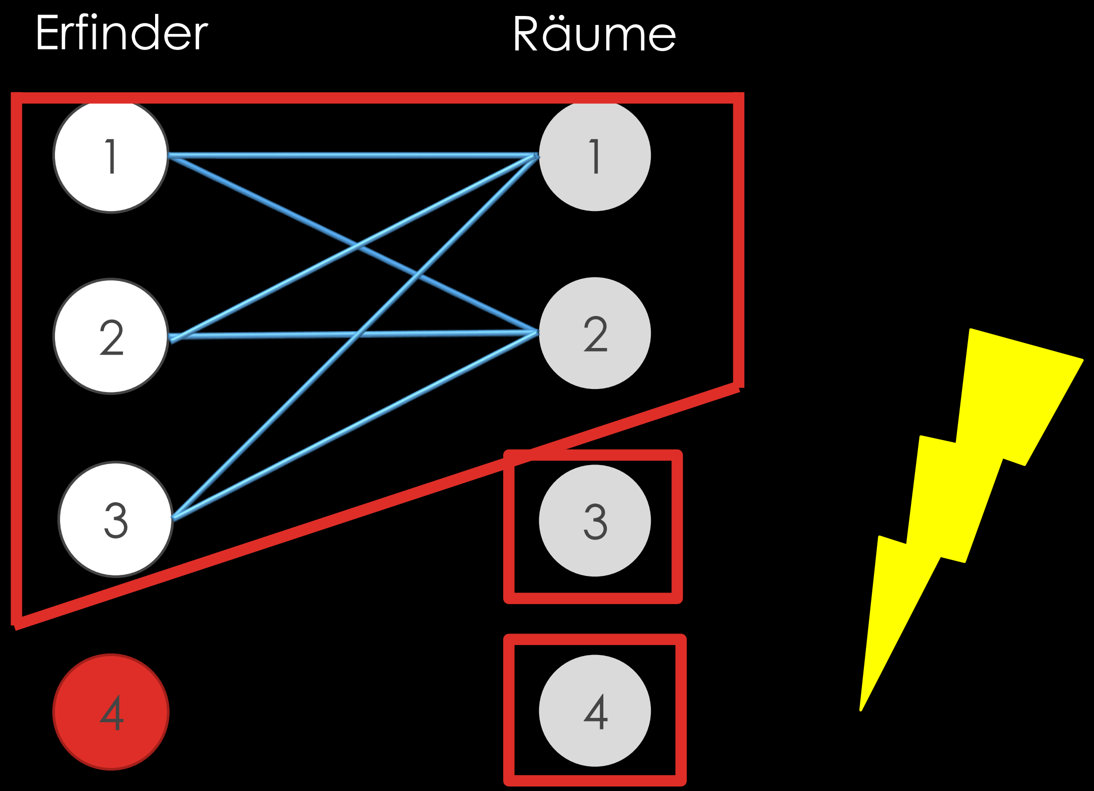
1 **3**



ANSATZ

Input

4
1 2
1 2
1 2
1 1 1 1



Input

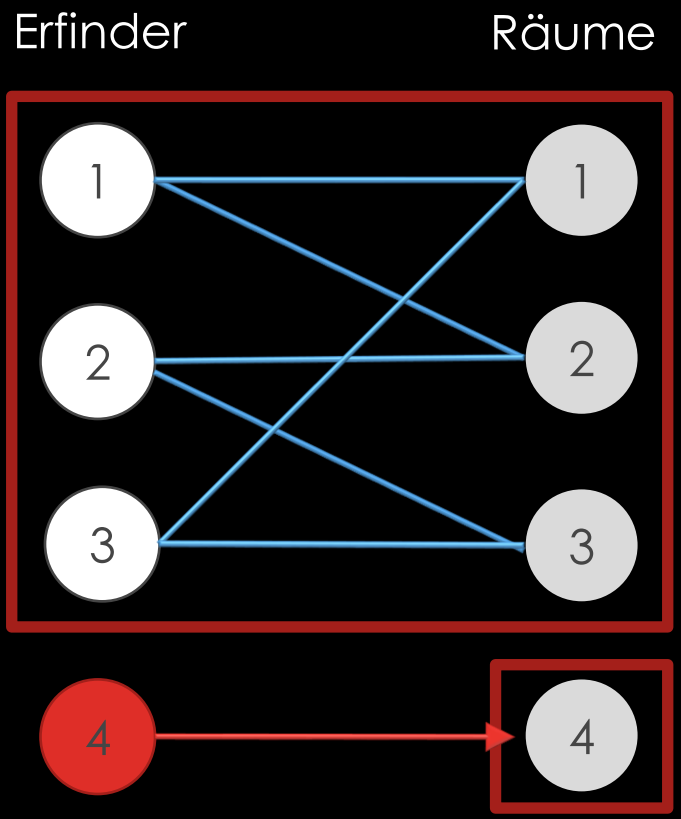
4

1 2

2 3

1 3

2 3 4 1



ZUSAMMENFASSUNG

1. Weniger Erfinder als Räume in ZK
   $\Rightarrow$ Alle Räume in ZK verfügbar

2. Gleich viele Erfinder wie Räume in ZK
   $\Rightarrow$ Kein Raum in ZK verfügbar

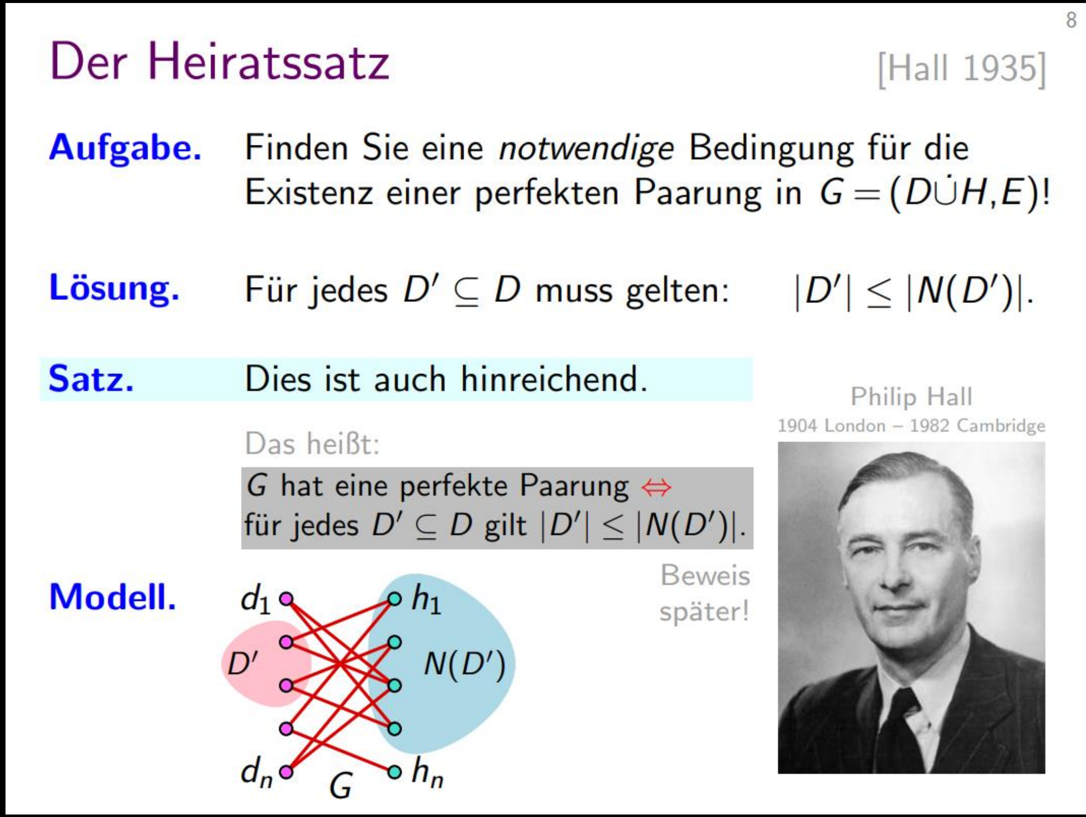
3. Mehr Erfinder als Räume in ZK
   $\Rightarrow$ Impossible

IMPLEMENTIERUNG

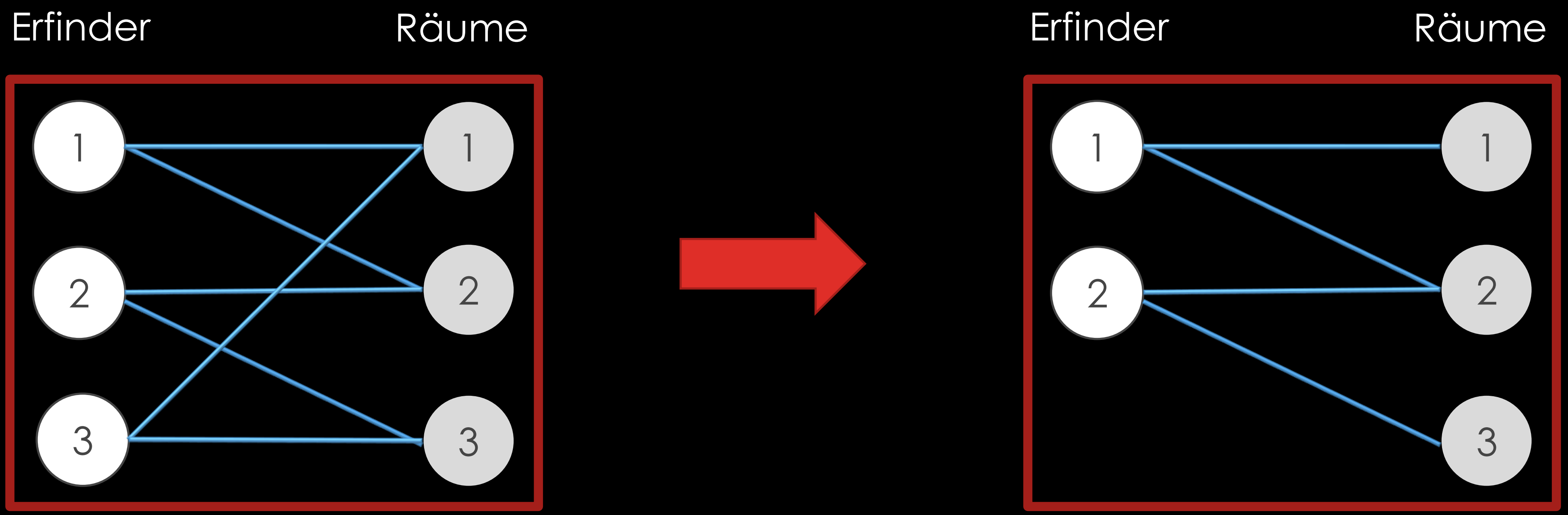
Es empfiehlt sich die verfügbaren Räume so abzuspeichern dass man für einen Raum in konstanter Zeit ermitteln kann ob er verfügbar ist
z.B. HashSet in Java

KORREKTHEIT

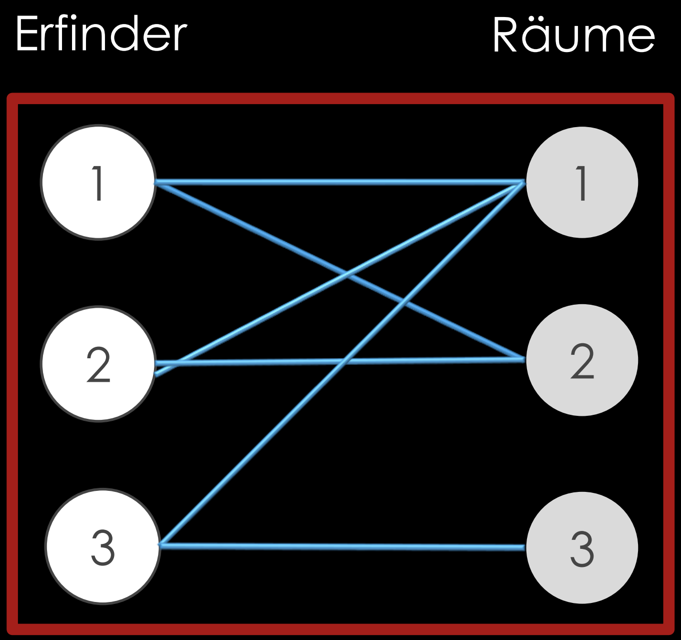
- Korrekte Raumverteilung ⇔ perfekte Paarung
- Spezifische Teilmengen reichen in unserem Fall aus

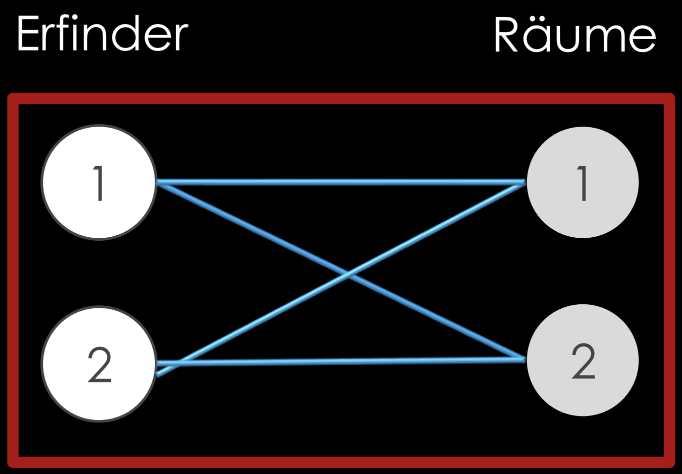


KEIN RAUM VERSCHWINDET

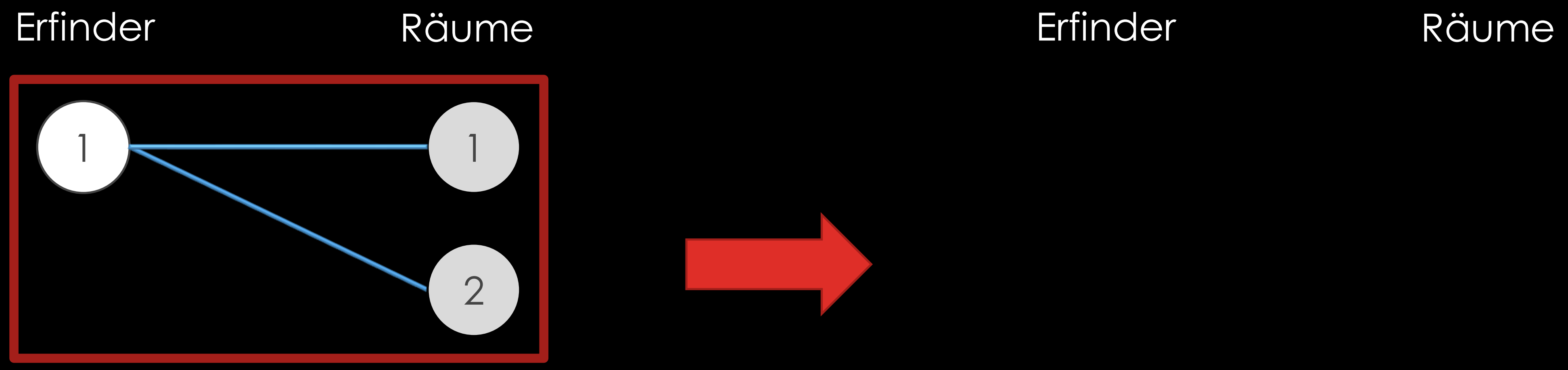


EIN RAUM VERSCHWINDET

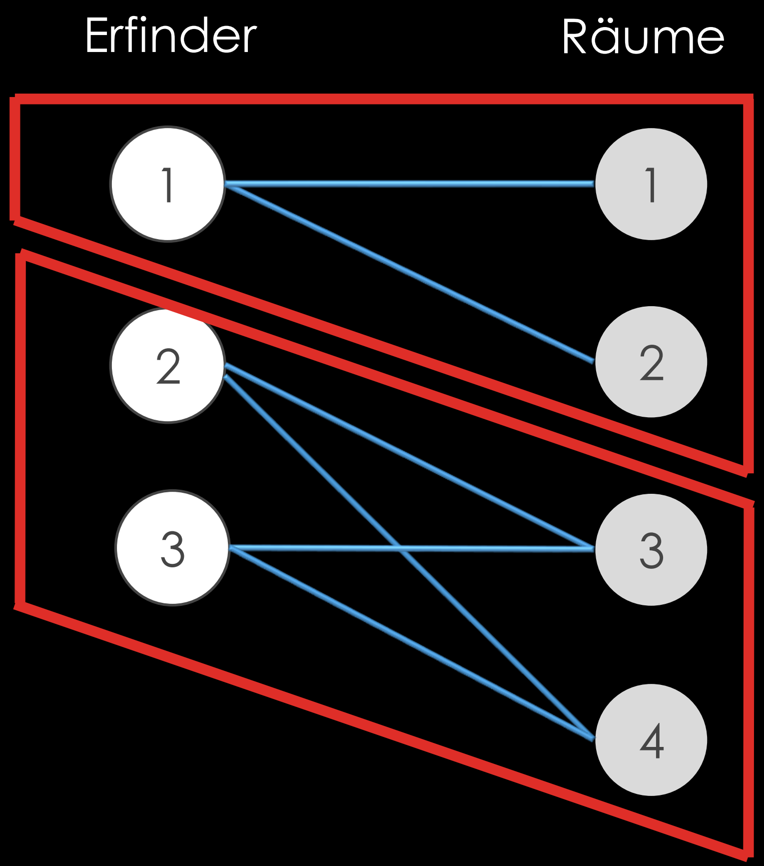




ZWEI RÄUME VERSCHWINDEN



VEREINIGUNG VON TEILMENGEN



BEI TECHNIKFRAGEN TECH-NICK FRAGEN