Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Problem Z

Jinxed Betting

Algorithmen für Programmierwettbewerbe

Jonathan Klawitter
Folien von Philipp Kindermann

Problem

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ausgangslage

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

Team 1 vs. Team 2

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

Team 1 vs. Team 2

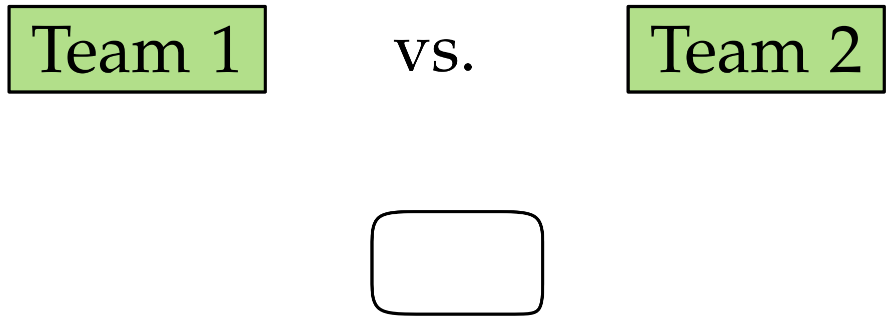
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

Team 1 vs. Team 2

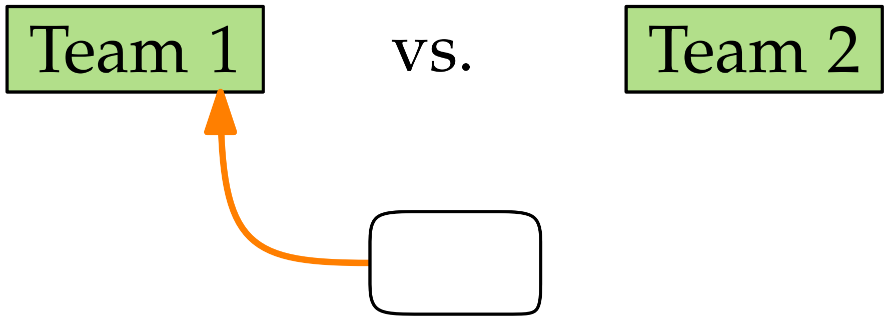
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



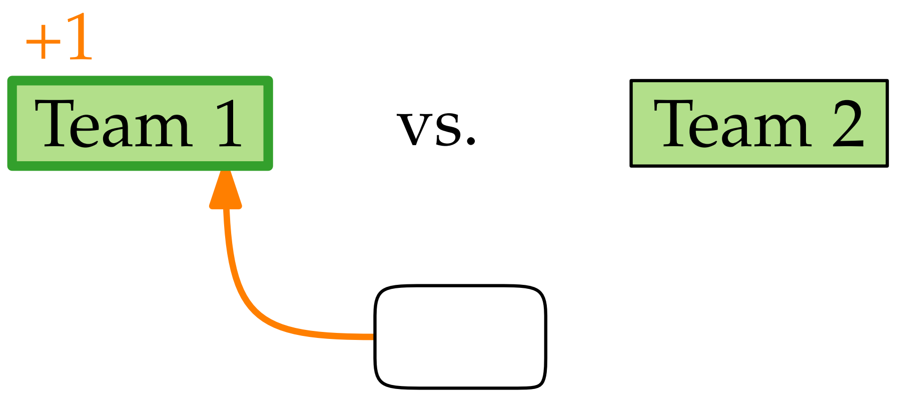
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



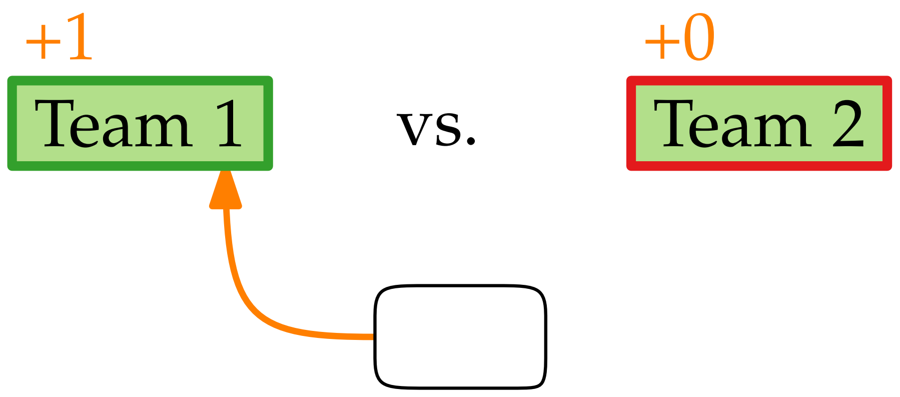
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



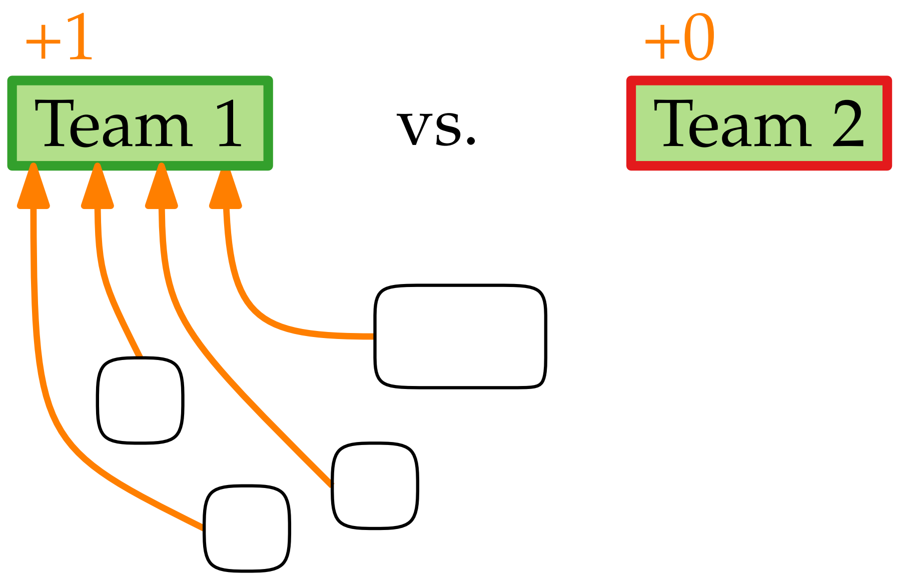
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



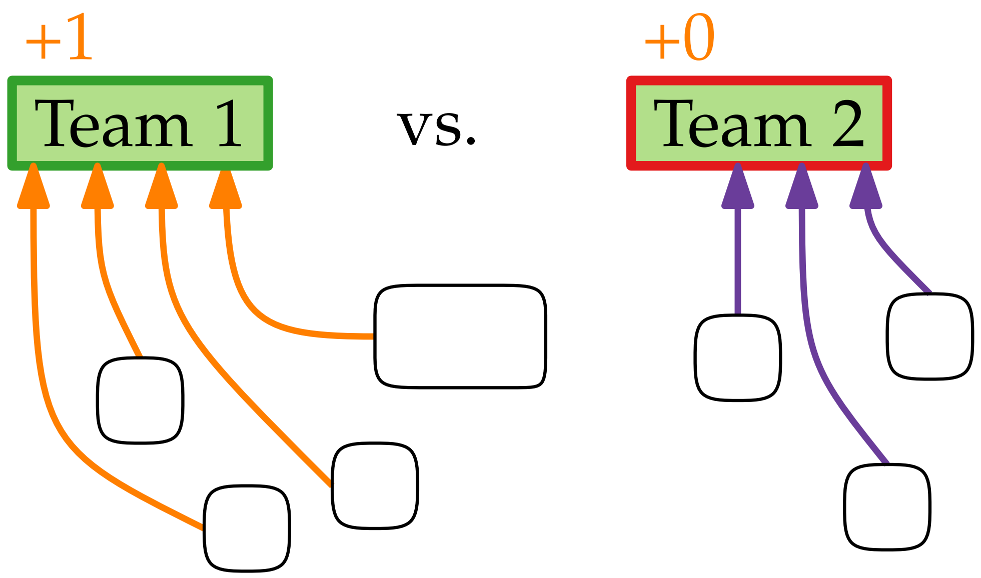
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



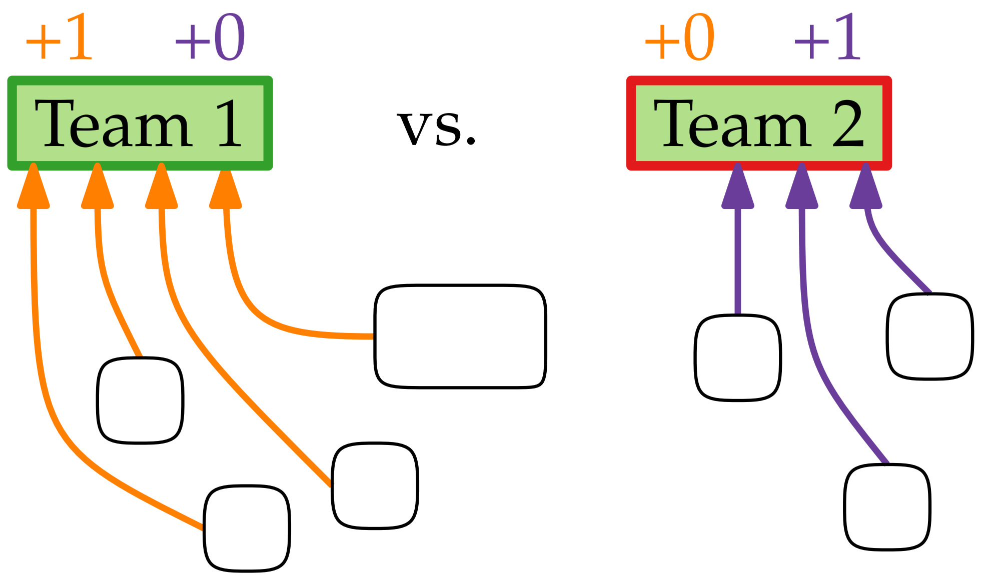
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



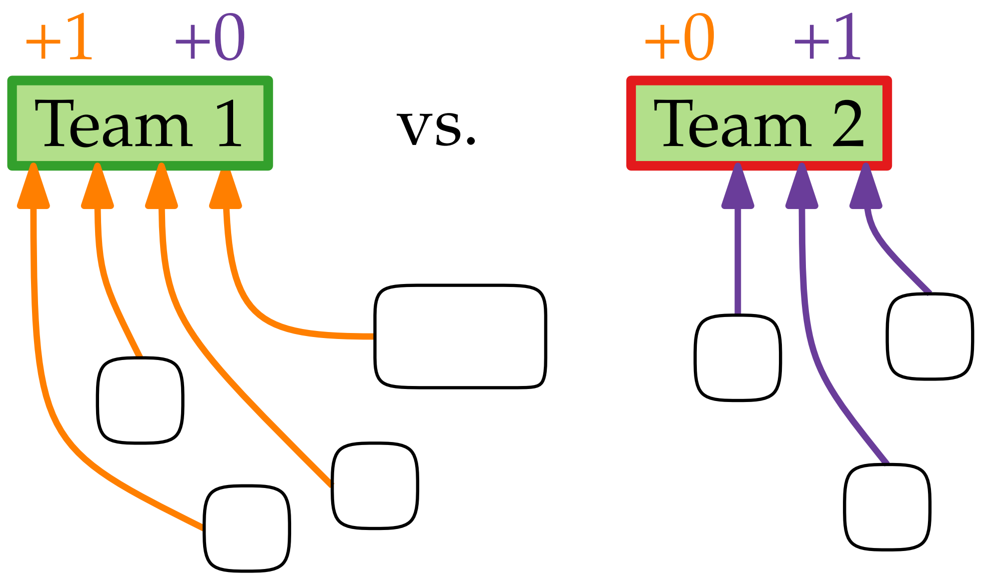
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



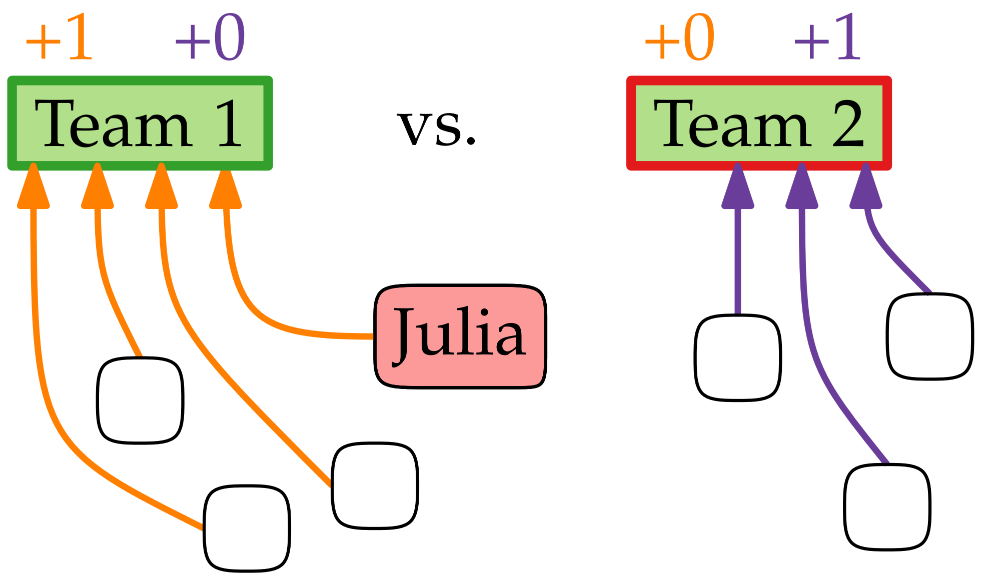
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage



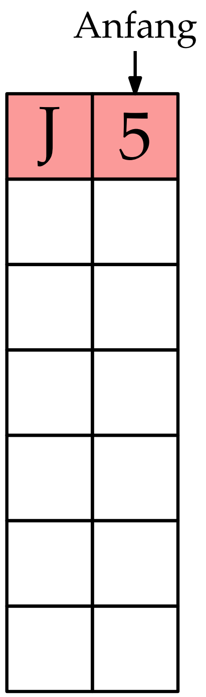
Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

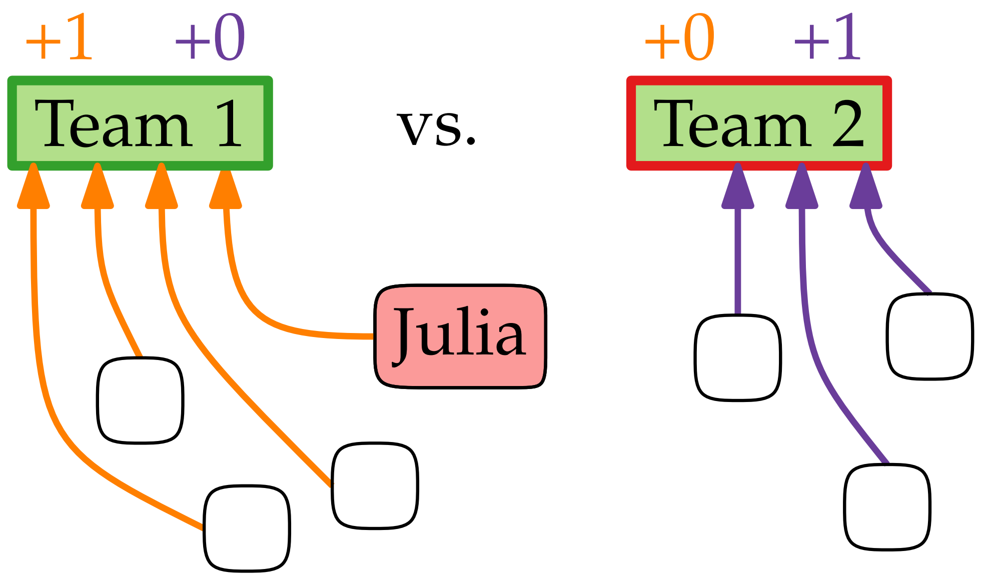
Ausgangslage



Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

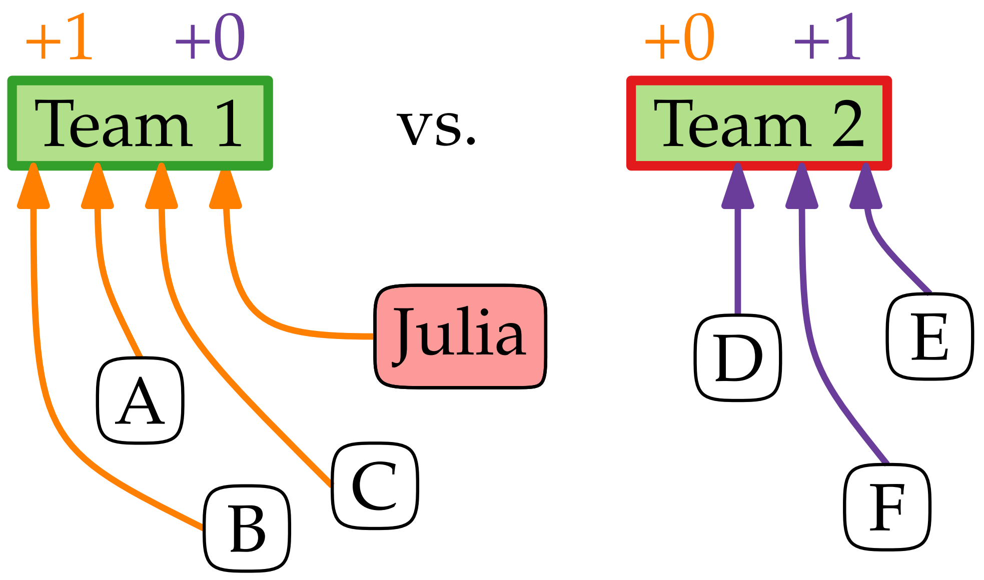




Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Ausgangslage

	Anfang
J	5
A	1
B	4
C	4
D	3
E	2
F	4

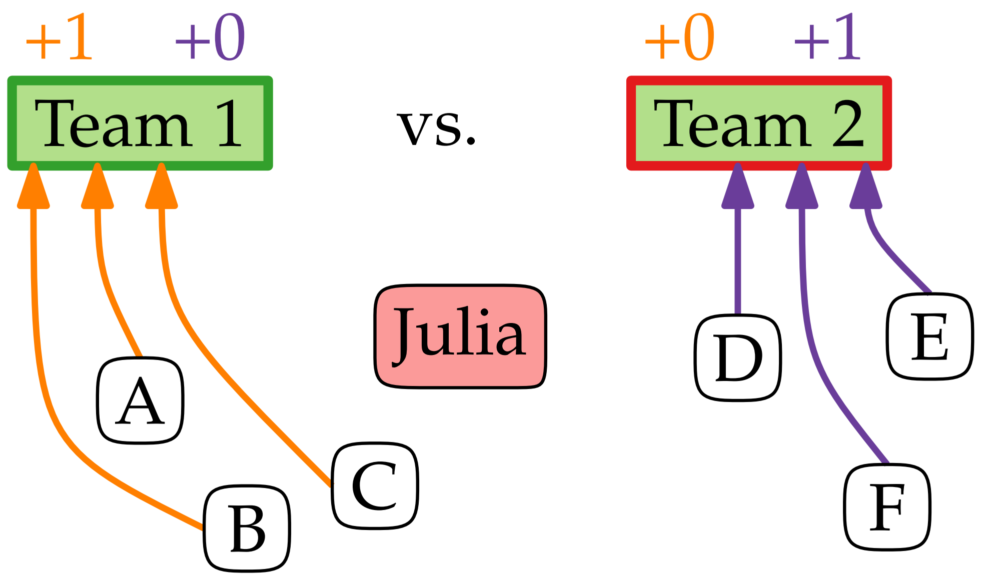


Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

Vorgehensweise

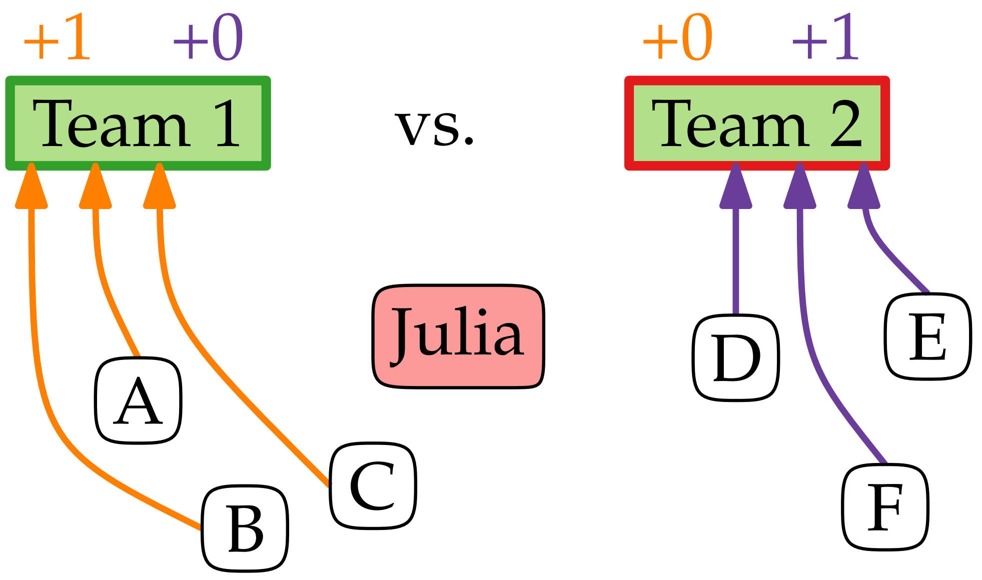
Anfang

J	5
A	1
B	4
C	4
D	3
E	2
F	4



She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Vorgehensweise



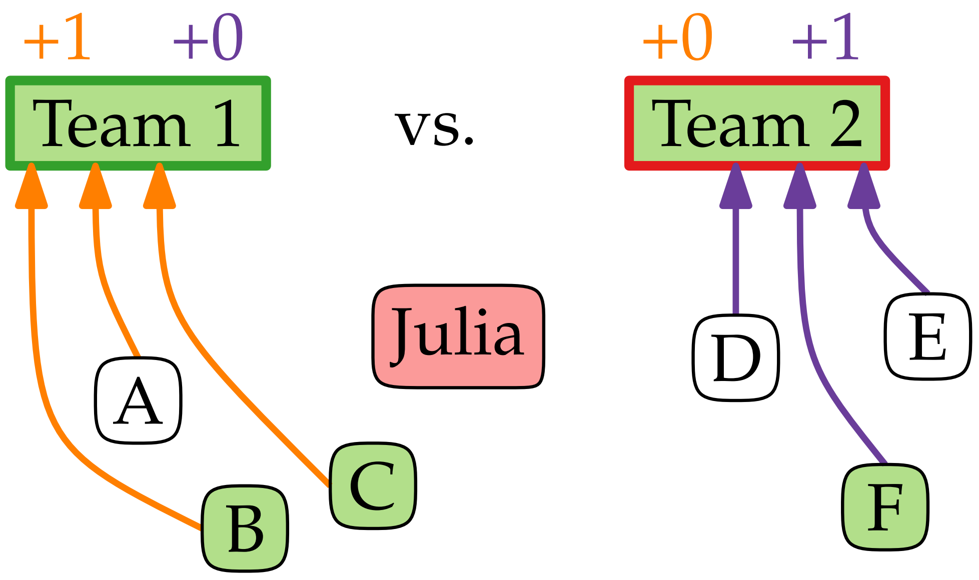
Anfang

J	5
A	1
B	4
C	4
D	3
E	2
F	4

She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Vorgehensweise

	Anfang
J	5
A	1
B	4
C	4
D	3
E	2
F	4

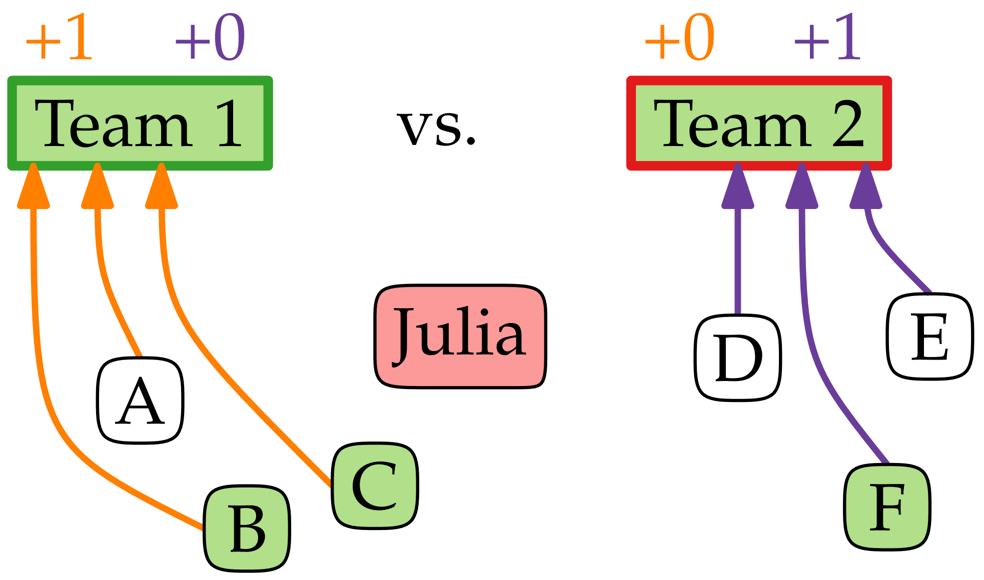


She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Vorgehensweise

Anfang

J	5
A	1
B	4
C	4
D	3
E	2
F	4

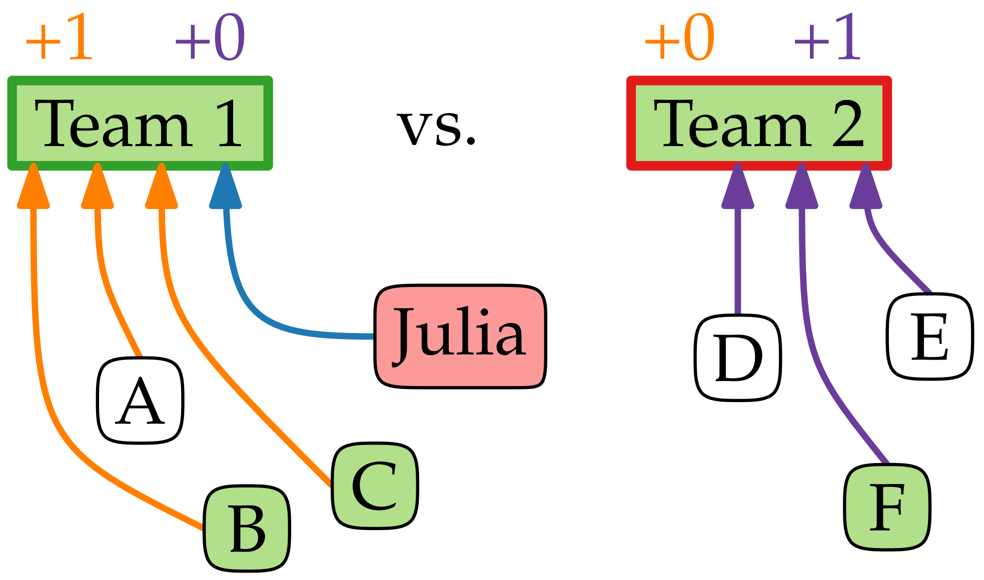


She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Vorgehensweise

Anfang

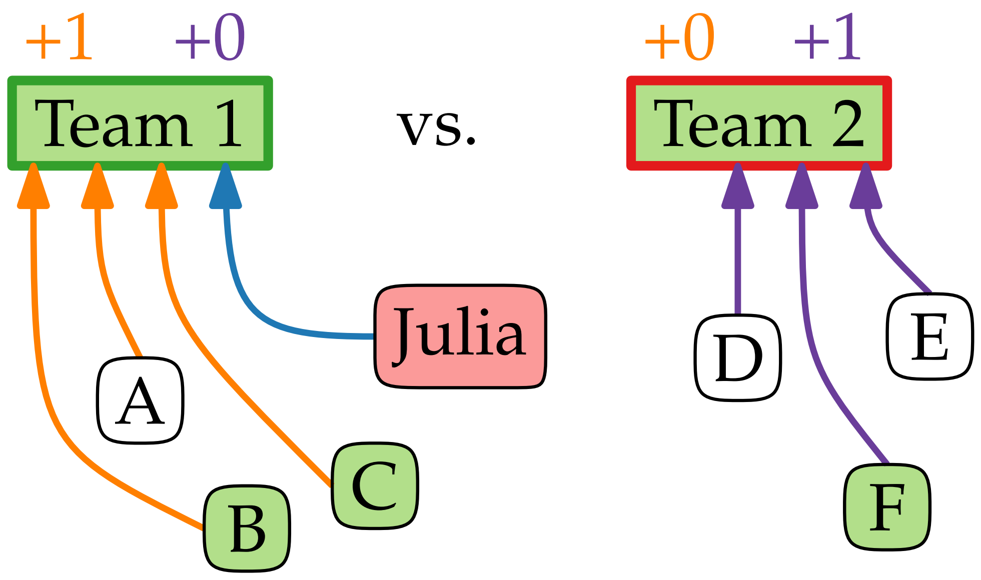
J	5
A	1
B	4
C	4
D	3
E	2
F	4



She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Ergebnis

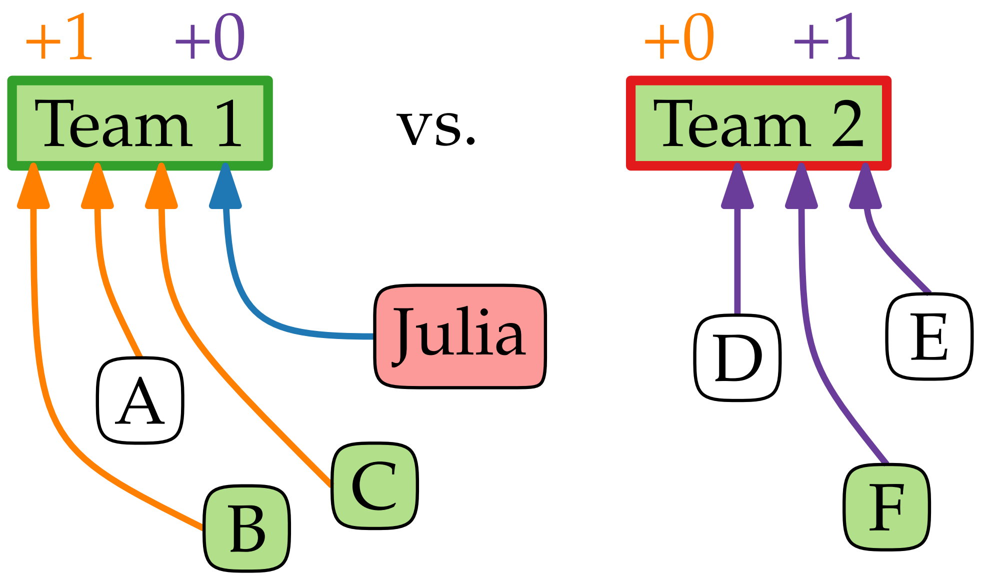
	Anfang
J	5
A	1
B	4
C	4
D	3
E	2
F	4



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

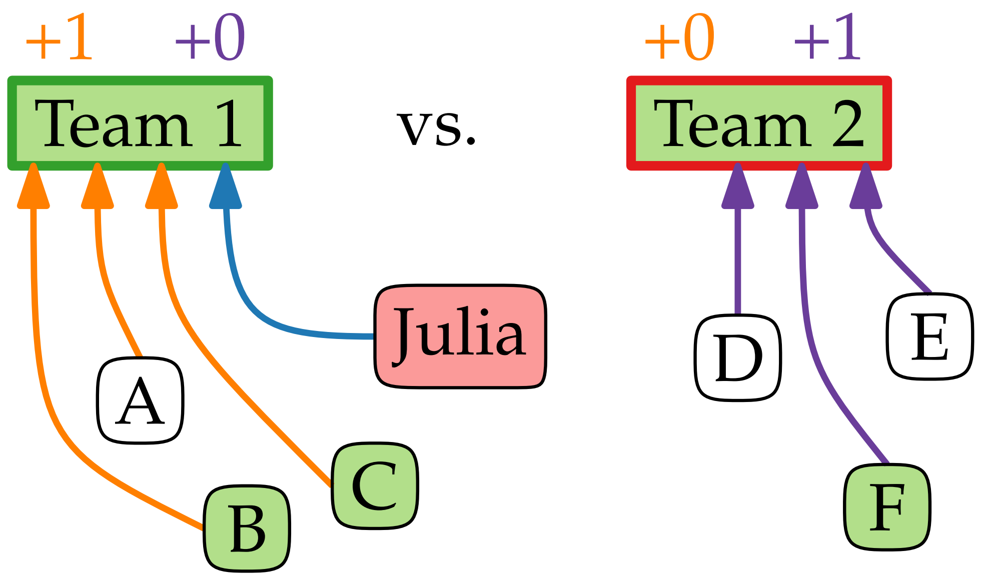
	Anfang
J	5
A	1
B	4
C	4
D	3
E	2
F	4



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!

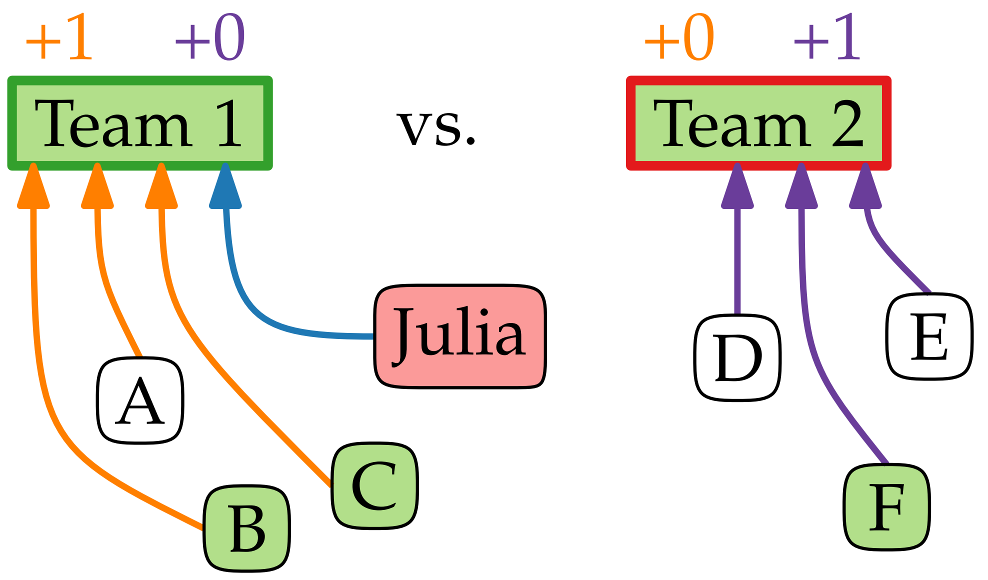


	Anfang
J	5
A	1
B	4
C	4
D	3
E	2
F	4

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!

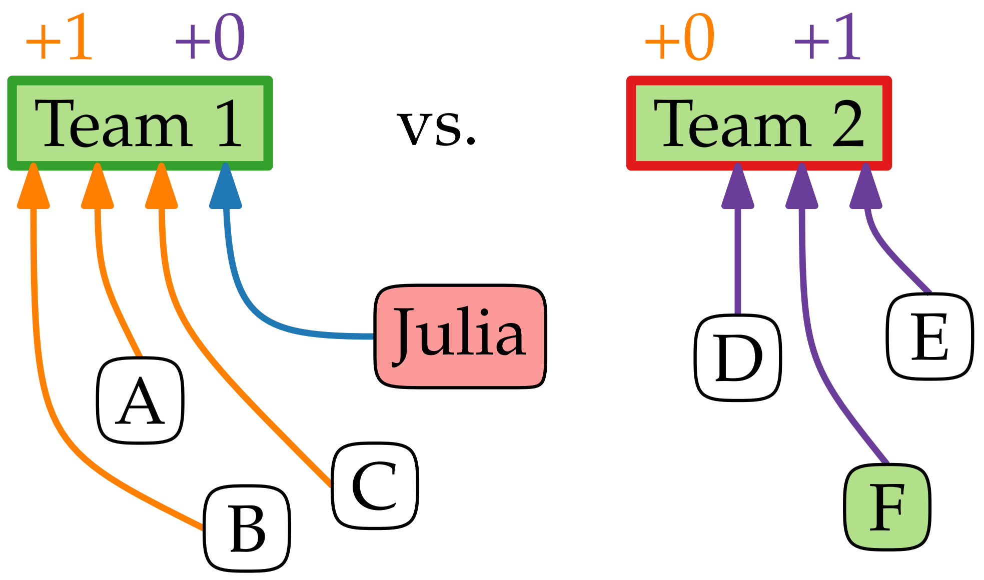


	Anfang	
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!



Anfang

J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

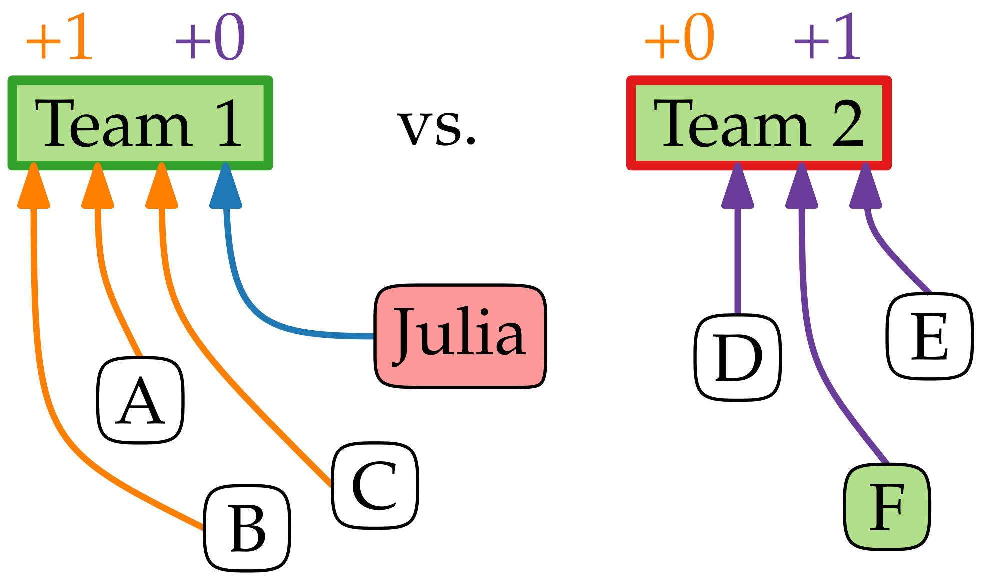
Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

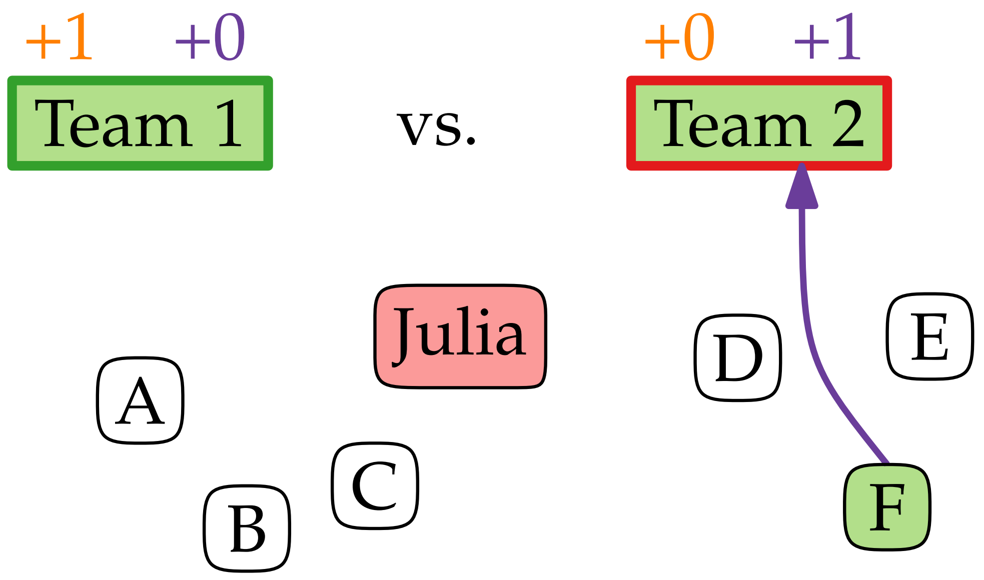
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

Team 2 wins!



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

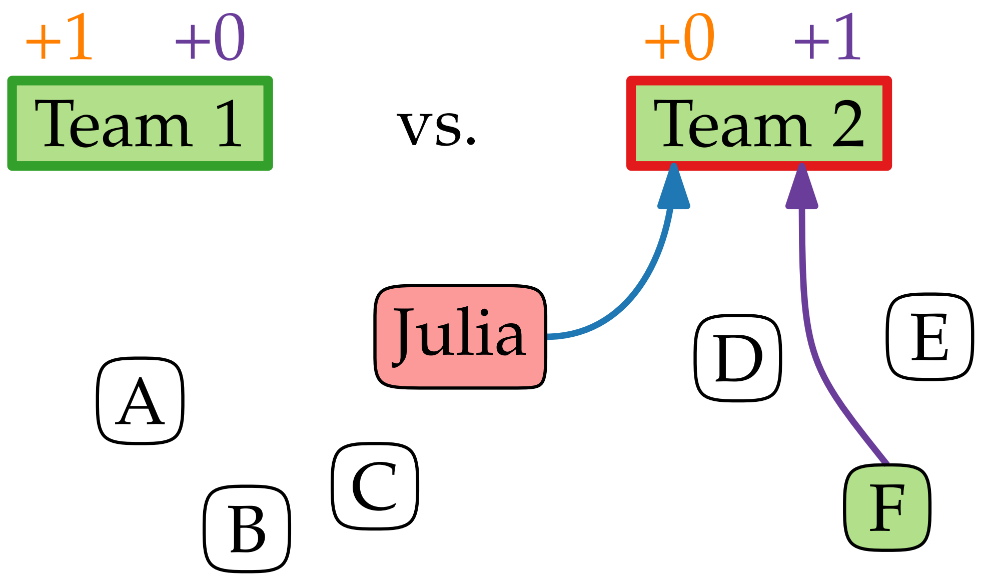


Anfang

J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis



Anfang

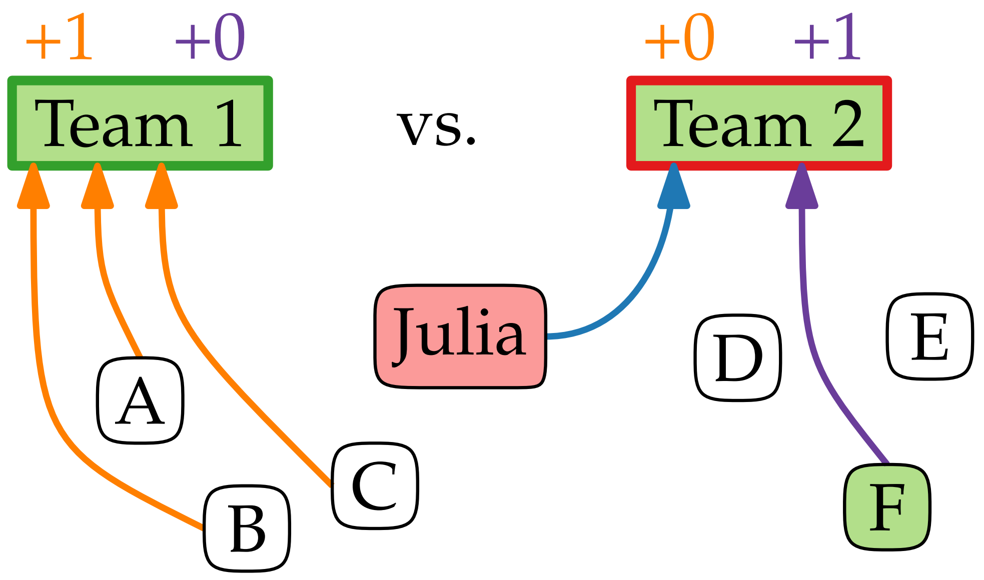
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

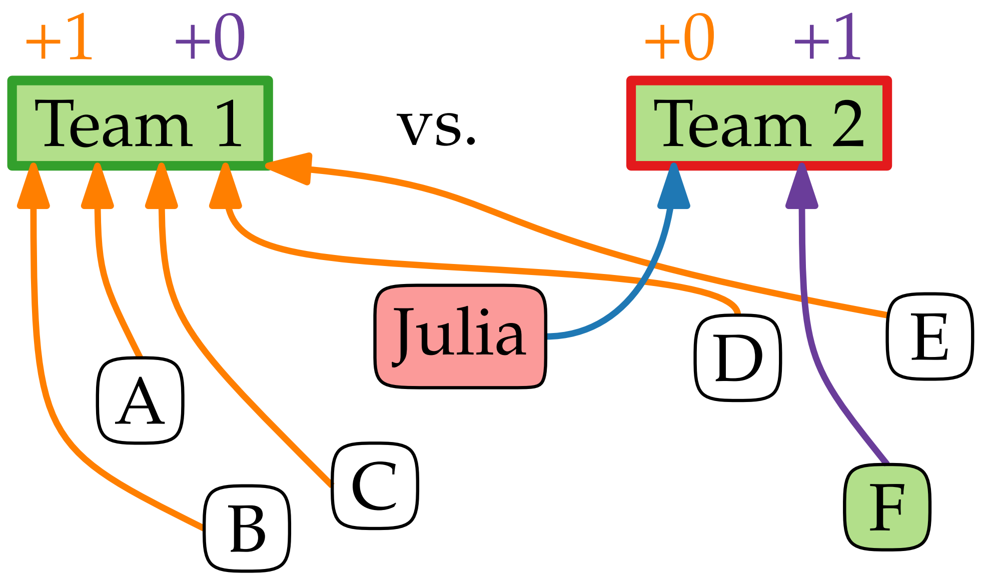
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

	Anfang	
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5

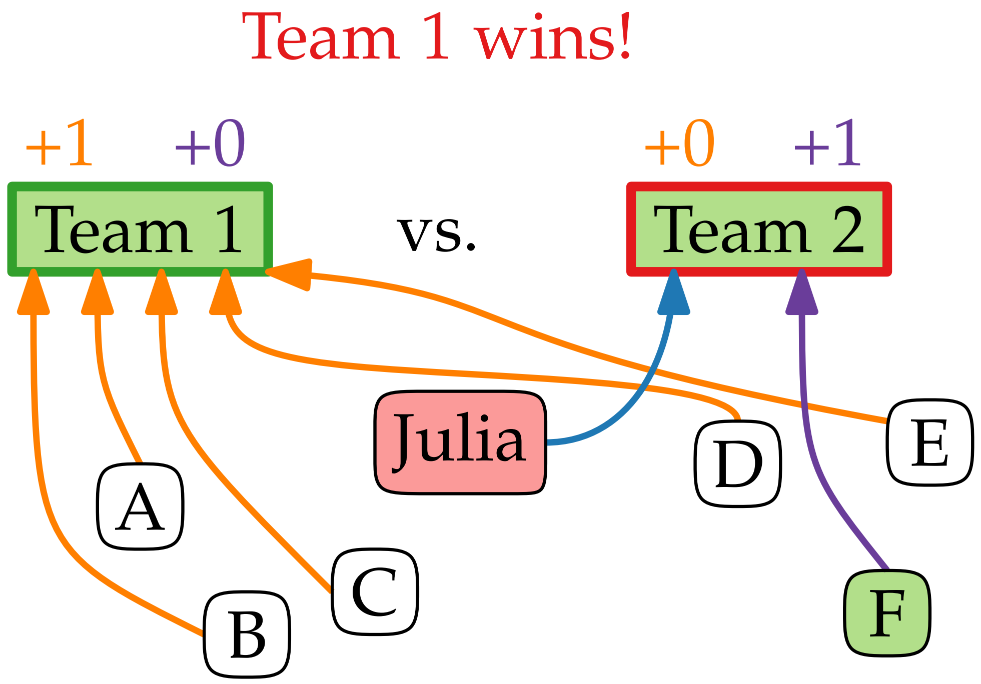


Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

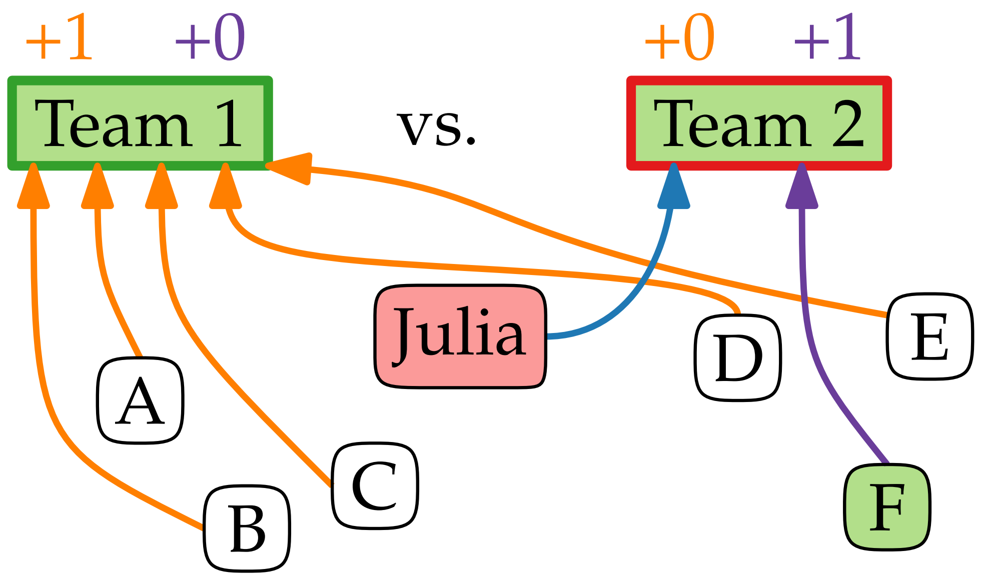
J	5	5
A	1	1
B	4	4
C	4	4
D	3	4
E	2	3
F	4	5



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 1 wins!



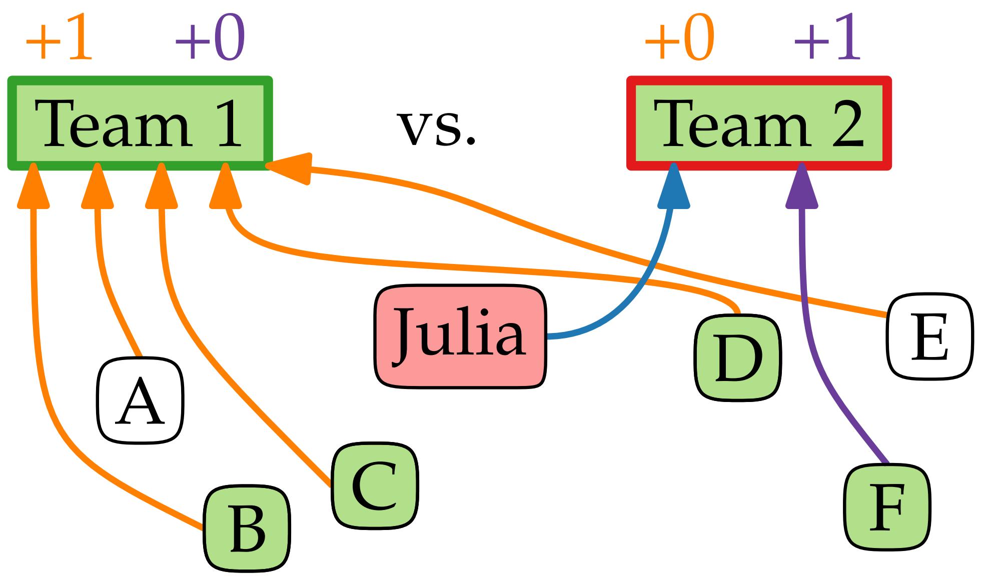
Anfang

J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 1 wins!



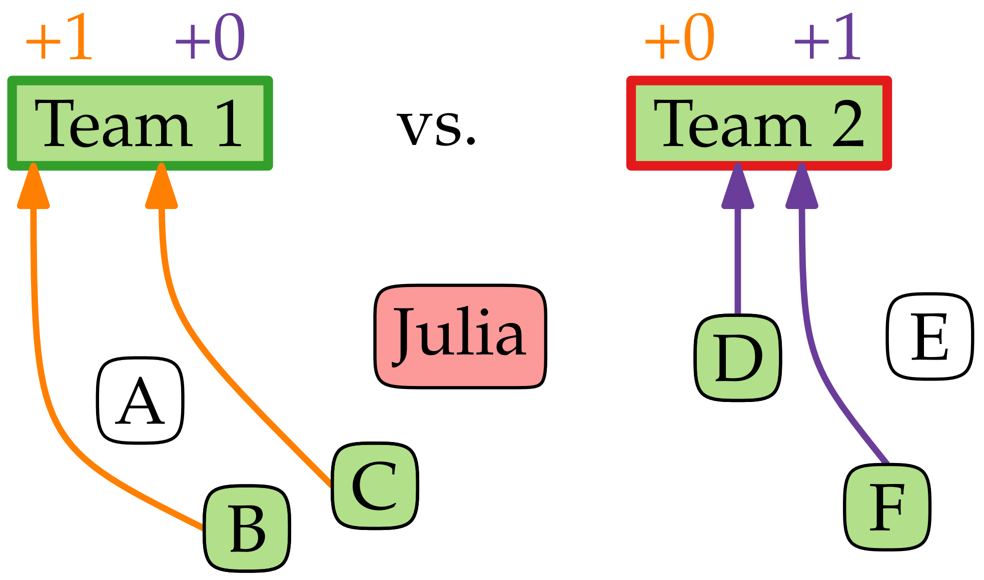
Anfang

J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

	Anfang		
J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5

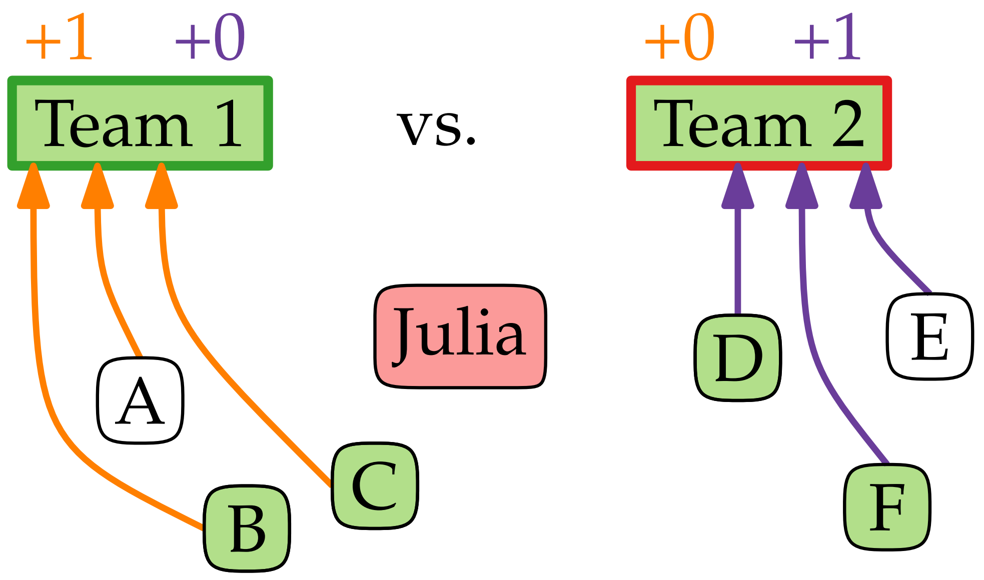


Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang ↓

J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5

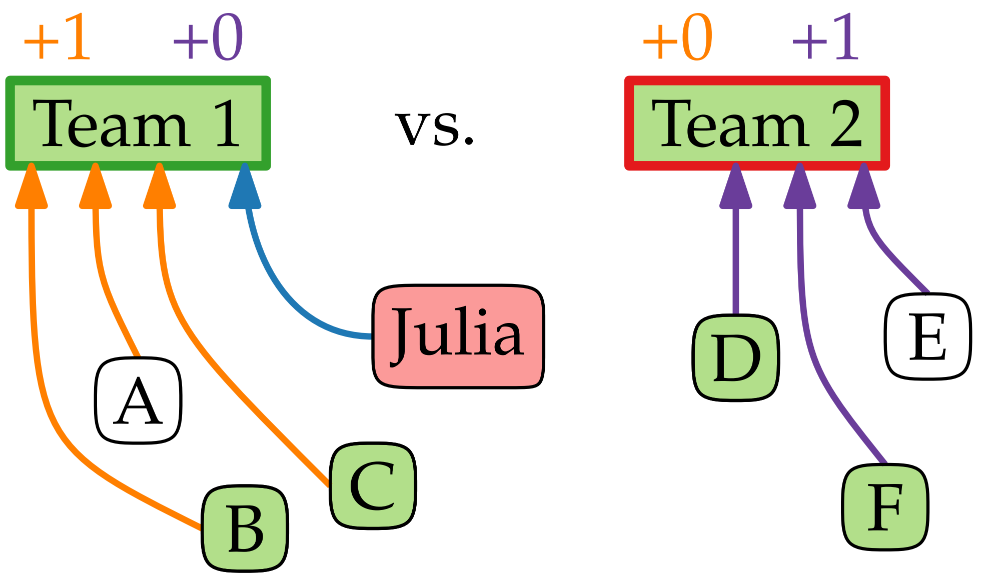


Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5



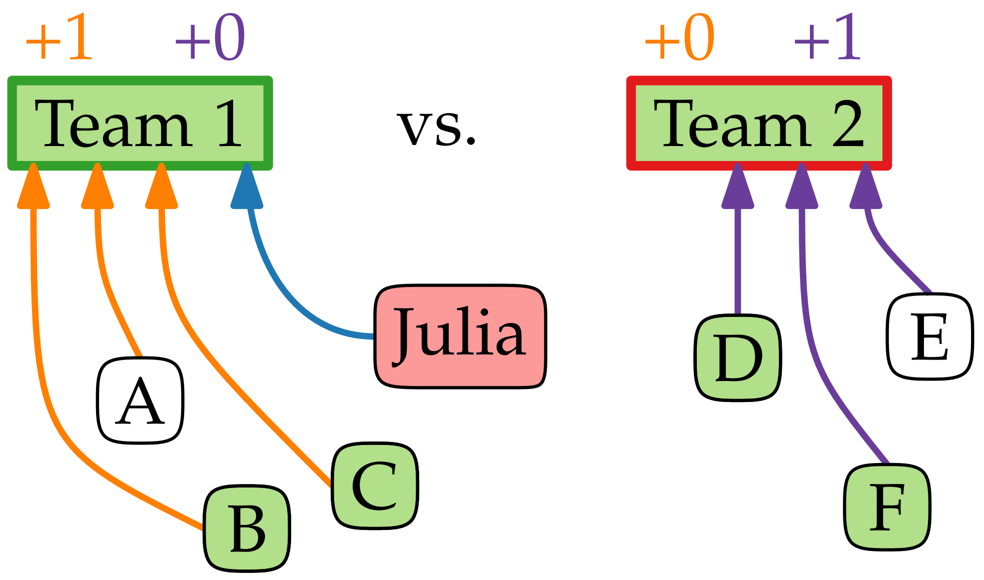
Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

J	5	5	5
A	1	1	2
B	4	4	5
C	4	4	5
D	3	4	5
E	2	3	4
F	4	5	5

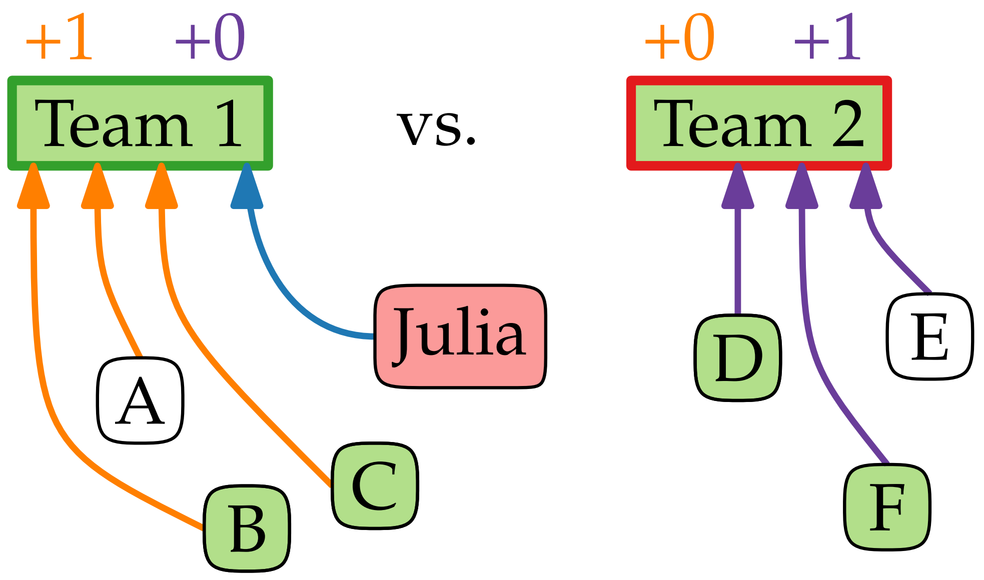
Team 2 wins!



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!



Anfang

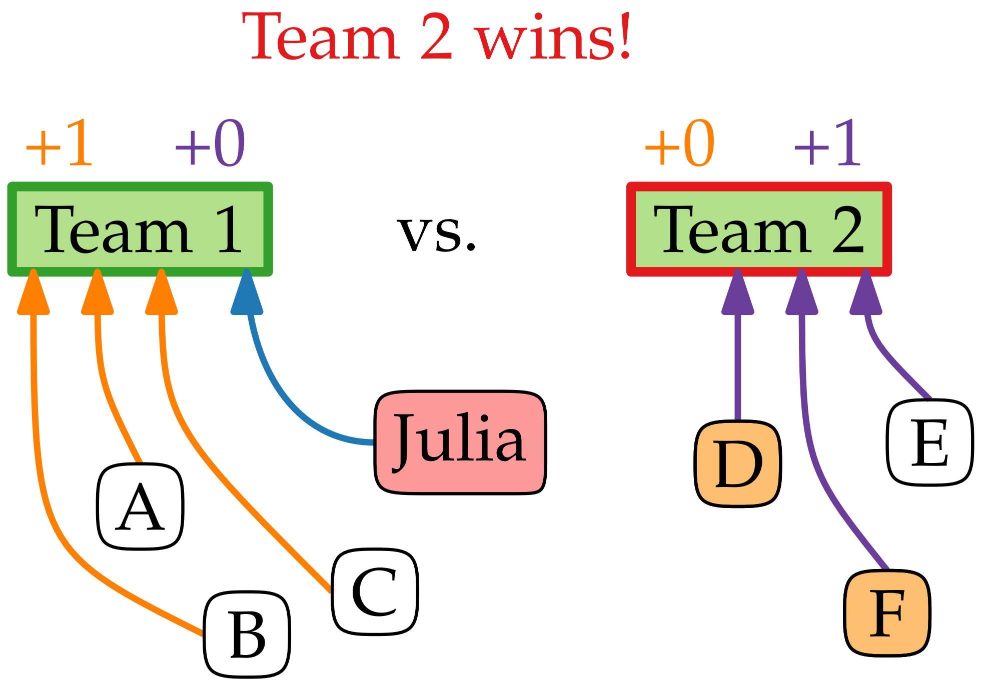
J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	5
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Anfang

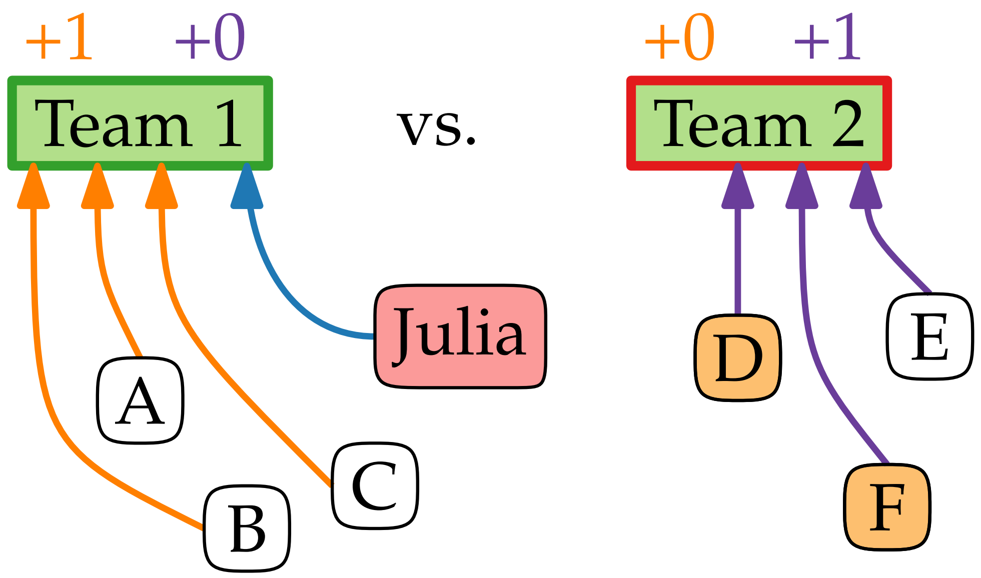
J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	5
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6



Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!



Anfang

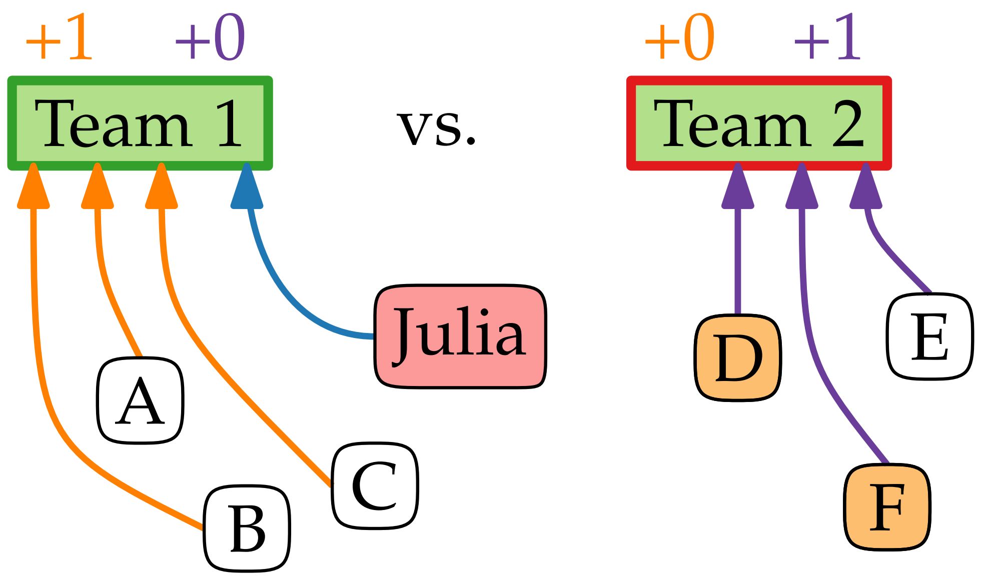
J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	5
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ergebnis:

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ergebnis

Team 2 wins!



Anfang

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	5
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ergebnis: 2

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Ein-/Ausgabe

The input consists of:

- One line with an integer n $(3 \leq n \leq 10^5)$, the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ $(0 \leq p_i \leq 10^{16}$ for each $i)$, the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Input:

```
7
5 1 4 4 3 2 4
```

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output the number of matches for which Julia is guaranteed to stay in the lead.

Input:

```
7
5 1 4 4 3 2 4
```

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output the number of matches for which Julia is guaranteed to stay in the lead.

Input:

```
7
5 1 4 4 3 2 4
```

Output:

```
2
```

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output the number of matches for which Julia is guaranteed to stay in the lead.

Input:

```
7
5 1 4 4 3 2 4
```

Output:

```
2
```

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Ein-/Ausgabe

The input consists of:

- One line with an integer n ($3 \leq n \leq 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \leq p_i \leq 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output the number of matches for which Julia is guaranteed to stay in the lead.

Input:

```
7
5 1 4 4 3 2 4
```

Output:

```
2
```

J	5	5	5	5
A	1	1	2	2
B	4	4	5	5
C	4	4	5	6
D	3	4	5	6
E	2	3	4	5
F	4	5	5	6

Bruteforce

Probiere alles aus.

Bruteforce

Probiere alles aus.

Pro Runde:

Bruteforce

Probiere alles aus.

Pro Runde:

- $n - 1$ Teilnehmer haben eine Wahl

Bruteforce

Probiere alles aus.

Pro Runde:

- $n - 1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$

Bruteforce

Probiere alles aus.

Pro Runde:	• $n-1$ Teilnehmer haben eine Wahl	$\Rightarrow 2^{n-1}$
	• 2 mögliche Ergebnisse	$\Rightarrow 2$

Bruteforce

Probiere alles aus.

Pro Runde:

- $n - 1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$
- 2 mögliche Ergebnisse $\Rightarrow 2$
- Bei Gleichstand: Julia hat eine Wahl $\Rightarrow 2$

Bruteforce

Probiere alles aus.

Pro Runde:

- $n-1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$
- 2 mögliche Ergebnisse $\Rightarrow 2$
- Bei Gleichstand: Julia hat eine Wahl $\Rightarrow 2$

r Runden:

Bruteforce

Probiere alles aus.

Pro Runde:

- $n-1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$
- 2 mögliche Ergebnisse $\Rightarrow 2$
- Bei Gleichstand: Julia hat eine Wahl $\Rightarrow 2$

r Runden: $2^{r(n+1)}$

Bruteforce

Probiere alles aus.

Pro Runde:

- $n-1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$
- 2 mögliche Ergebnisse $\Rightarrow 2$
- Bei Gleichstand: Julia hat eine Wahl $\Rightarrow 2$

r Runden: $2^{r(n+1)}$

$n \leq 10^5$, $r \leq 10^{16}$

Bruteforce

Probiere alles aus.

Pro Runde:

- $n-1$ Teilnehmer haben eine Wahl $\Rightarrow 2^{n-1}$
- 2 mögliche Ergebnisse $\Rightarrow 2$
- Bei Gleichstand: Julia hat eine Wahl $\Rightarrow 2$

r Runden: $2^{r(n+1)}$

$n \leq 10^5$, $r \leq 10^{16}$

$\Rightarrow 2^{10^{21}}$

Worst Case

Teile in Gruppen ein:

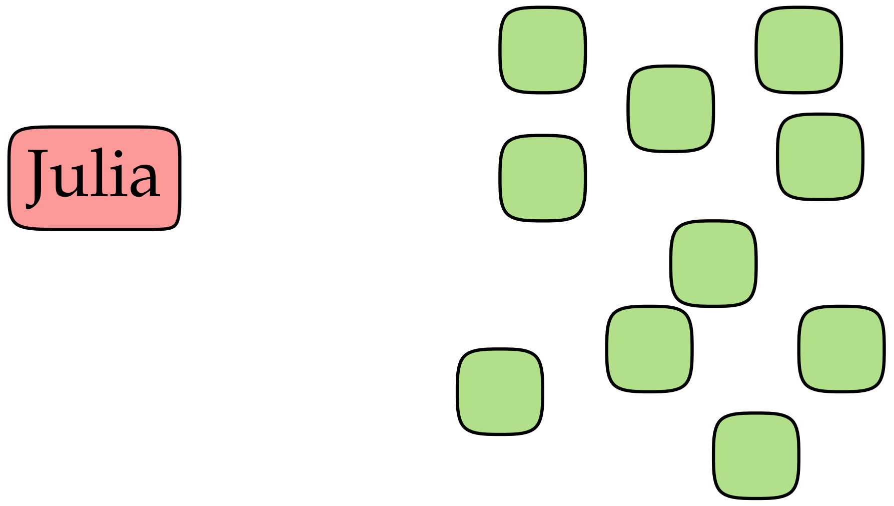
Worst Case

Teile in Gruppen ein:

Julia

Worst Case

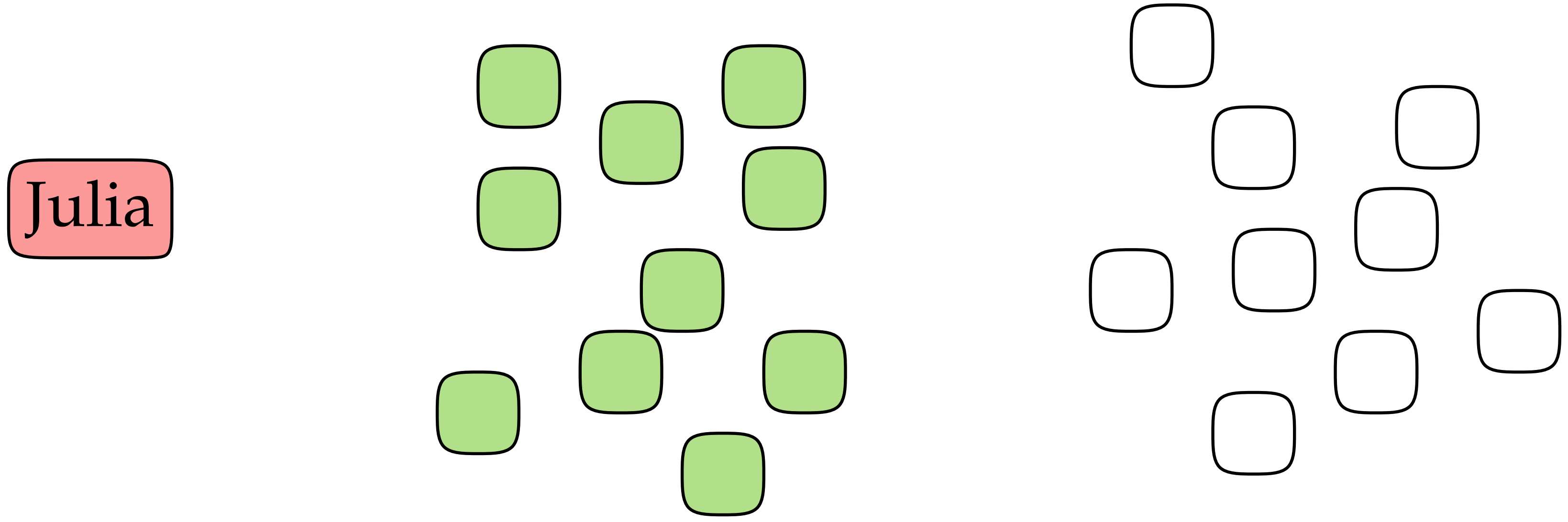
Teile in Gruppen ein:



Meisten Punkte

Worst Case

Teile in Gruppen ein:

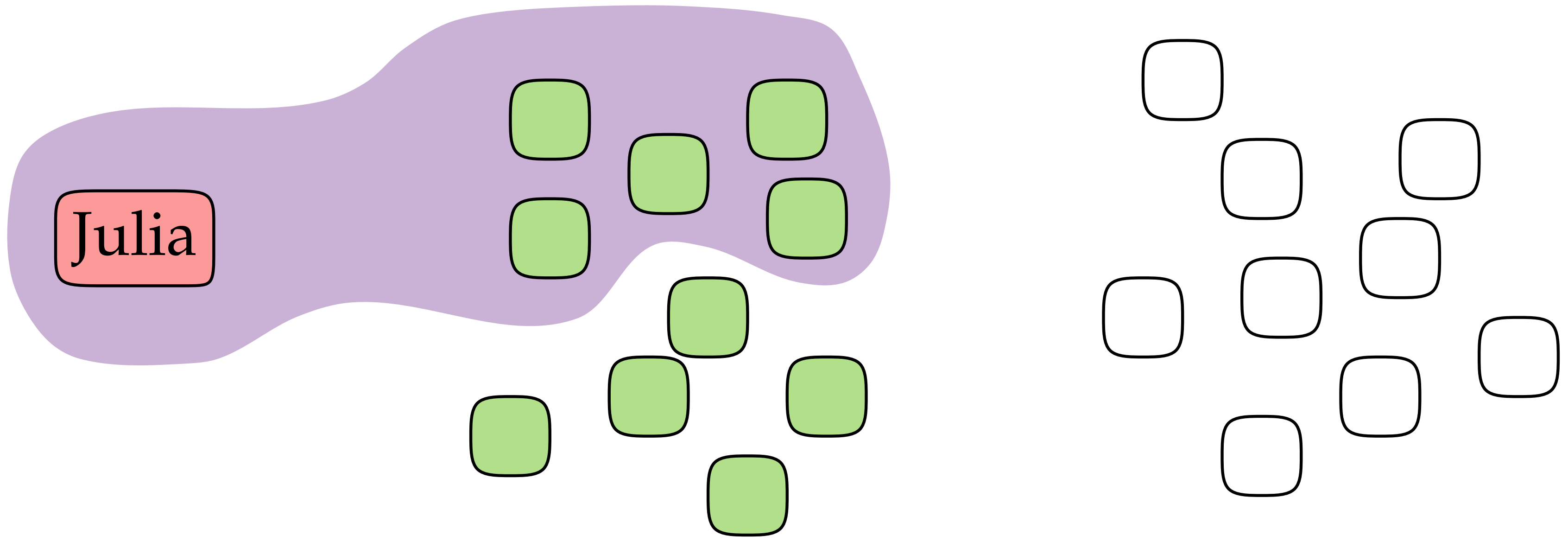


Meisten Punkte

Weniger Punkte

Worst Case

Teile in Gruppen ein:

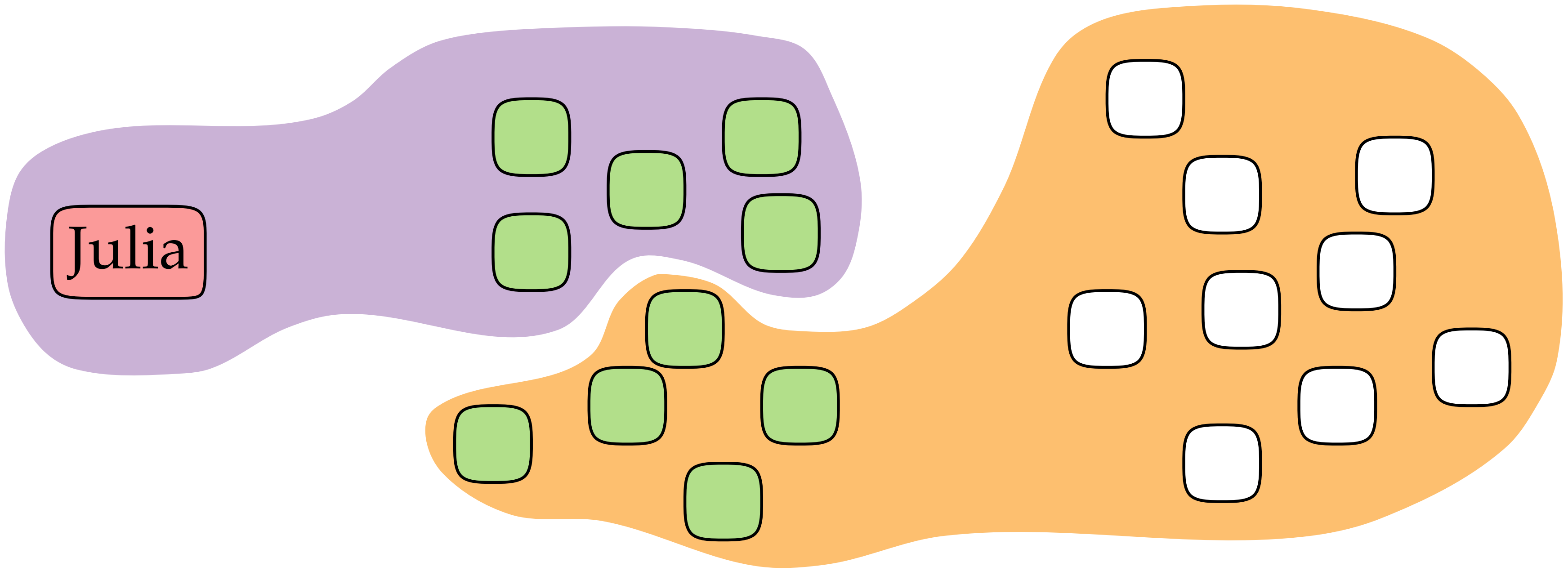


Meisten Punkte

Weniger Punkte

Worst Case

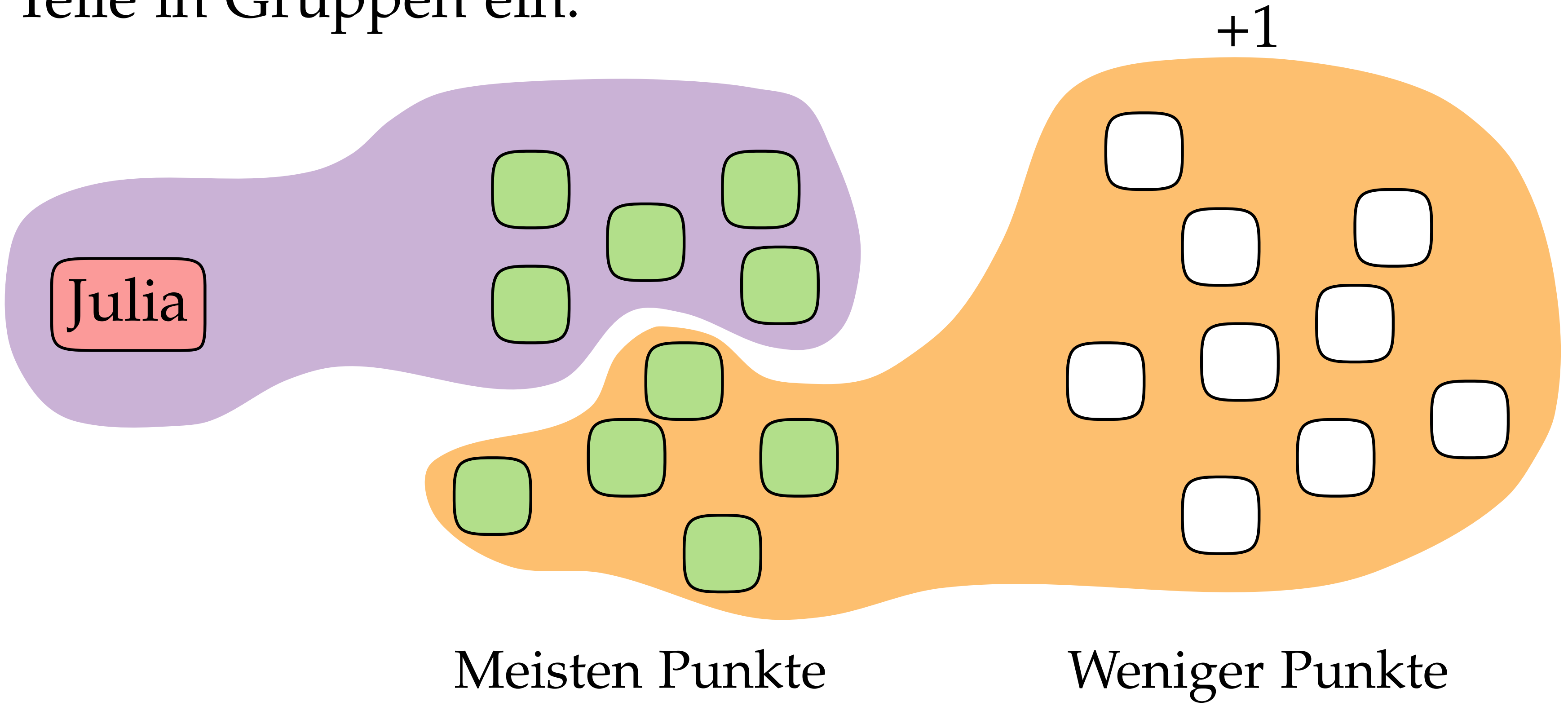
Teile in Gruppen ein:



Meisten Punkte Weniger Punkte

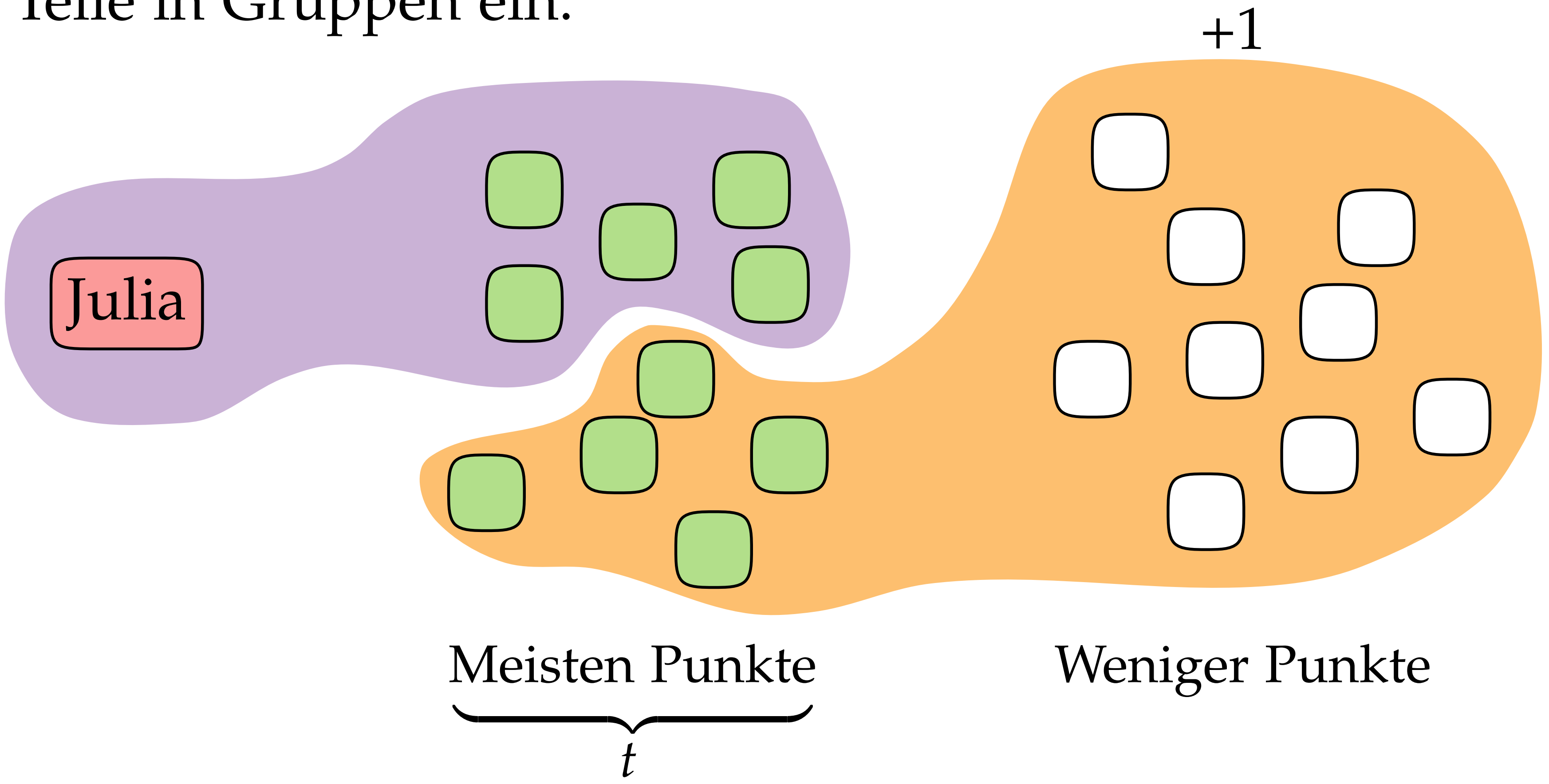
Worst Case

Teile in Gruppen ein:



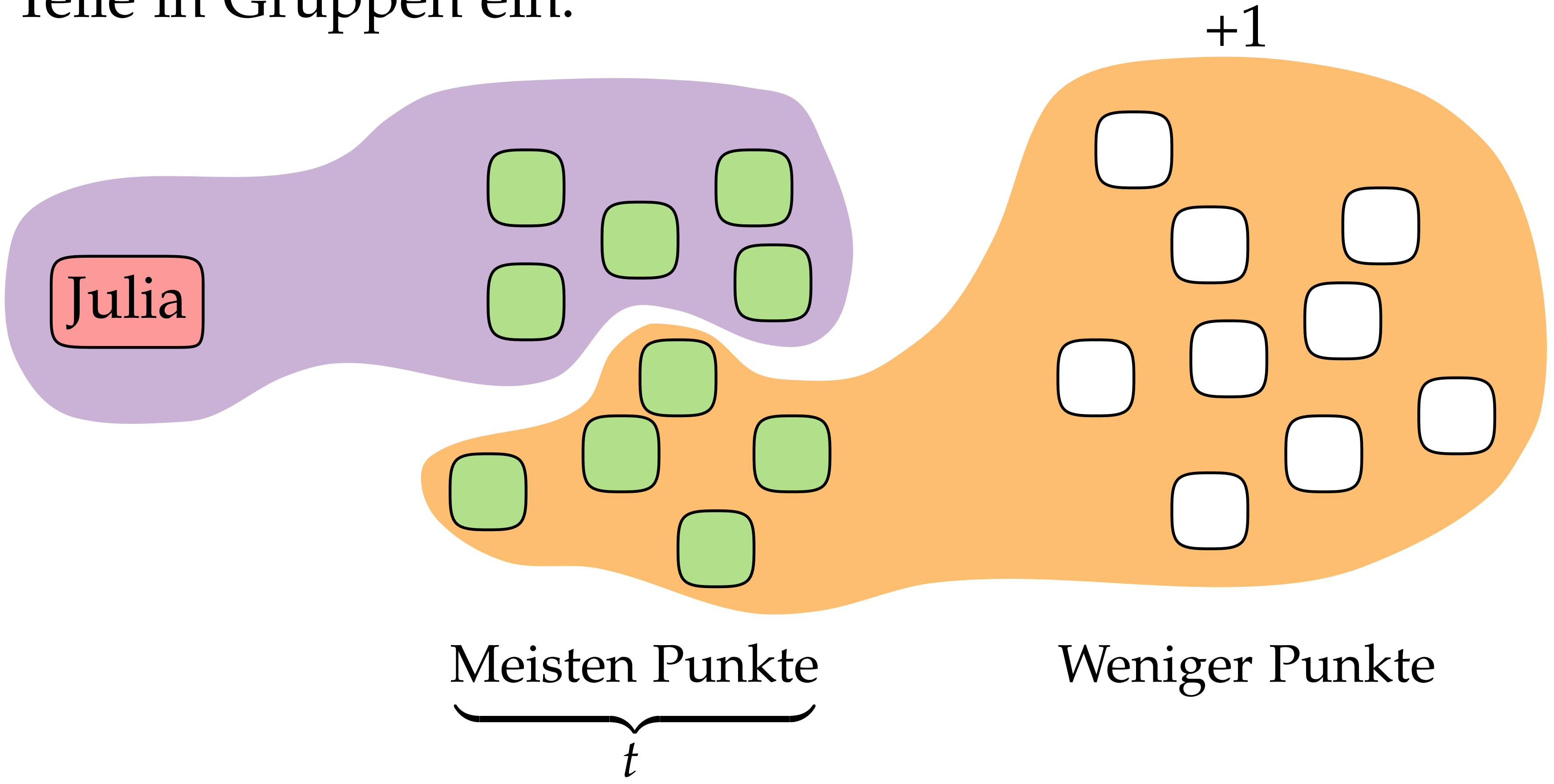
Worst Case

Teile in Gruppen ein:



Worst Case

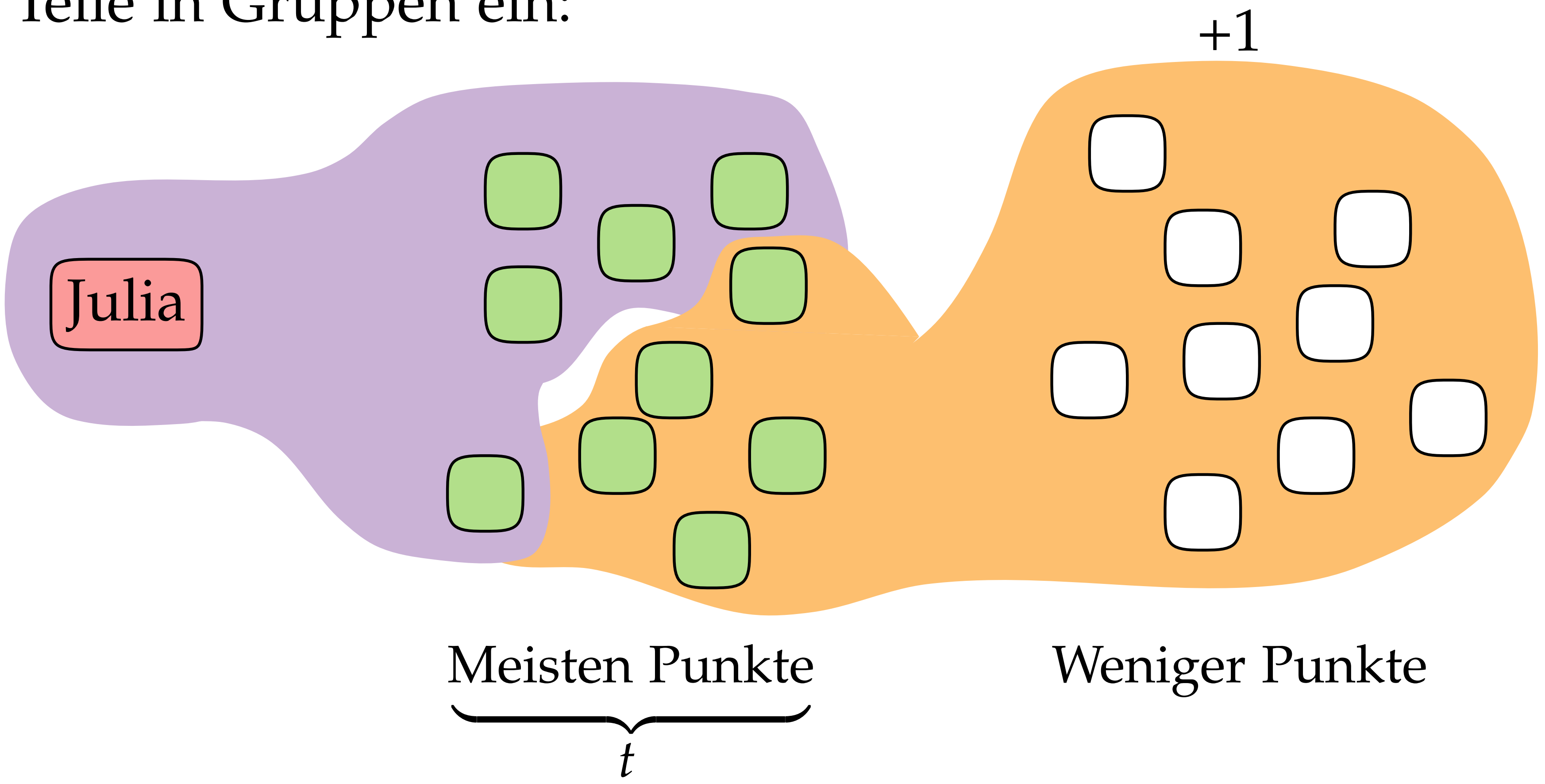
Teile in Gruppen ein:



- Worst Case: Tippen genau wie $\lceil t/2 \rceil$ Verfolger

Worst Case

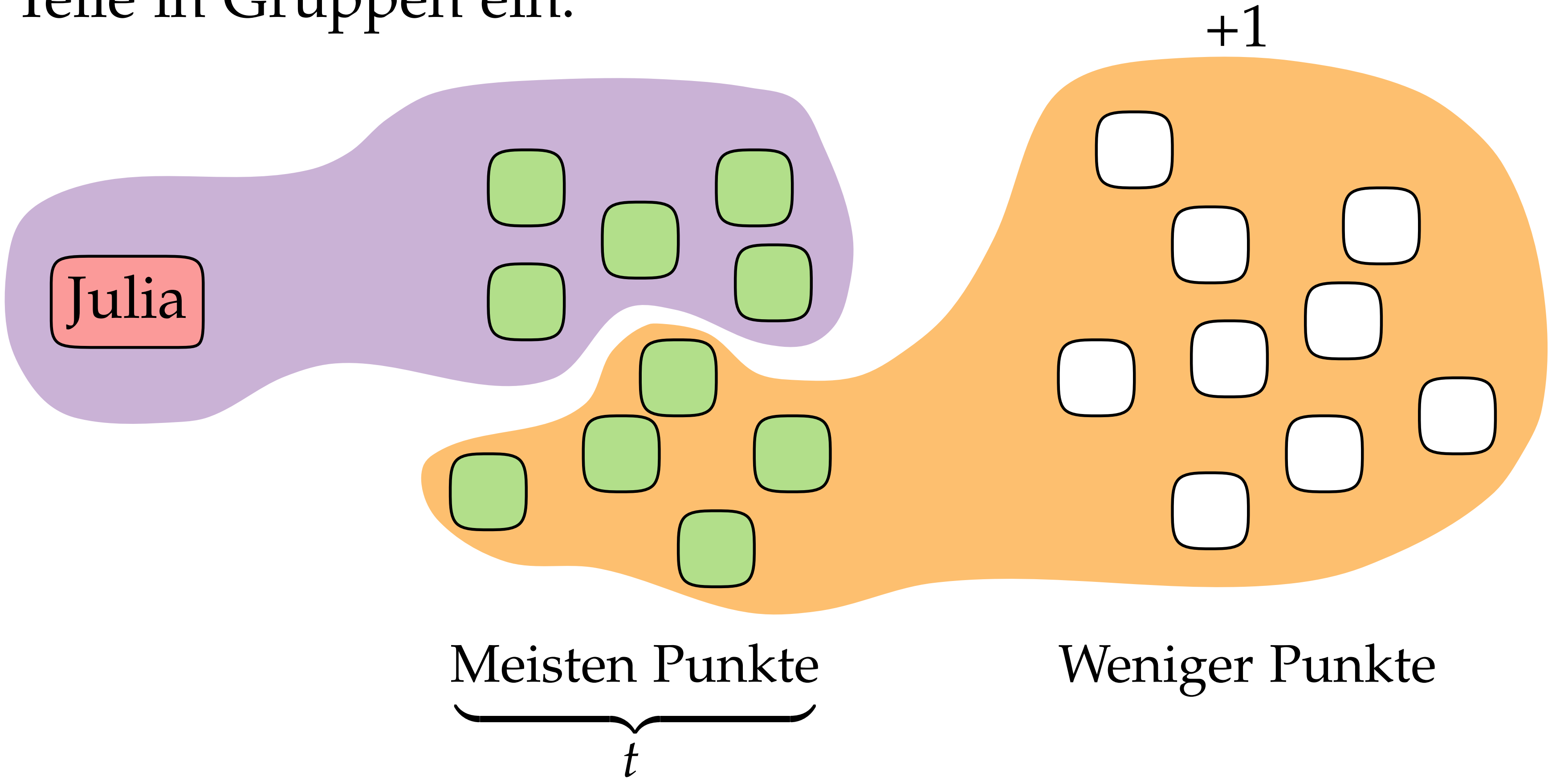
Teile in Gruppen ein:



- Worst Case: Tippen genau wie $\lceil t/2 \rceil$ Verfolger

Worst Case

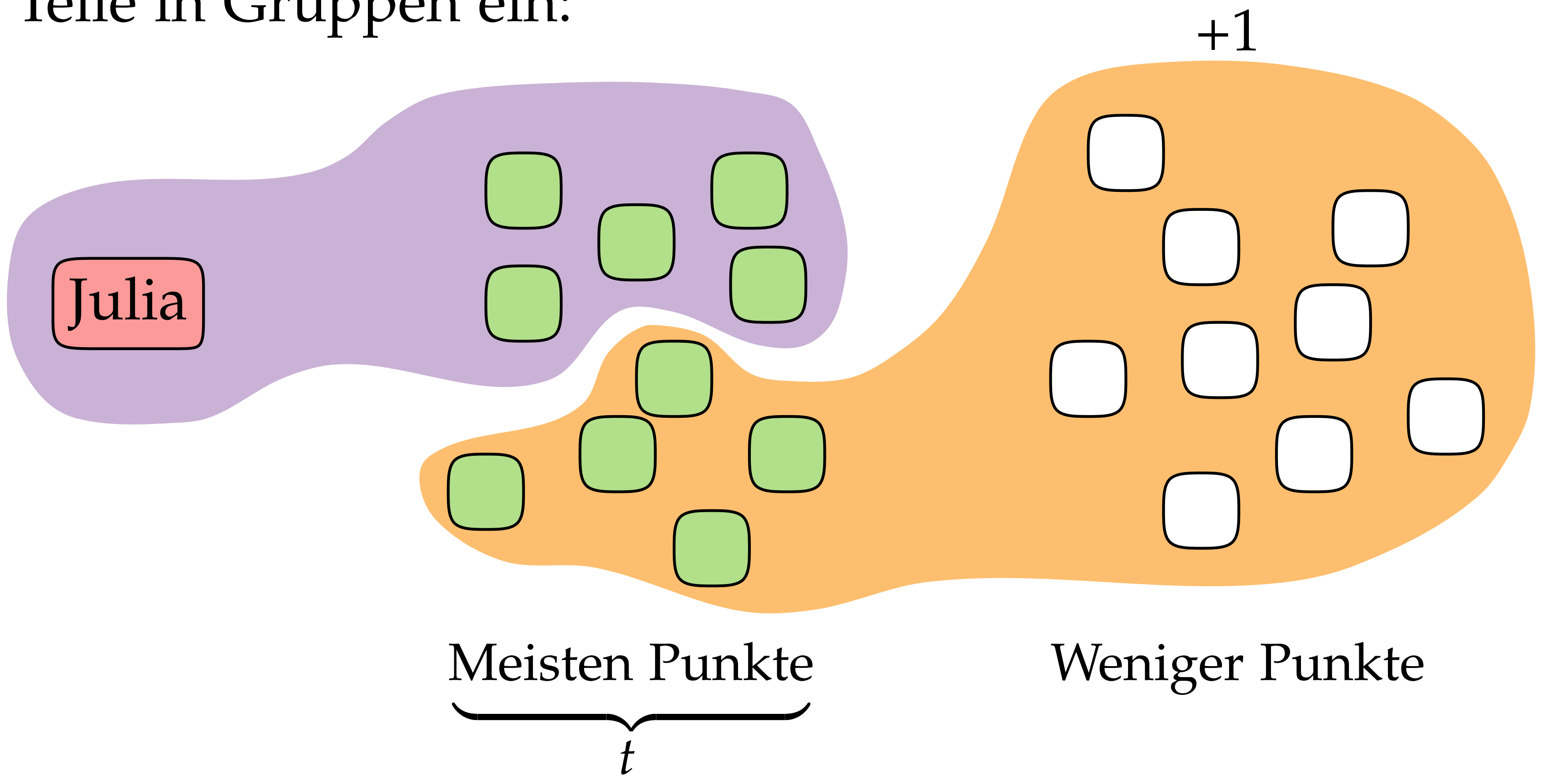
Teile in Gruppen ein:



- Worst Case: Tippen genau wie $\lceil t/2 \rceil$ Verfolger

Worst Case

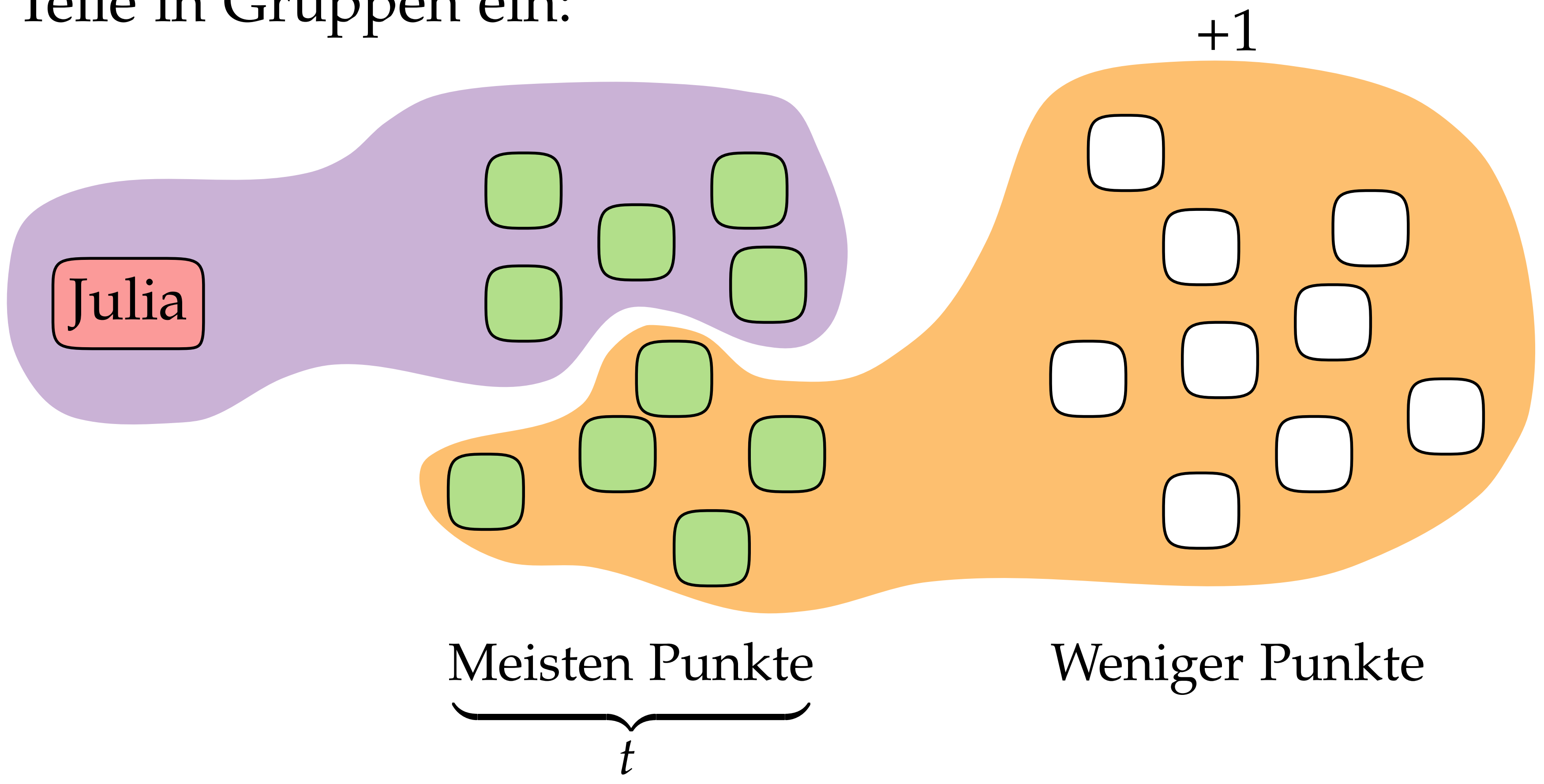
Teile in Gruppen ein:



- Worst Case: Tippen genau wie $\lceil t/2 \rceil$ Verfolger
- Teilnehmer mit gleicher Punktzahl äquivalent

Worst Case

Teile in Gruppen ein:



- Worst Case: Tippen genau wie $\lceil t/2 \rceil$ Verfolger
- Teilnehmer mit gleicher Punktzahl äquivalent

$\Rightarrow$ eindeutige Wahl pro Runde

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand
```

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand
```

J	A	B	C	D	E	F
5	1	4	4	3	2	4

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand
```

J
5

A	B	C	D	E	F
1	4	4	3	2	4

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand
```

J
5

A	B	C	D	E	F
4	1	1	2	3	1

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

Naiver Algorithmus

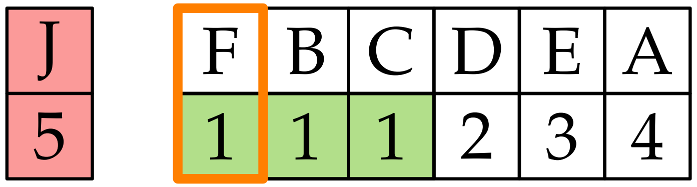
```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird




print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1



print (it - 1)
```



Naiver Algorithmus

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1




print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

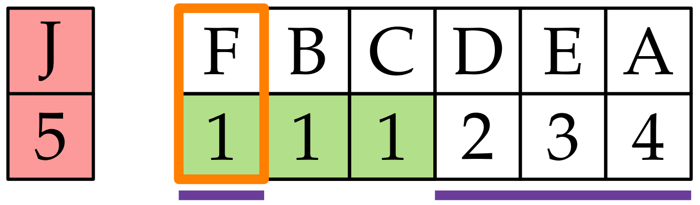
Naiver Algorithmus

J
5

F	B	C	D	E	A
1	1	1	2	3	4

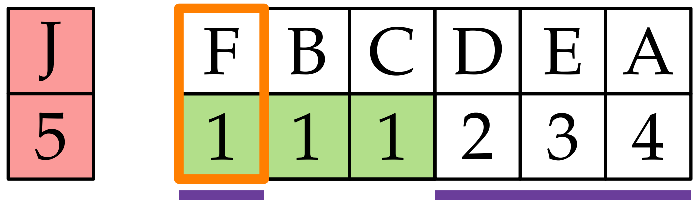
```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher




print (it - 1)
```

Naiver Algorithmus



```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher

    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])
print (it - 1)
```

Naiver Algorithmus

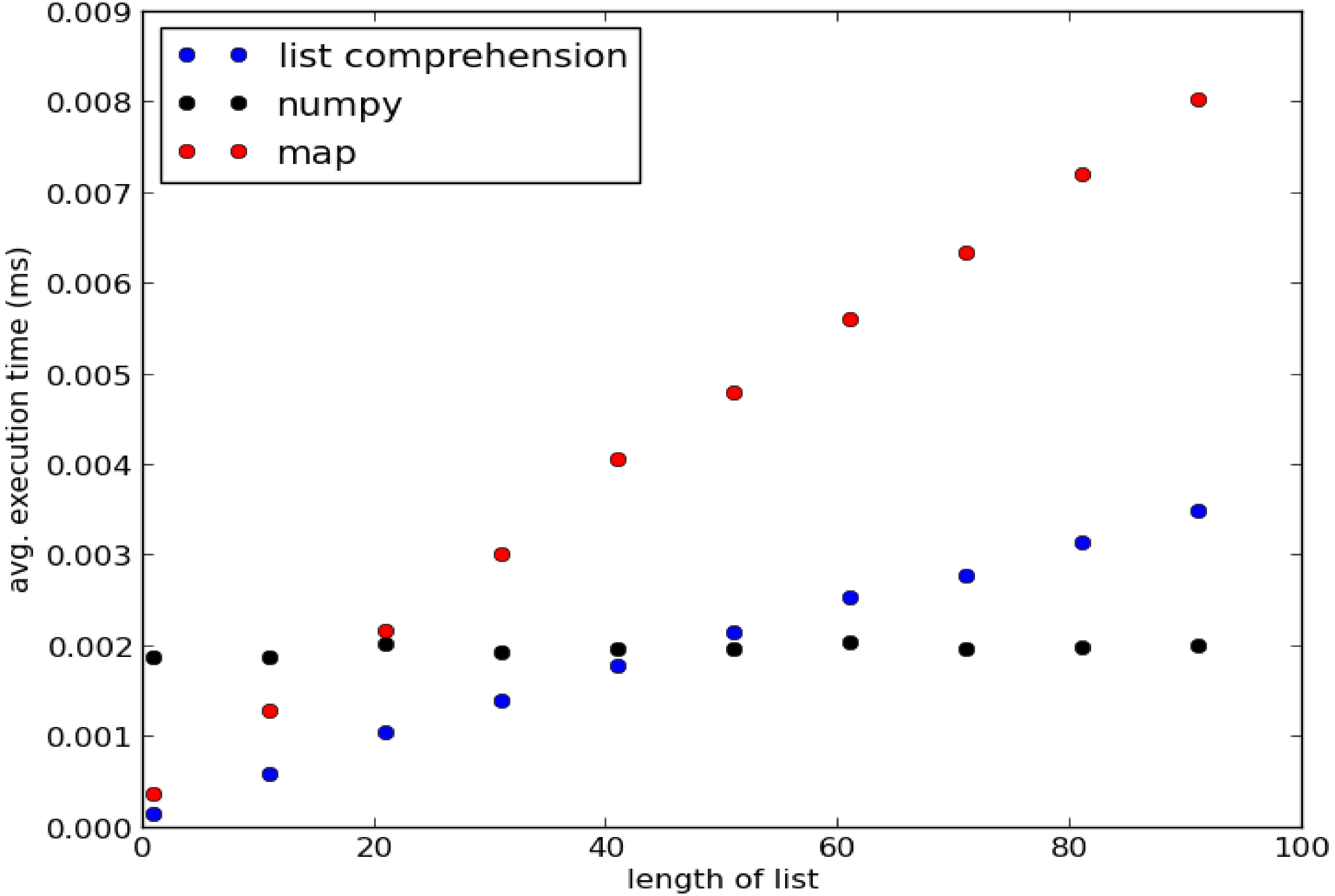


```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher
```

list comprehension

```
    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])

print (it - 1)
```

Python: Listen modifizieren



Naiver Algorithmus: Laufzeit

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher

    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])

print (it - 1)
```

Naiver Algorithmus: Laufzeit

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher

    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])

print (it - 1)
```

$n \leq 10^5$

Naiver Algorithmus: Laufzeit

```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher

    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])

print (it - 1)
```

$r \leq 10^{16}$

$n \leq 10^5$

Naiver Algorithmus: Laufzeit

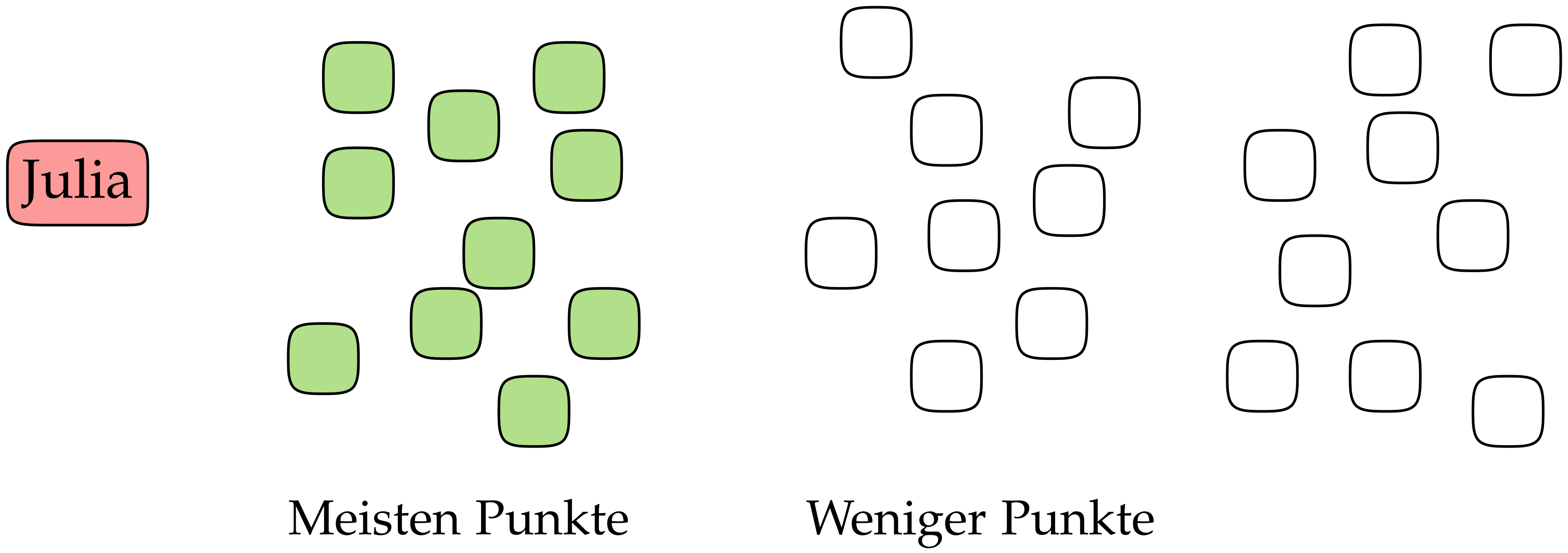
```
n = int(input()) # Python2: int(raw_input())
scores = [int(x) for x in input().split()]
# Python2: map(int, raw_input().split())
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]]) # speichere abstand

it = 0 # wie viele schritte bis julia ueberholt wird

while scores[0] >= 0:
    it += 1

    closest = 1 # wie viele sind am naehesten
    while (closest < n - 1 and scores[closest] == scores[0]):
        closest += 1

    cutoff = int(closest / 2) # so viele direkte verfolger kommen
    naeher

    scores[0:cutoff] = [x - 1 for x in scores[0:cutoff]]
    #scores[0:cutoff] = map(lambda x: x - 1, scores[0:cutoff])
    scores[closest:] = [x - 1 for x in scores[closest:]]
    #scores[closest:] = map(lambda x: x - 1, scores[closest:])
print (it - 1)
```

$r \leq 10^{16}$

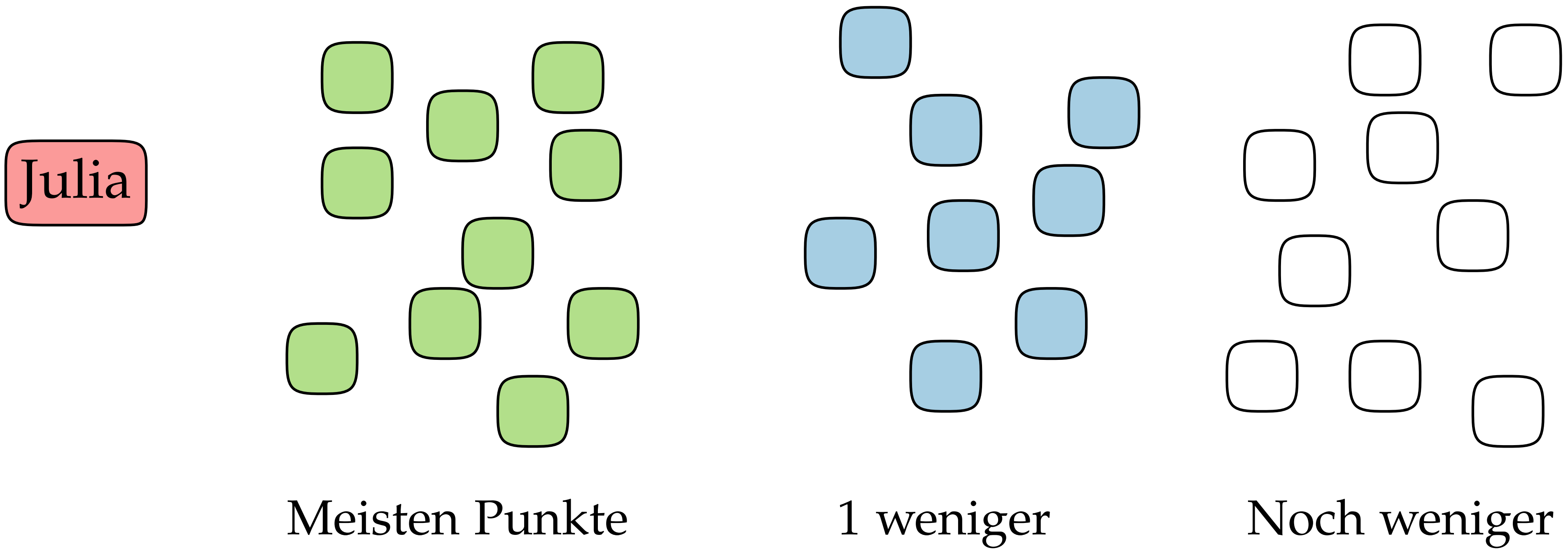
$n \leq 10^5$

$n \cdot r \leq 10^{21}$

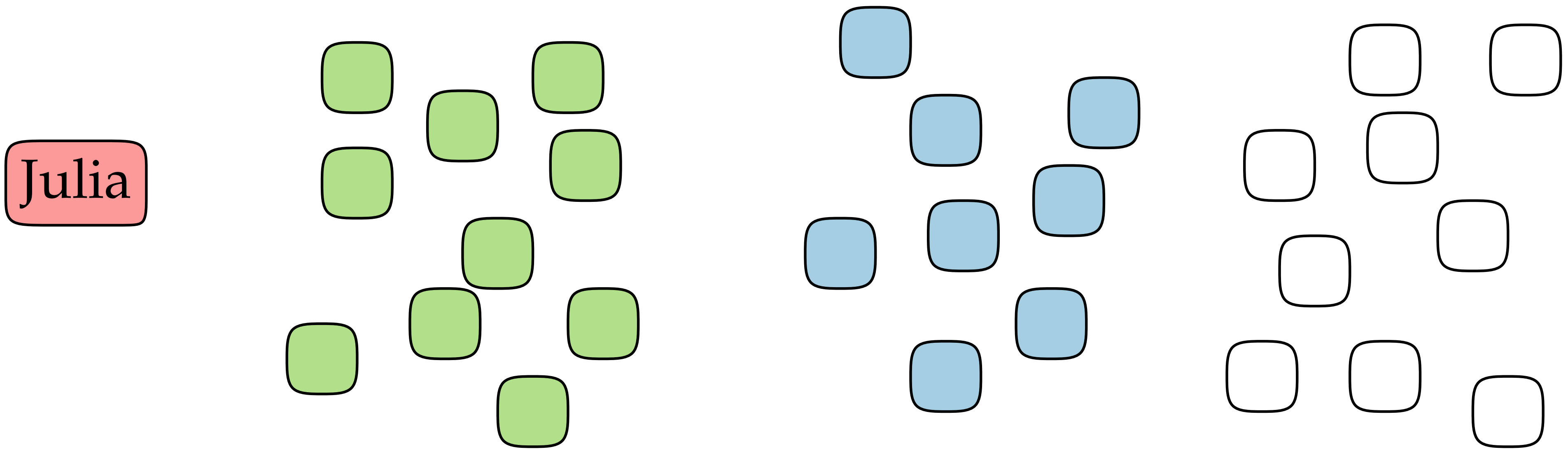
Schneller?



Schneller?

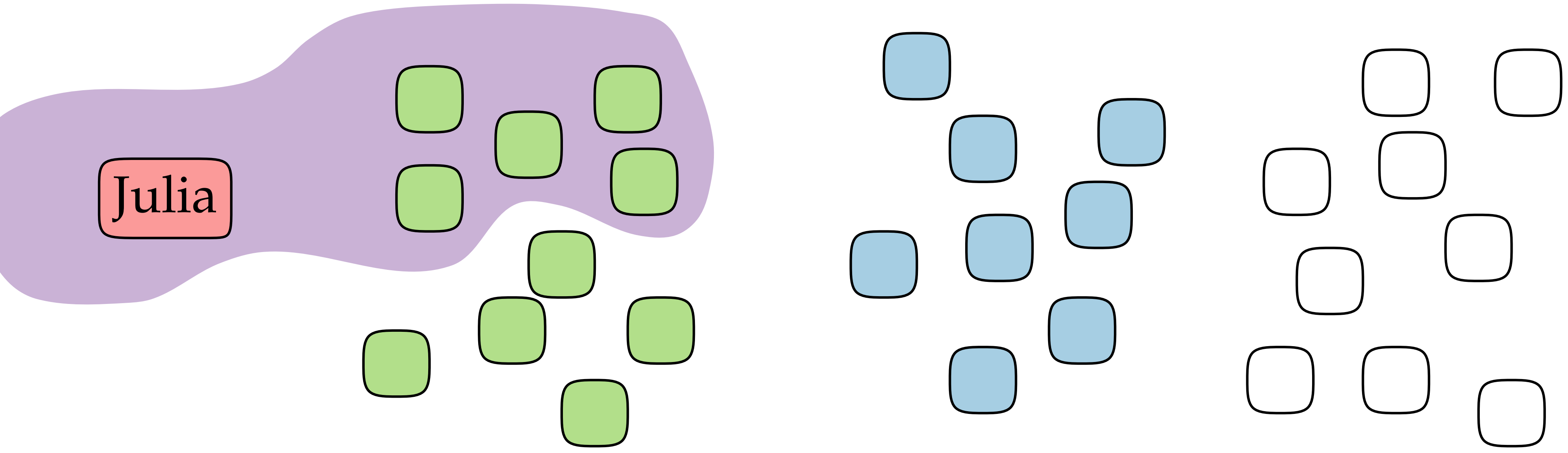


Schneller?



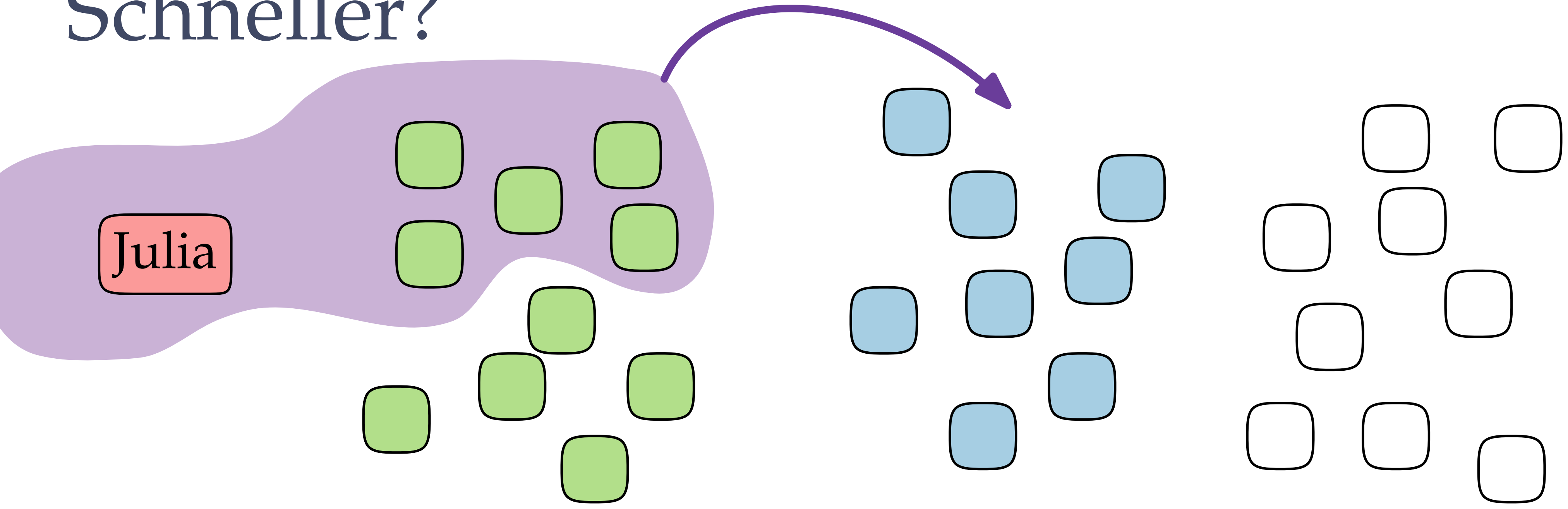
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$

Schneller?



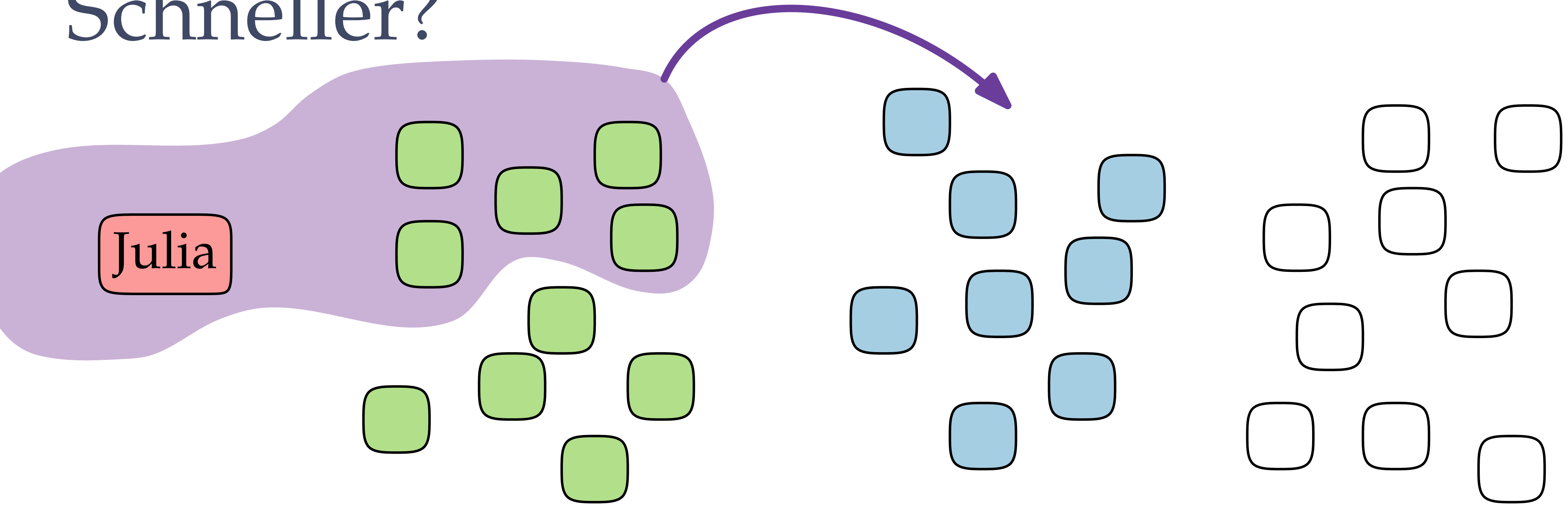
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$

Schneller?



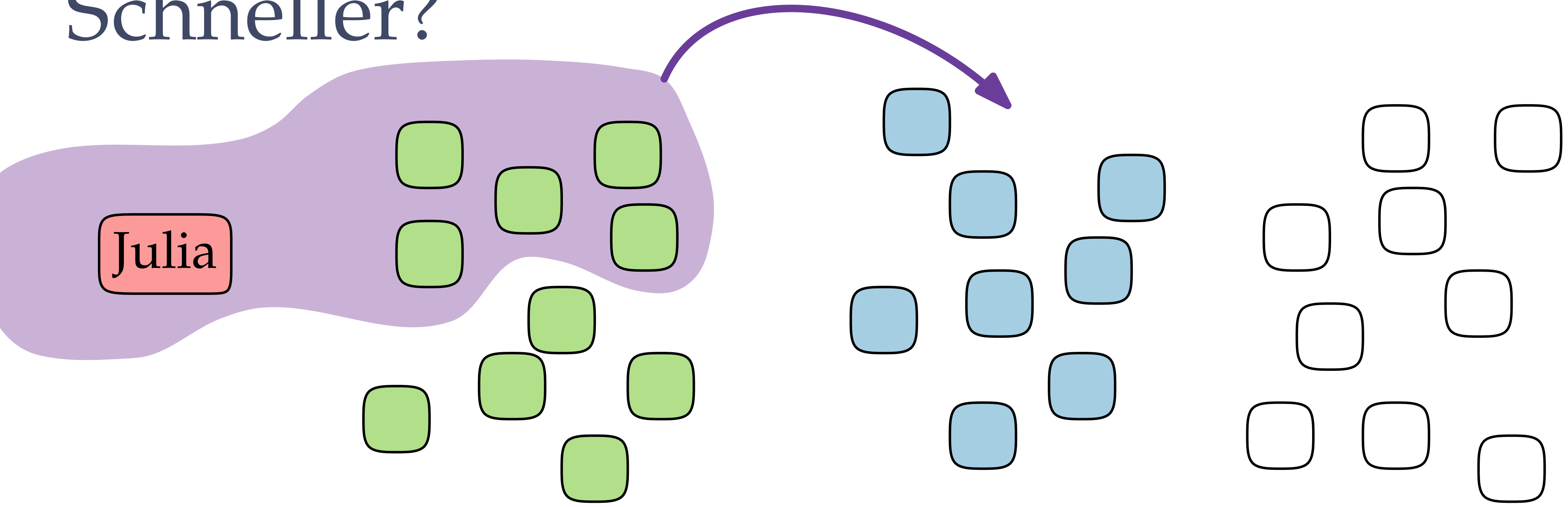
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$

Schneller?

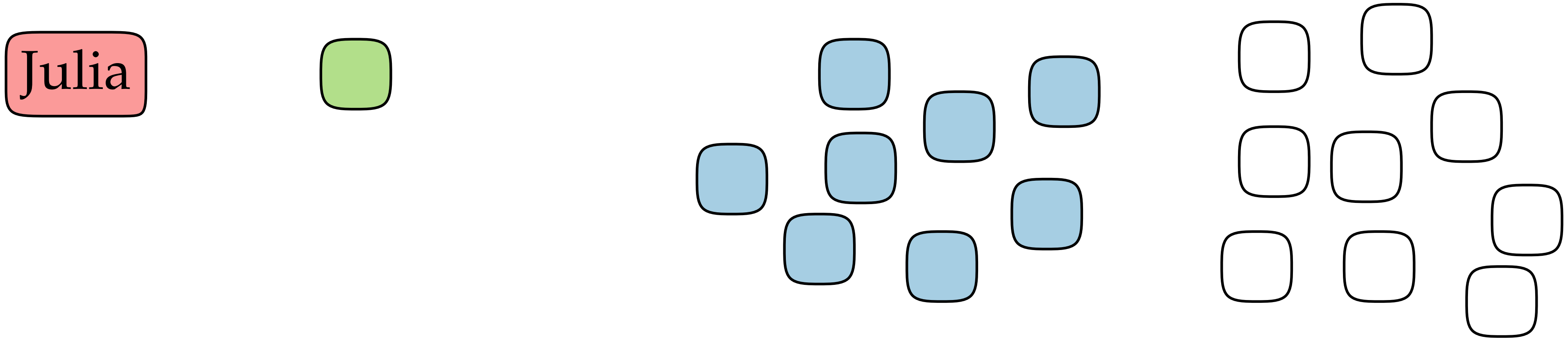


	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$
	$k-1$	k	$> k$

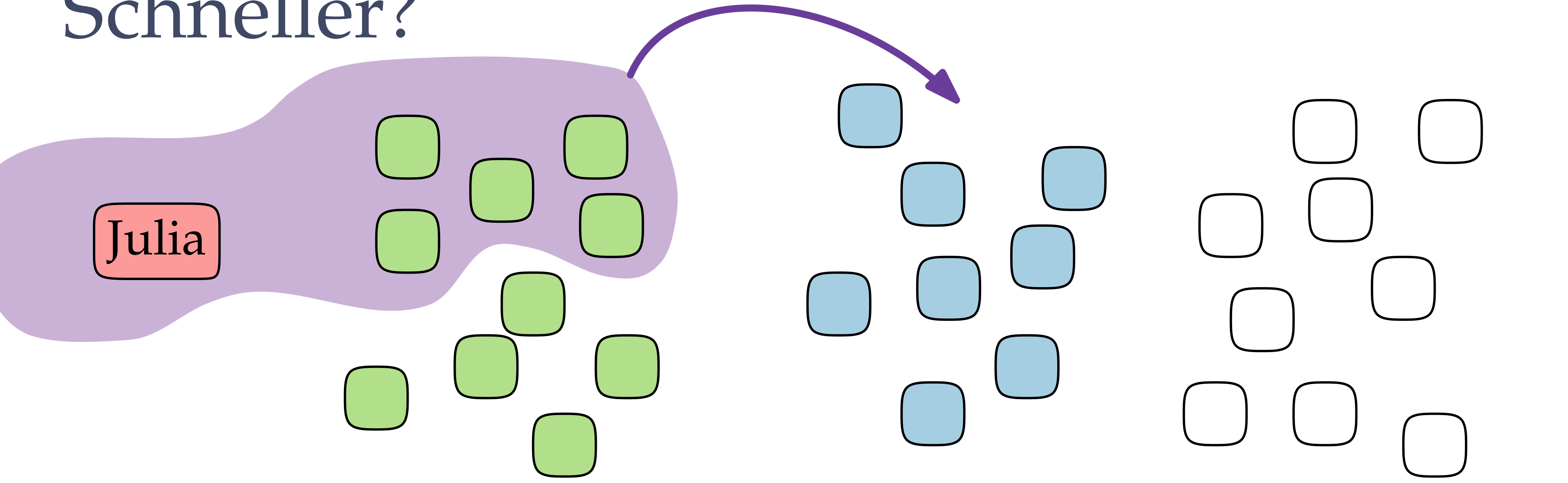
Schneller?



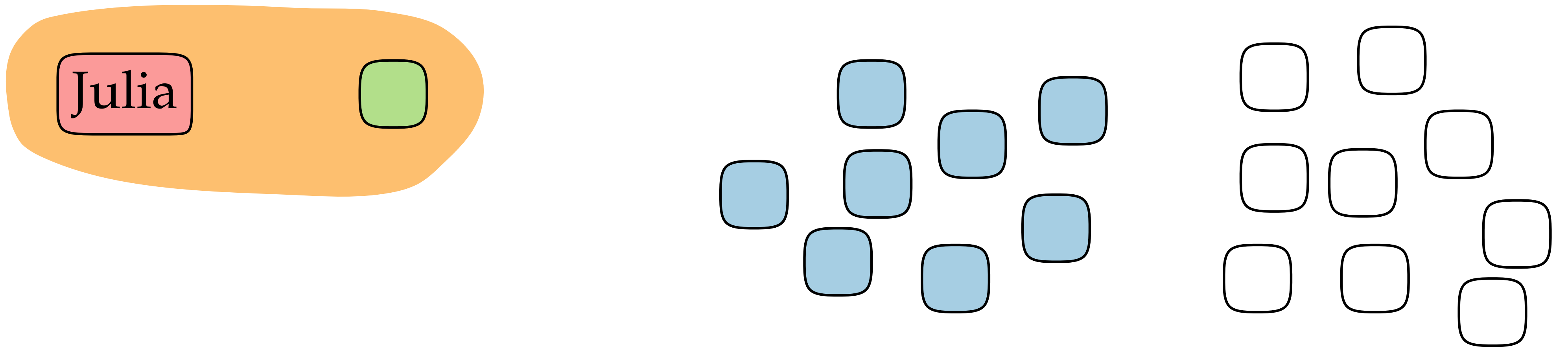
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$
	$k-1$	k	$> k$



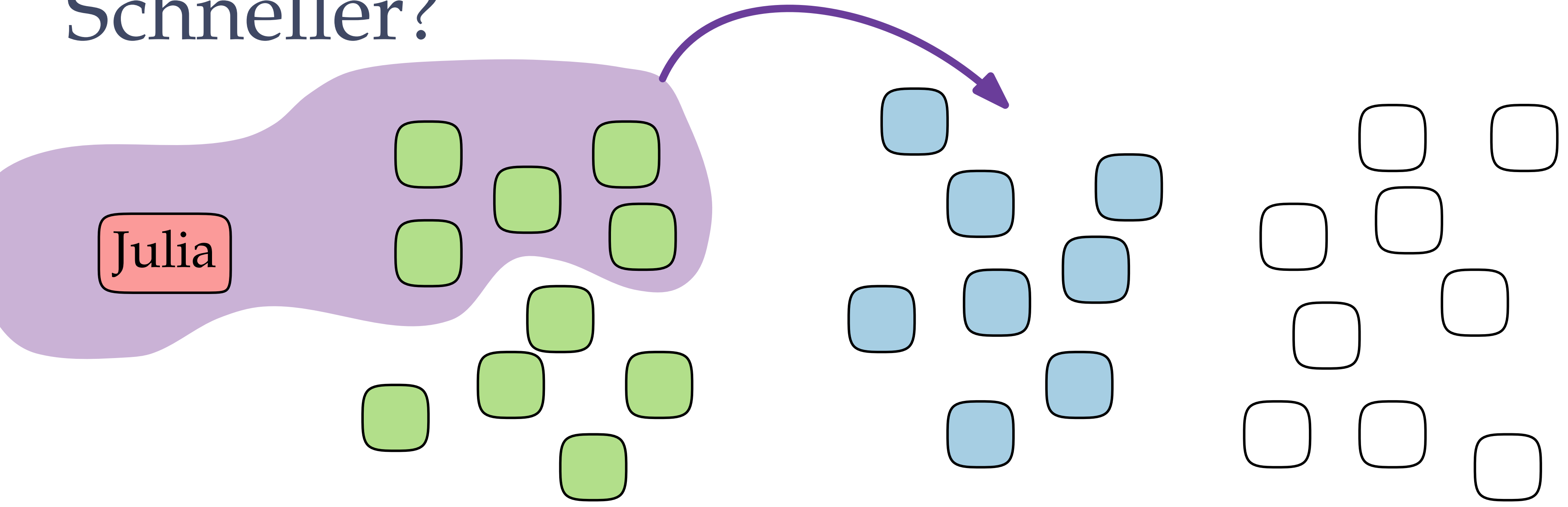
Schneller?



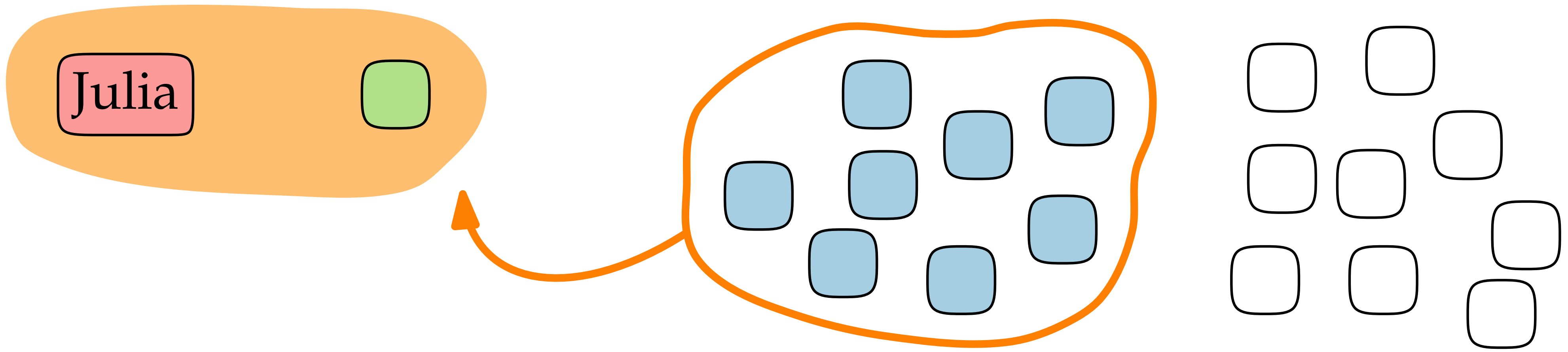
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$>k+1$
	$k-1$	k	$>k$



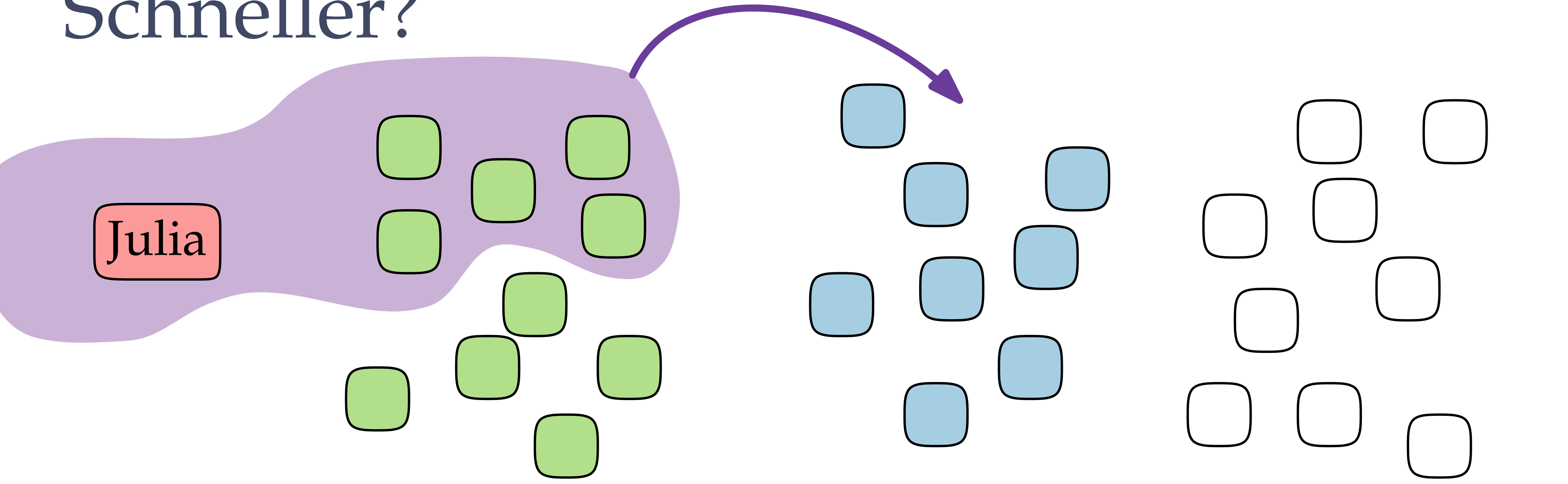
Schneller?



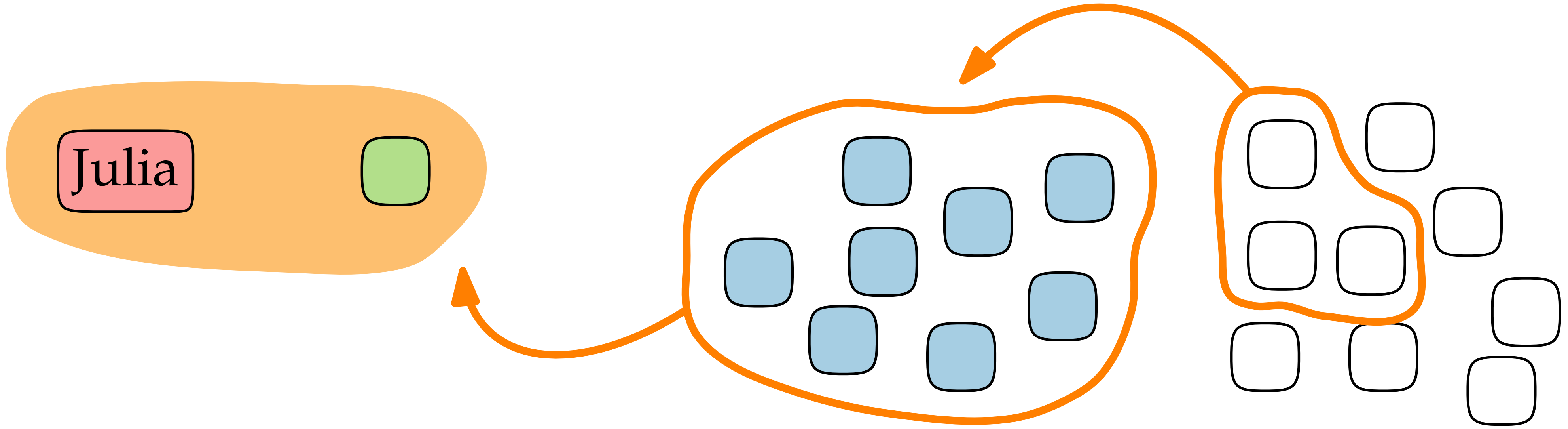
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$>k+1$
	$k-1$	k	$>k$



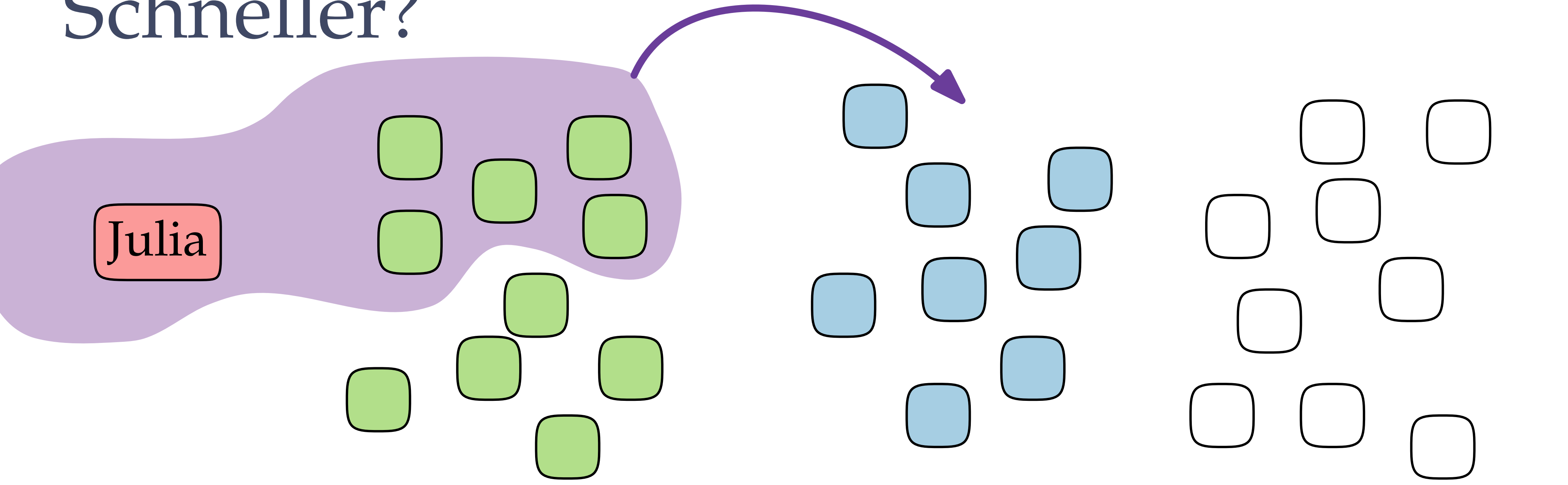
Schneller?



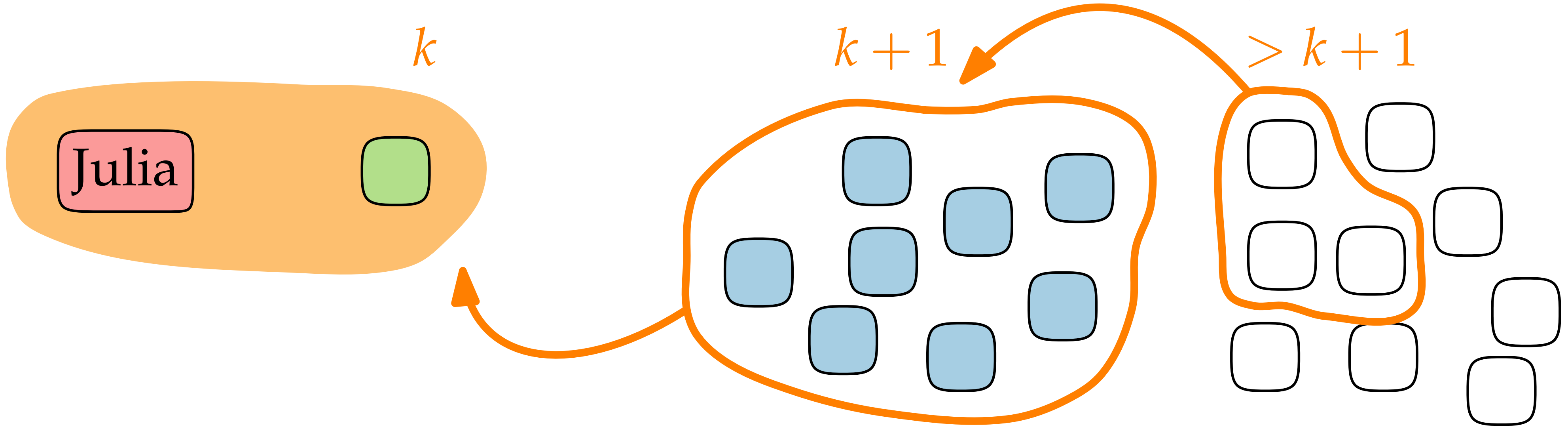
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$
	$k-1$	k	$> k$



Schneller?



	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$
	$k-1$	k	$> k$
	k	$k+1$	$> k+1$



```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
it = 0
while distance >= 0:
    it += 1
```

```
print (it - 1)
```

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1







it = 0
while distance >= 0:
    it += 1











print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1




print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```python
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1



print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

$r \leq 10^{16}$

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

$r \leq 10^{16}$

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

$r \leq 10^{16}$

insgesamt $n \leq 10^5$

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

$O(n \log n)$

$r \leq 10^{16}$

insgesamt $n \leq 10^5$

```
n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]

closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    it += 1
    if closest > 1: # oberer fall
        almost_closest += closest - int(closest / 2)
        closest = int(closest / 2)
        distance -= 1
    else: # unterer fall
        closest += almost_closest
        # berechne almost_closest neu
        almost_closest = 0
        while closest + almost_closest < n - 1 and \
                scores[closest + almost_closest] <= distance + it + 1:
            almost_closest += 1
print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

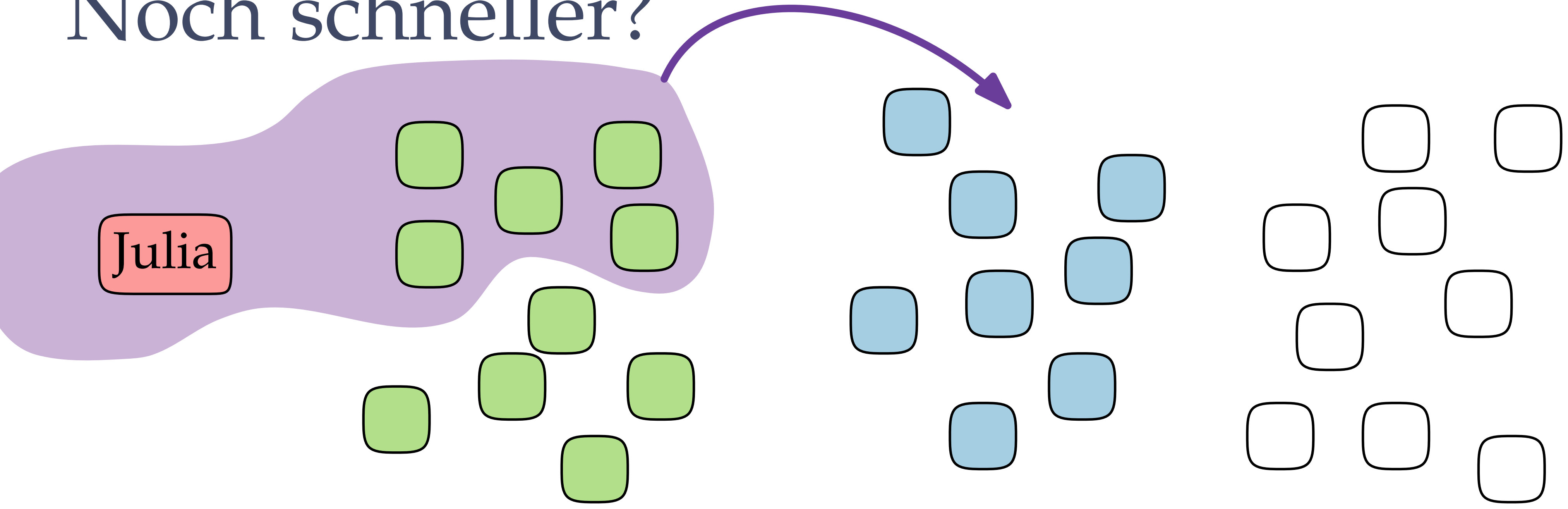
$O(n \log n)$

$r \leq 10^{16}$

Immer noch riesig

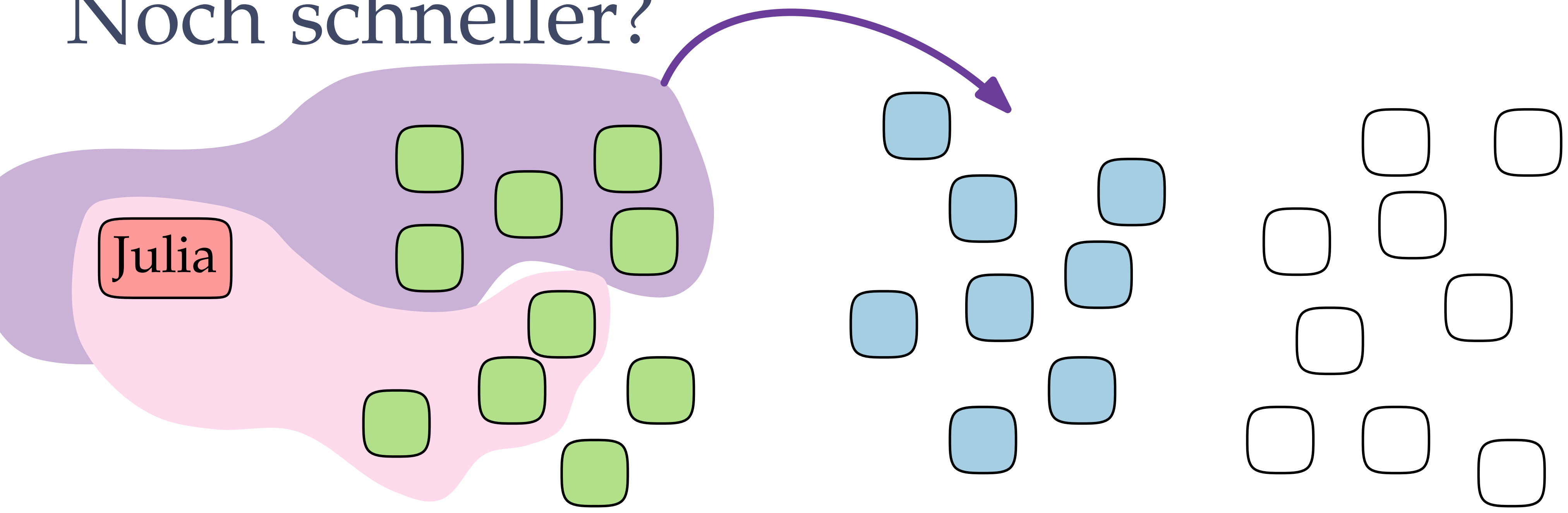
insgesamt $n \leq 10^5$

Noch schneller?



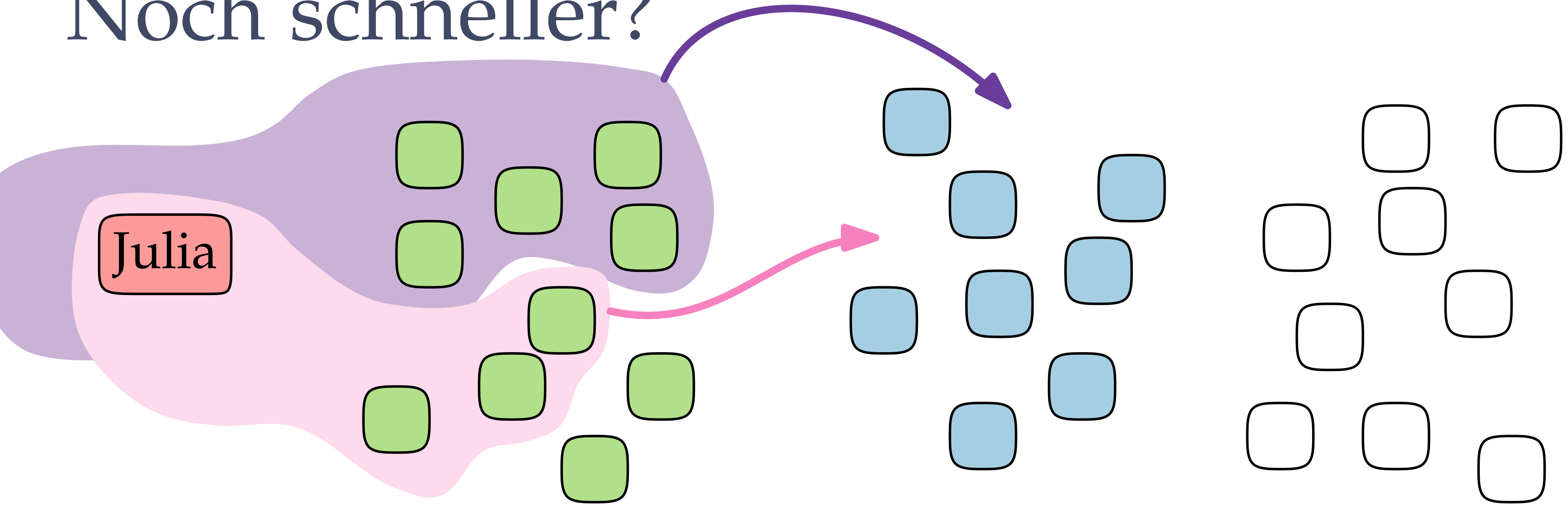
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$

Noch schneller?



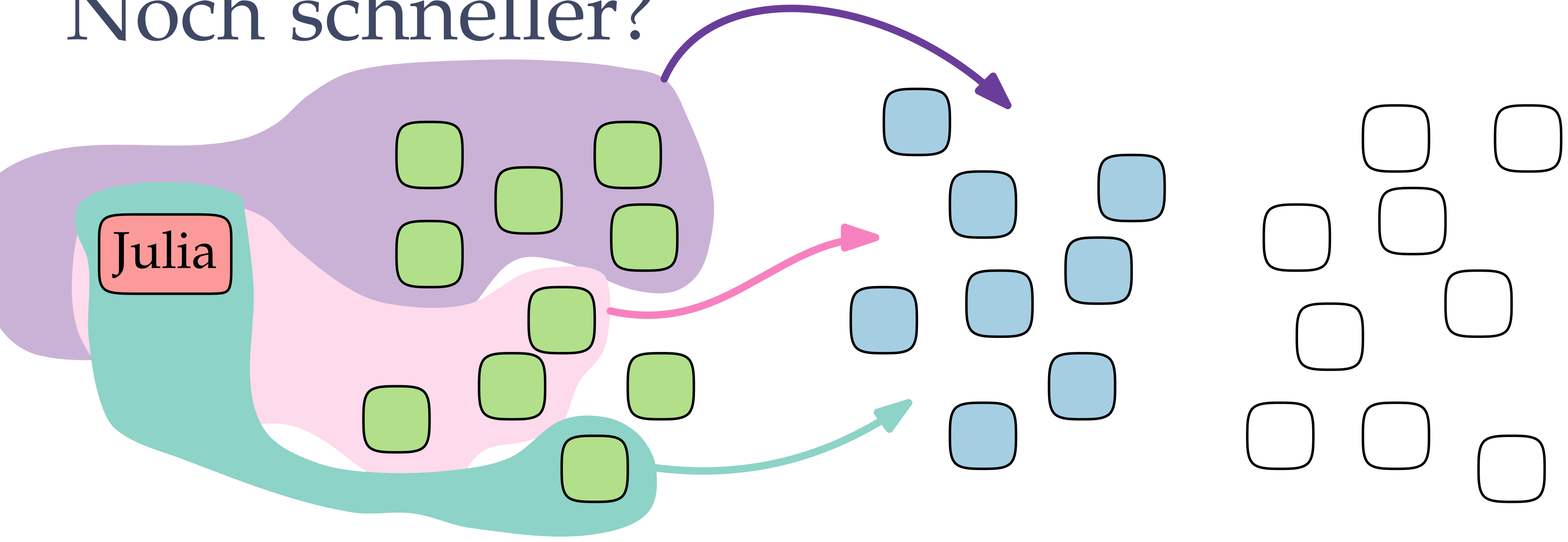
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$>k+1$

Noch schneller?



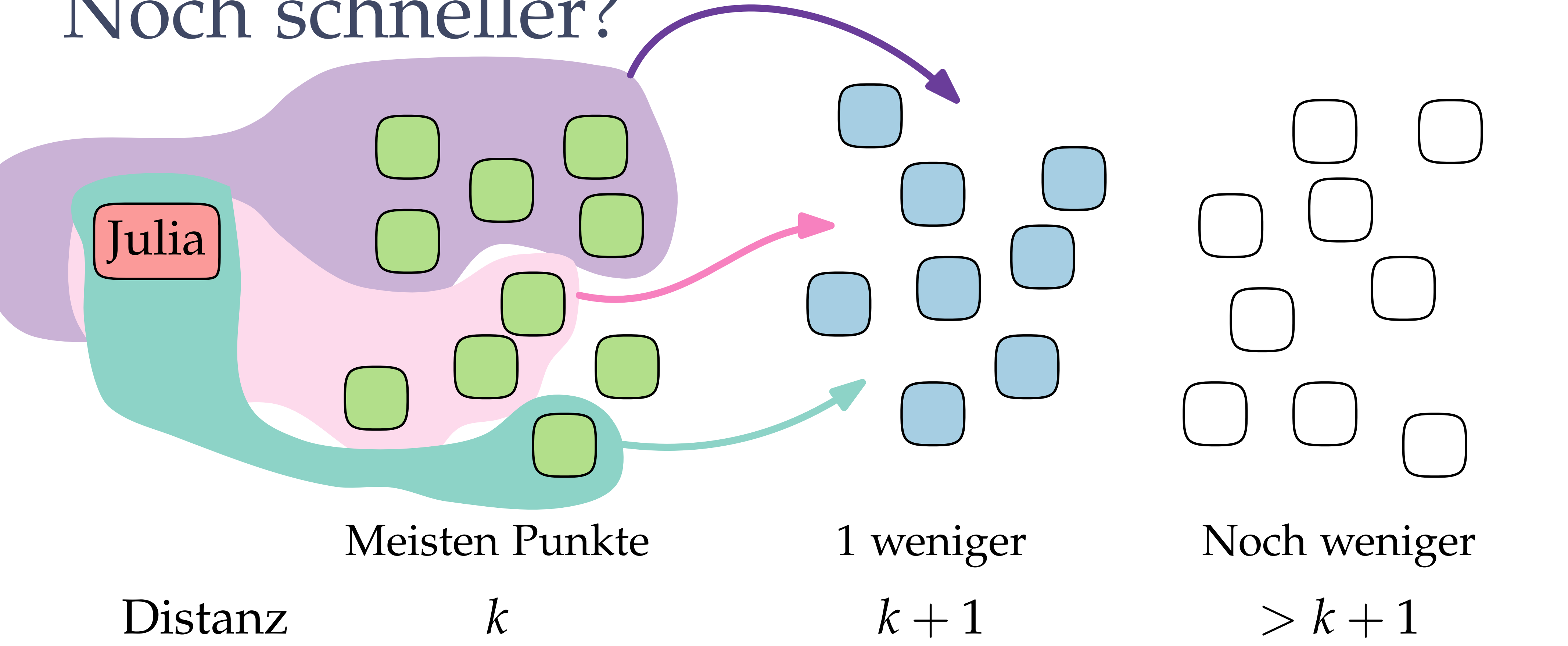
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$>k+1$

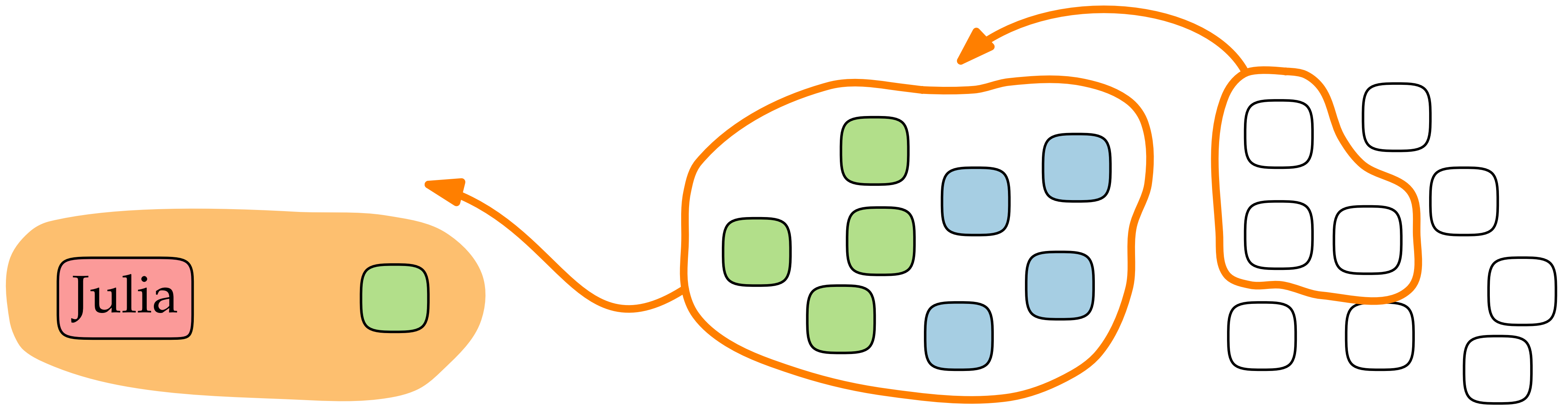
Noch schneller?



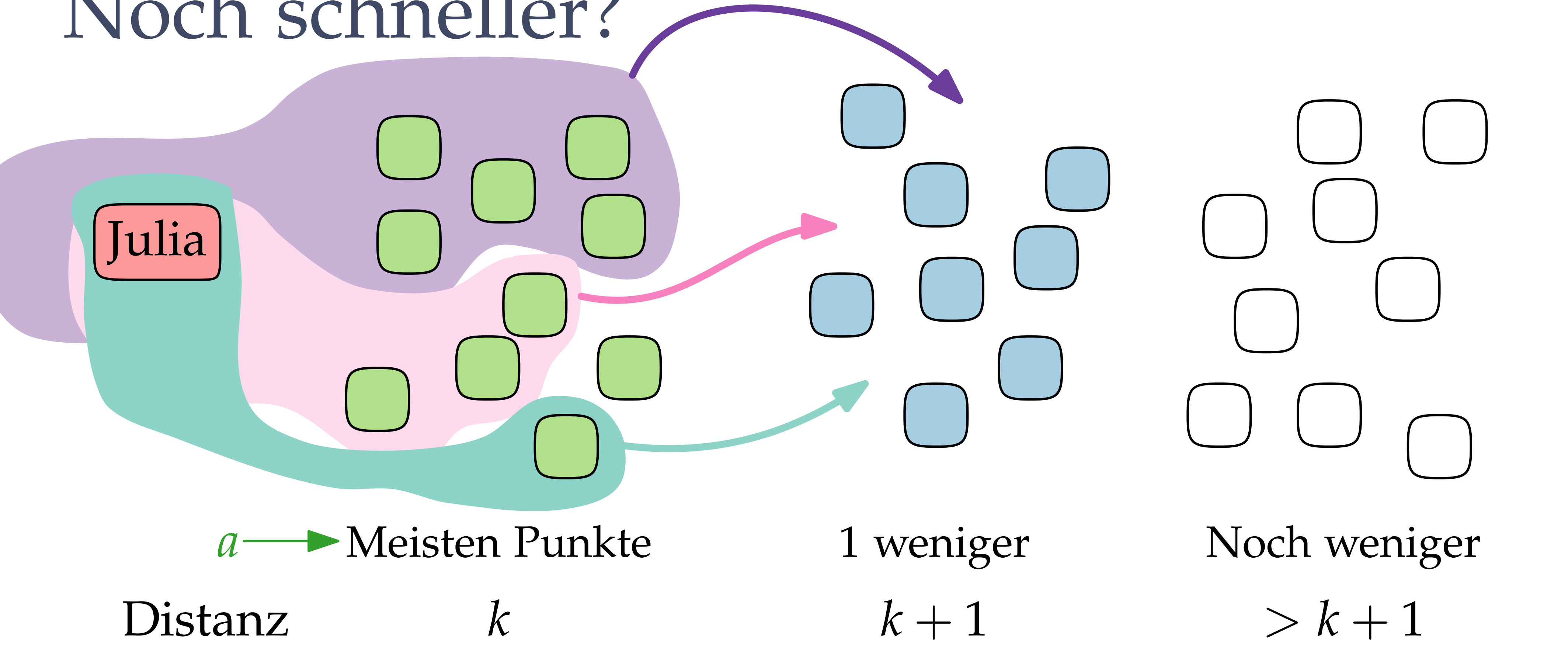
	Meisten Punkte	1 weniger	Noch weniger
Distanz	k	$k+1$	$> k+1$

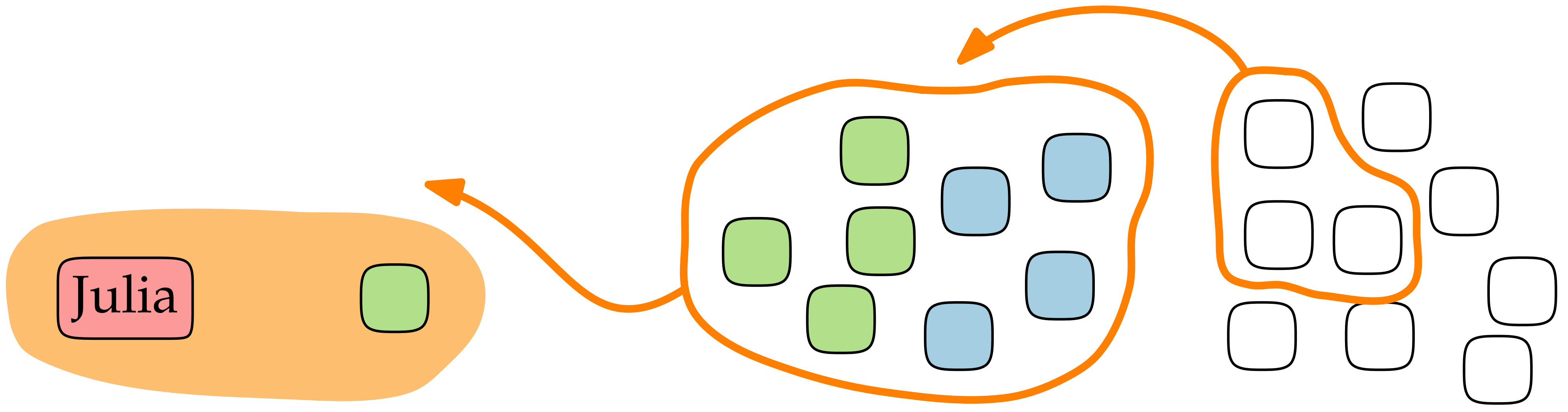
Noch schneller?



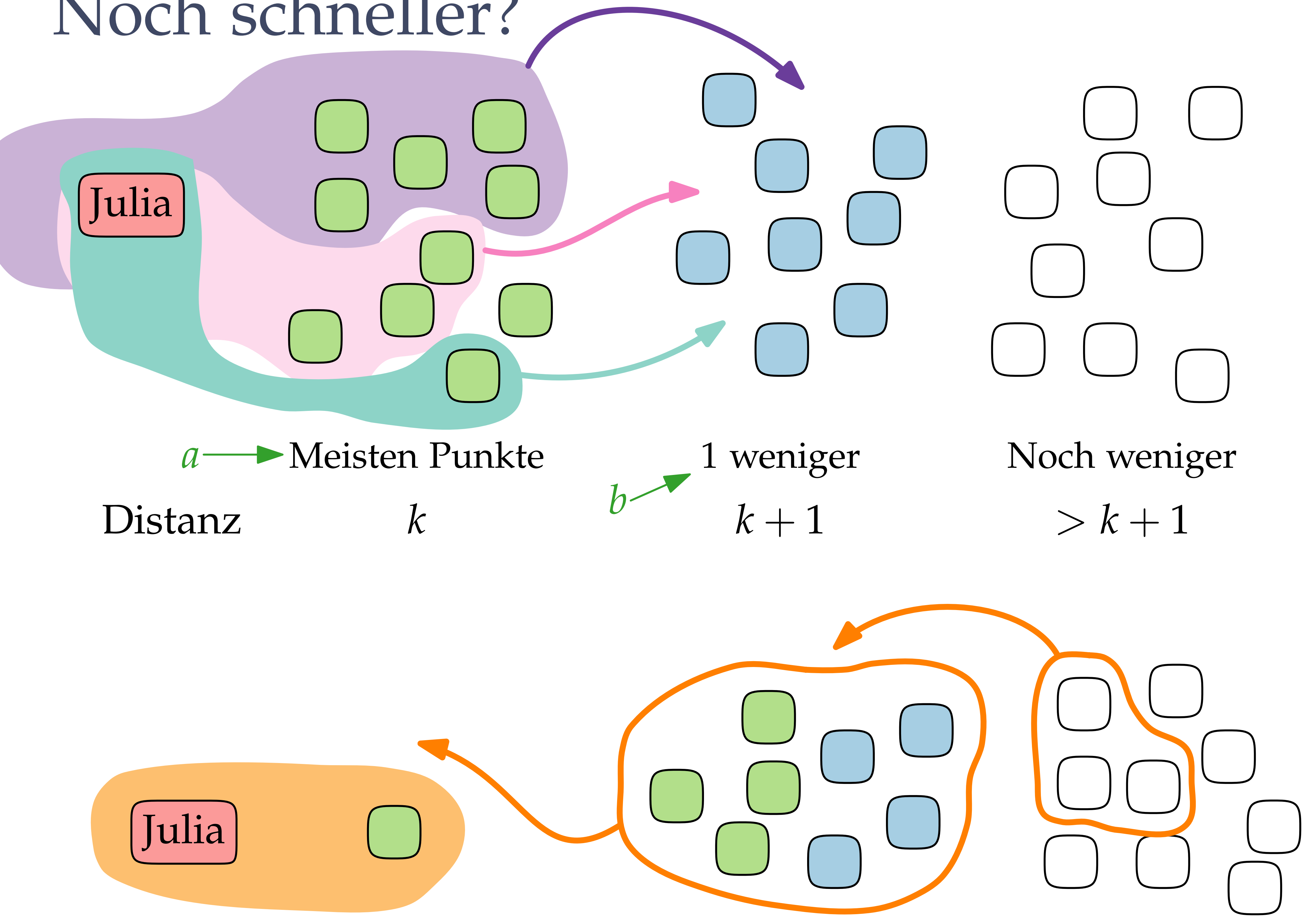


Noch schneller?

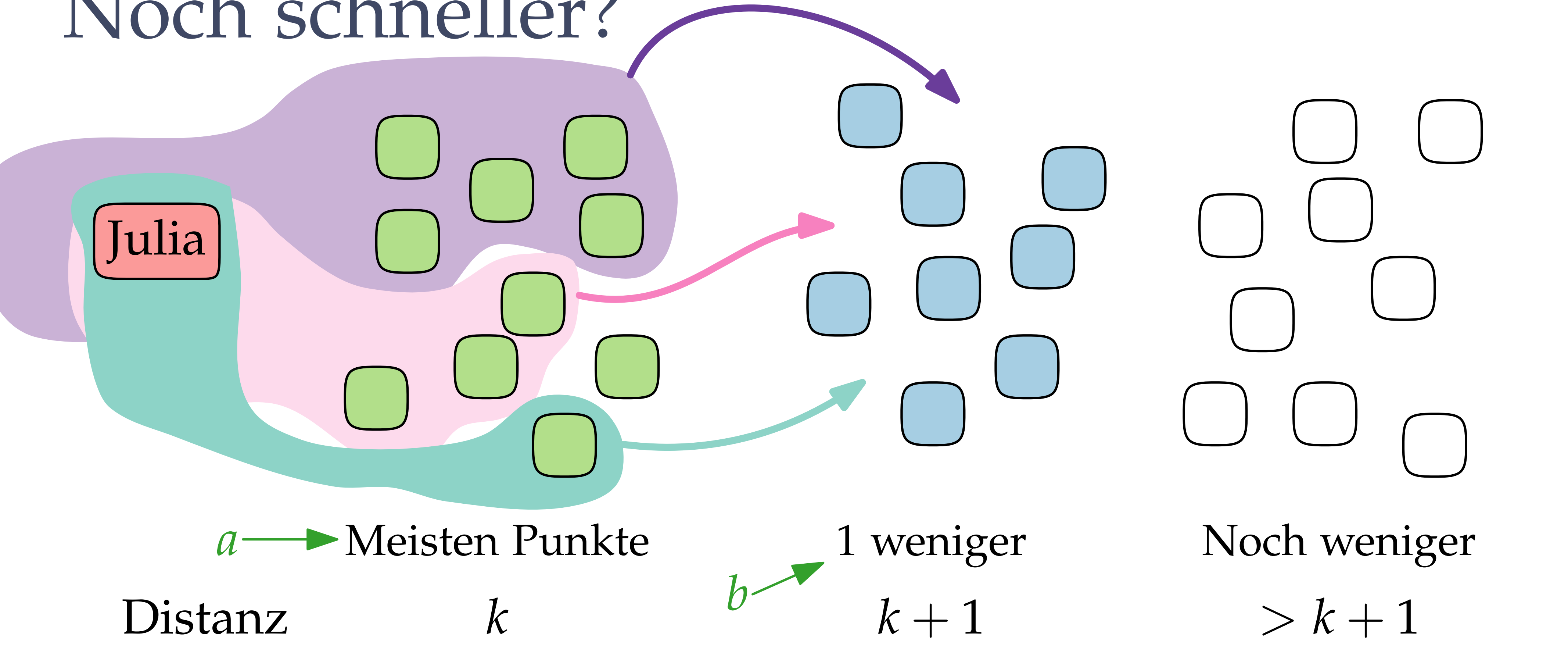




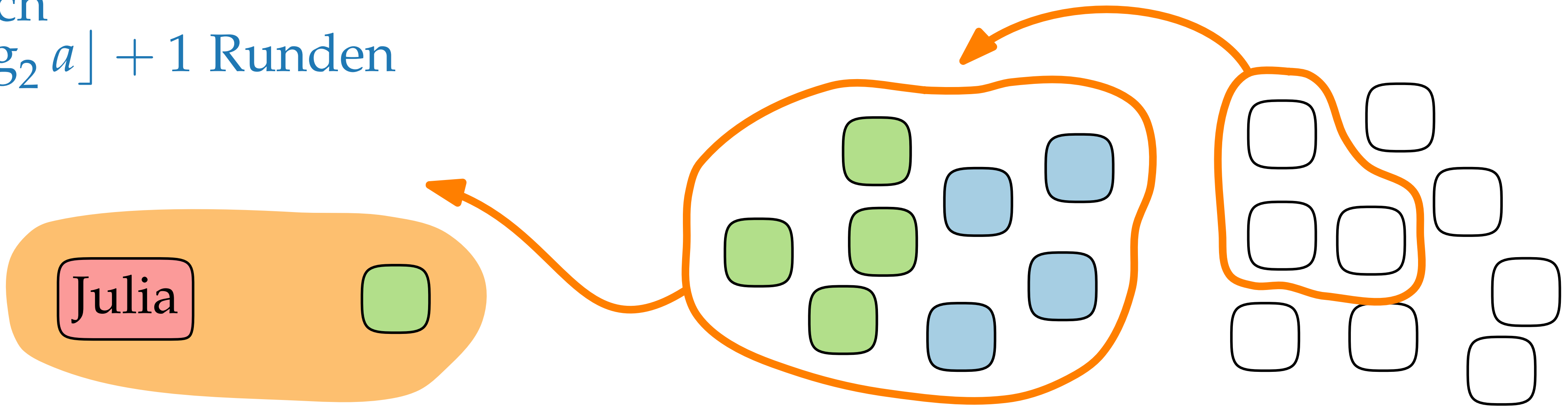
Noch schneller?



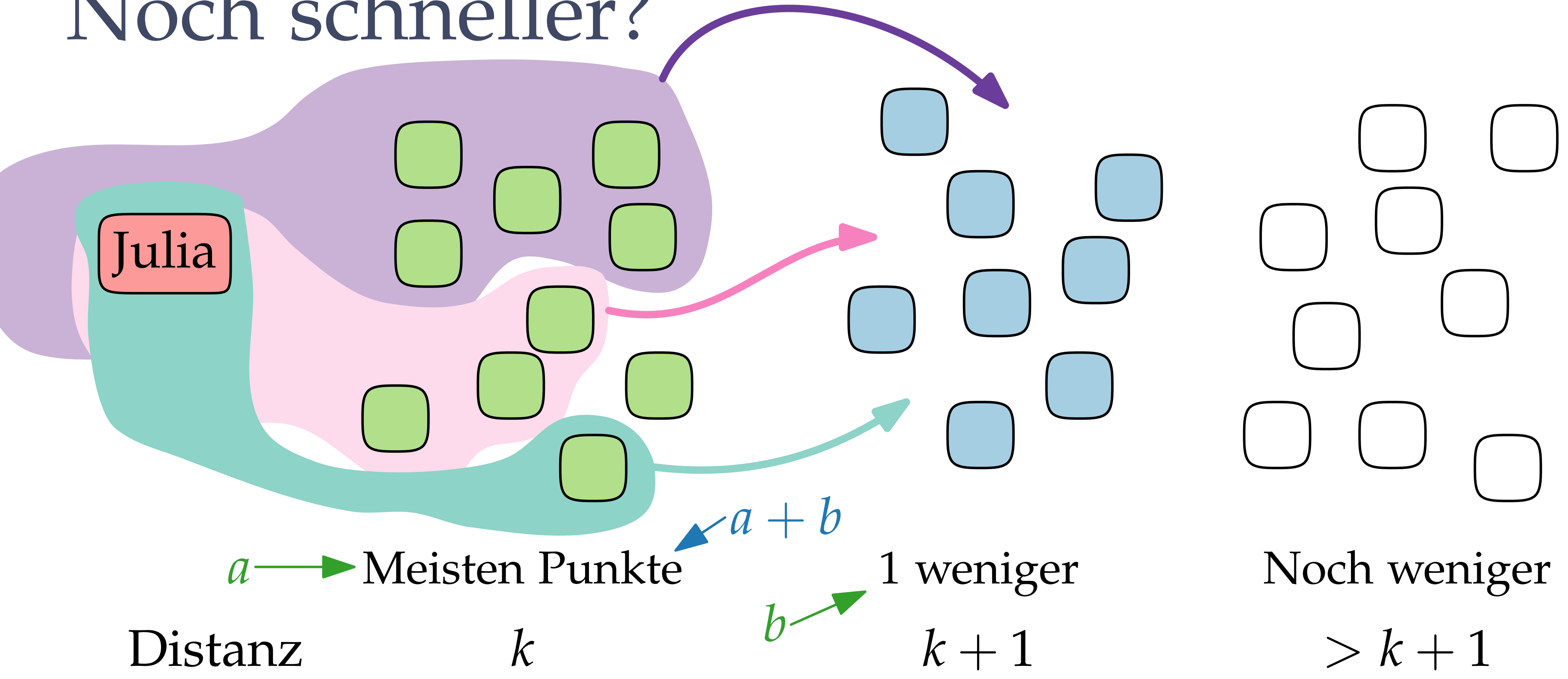
Noch schneller?



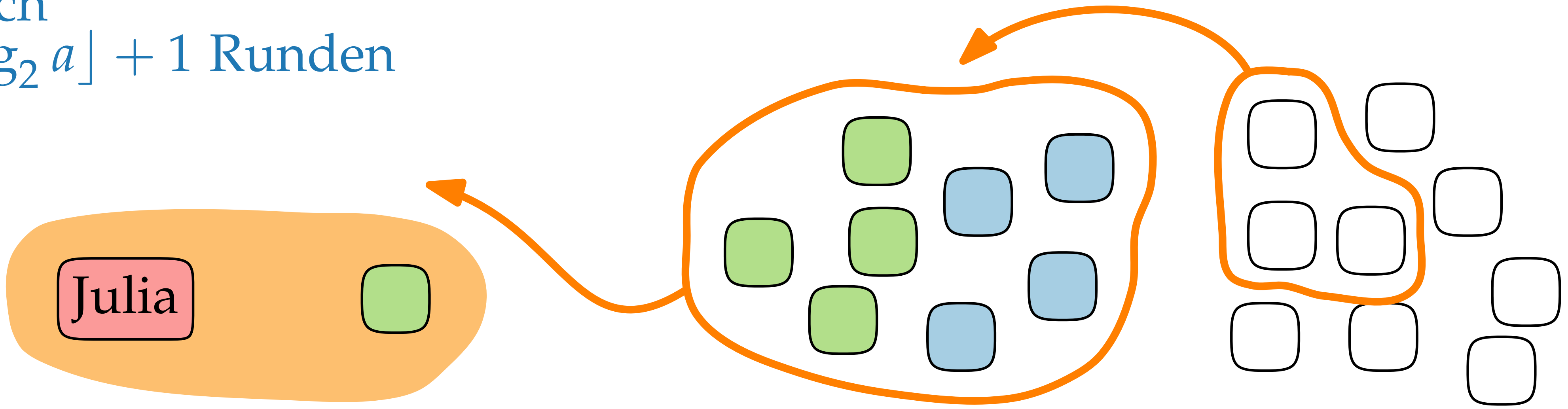
Nach
$\lfloor \log_2 a \rfloor + 1$ Runden



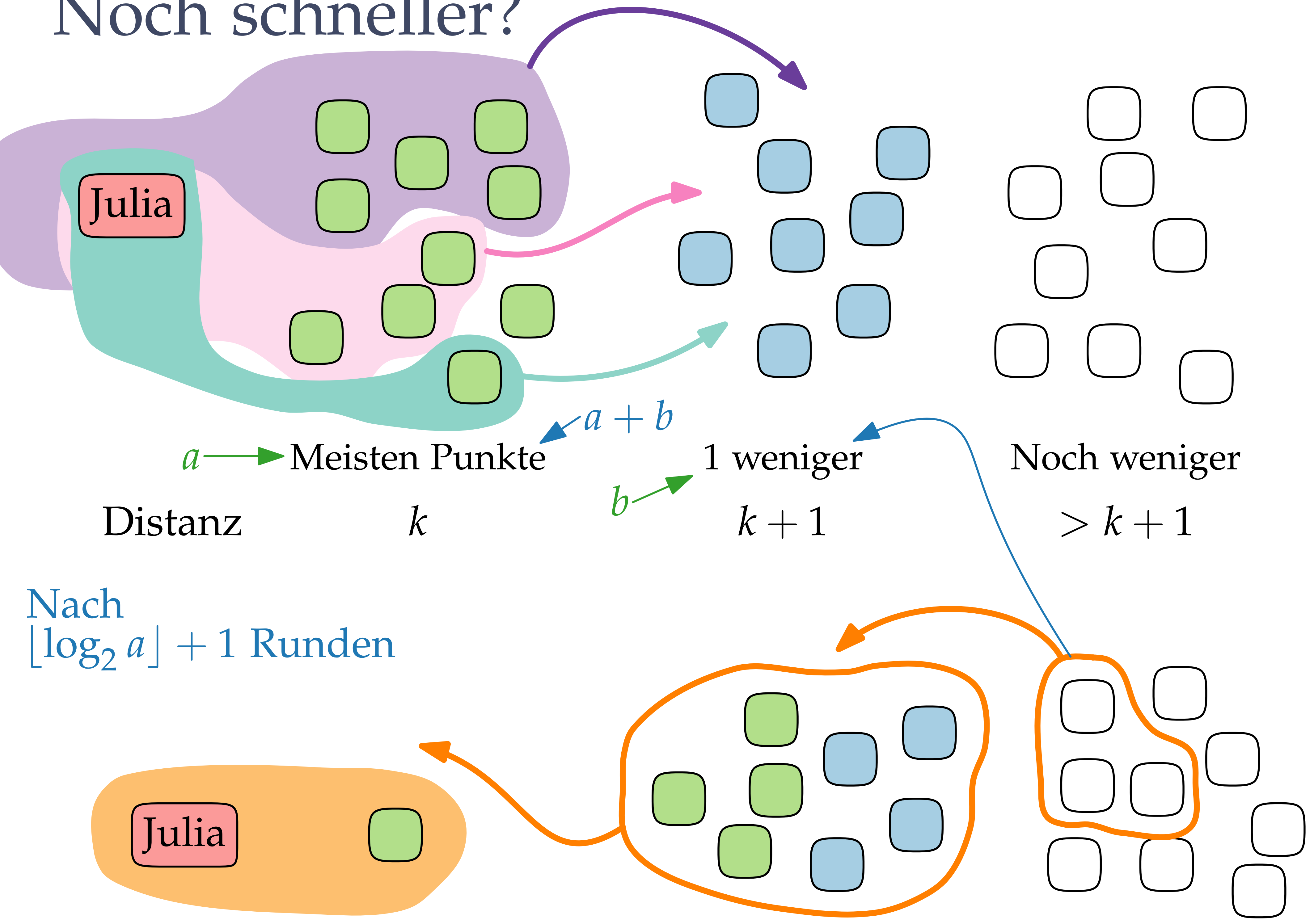
Noch schneller?



Nach
$\lfloor \log_2 a \rfloor + 1$ Runden

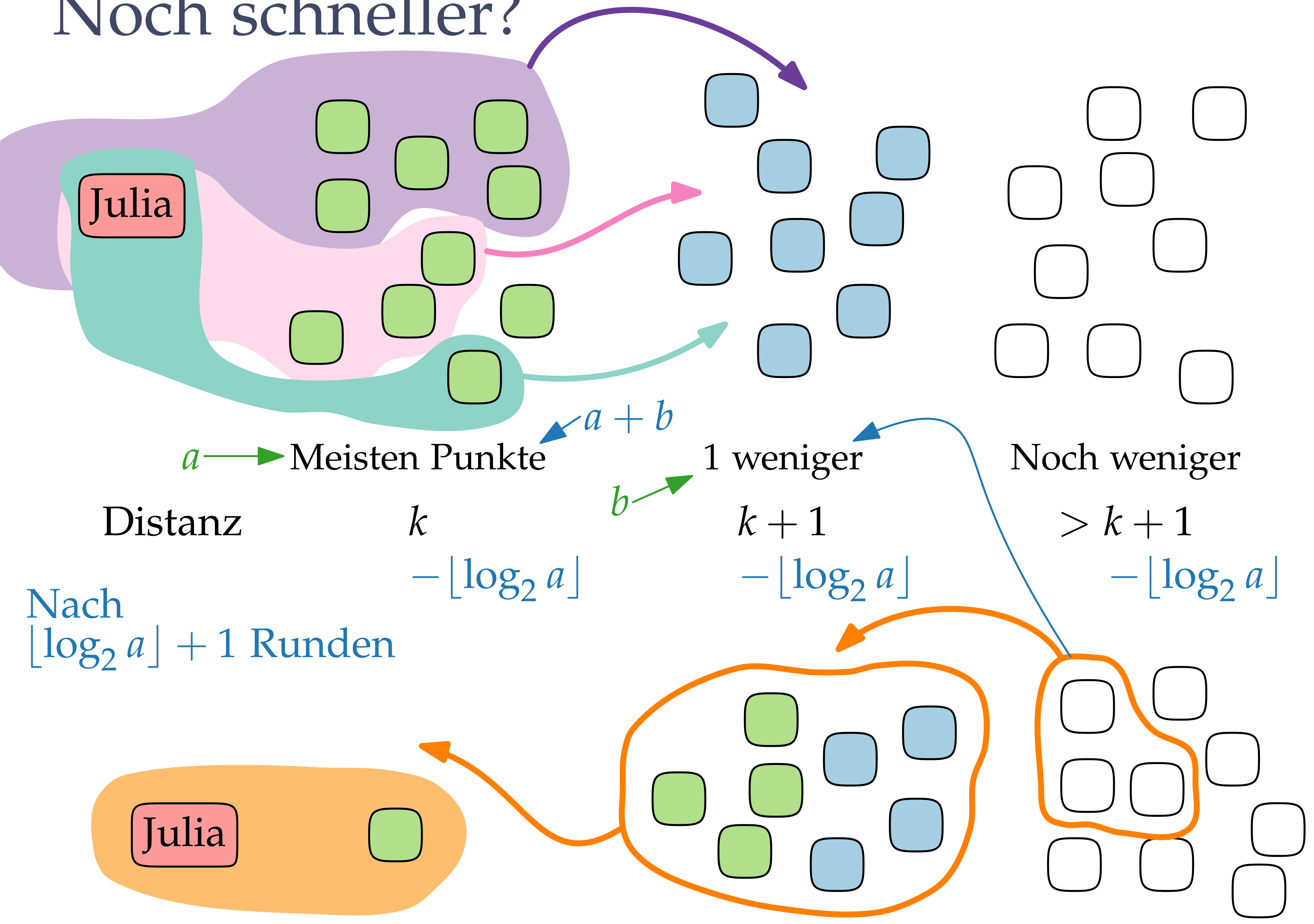


Noch schneller?



Nach
$\lfloor \log_2 a \rfloor + 1$ Runden

Noch schneller?



Julia
a
Meisten Punkte
$a+b$
b
1 weniger
Noch weniger
Distanz
k
$k+1$
$> k+1$
$-\lfloor \log_2 a \rfloor$
$-\lfloor \log_2 a \rfloor$
$-\lfloor \log_2 a \rfloor$
Nach $\lfloor \log_2 a \rfloor + 1$ Runden
Julia

```python
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:








print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    closest_log = int(log2(closest))









print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```python
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    closest_log = int(log2(closest))
    if distance < closest_log:
        it += distance + 1
        break











print (it - 1)
```

```python
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    closest_log = int(log2(closest))
    if distance < closest_log:
        it += distance + 1
        break
    distance -= closest_log
    closest += almost_closest
    it += closest_log + 1




print (it - 1)
```

J
5

F	B	C	D	E	A
1	1	1	2	3	4

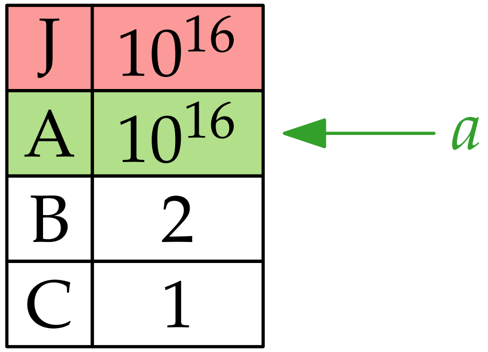
J
5

F	B	C	D	E	A
1	1	1	2	3	4

```python
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    closest_log = int(log2(closest))
    if distance < closest_log:
        it += distance + 1
        break
    distance -= closest_log
    closest += almost_closest
    it += closest_log + 1
    # finde neue almost_closest
    almost_closest = 0
    while closest + almost_closest < len(scores) and \
            scores[closest + almost_closest] <= distance + it + 1:
        almost_closest += 1
print (it - 1)
```

Immer noch zu langsam?

Immer noch zu langsam?



J	10^{16}
A	10^{16}
B	2
C	1

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\longleftarrow a$

$\Rightarrow$ immer noch 10^{16} Schritte

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\longleftarrow a$

$\Rightarrow$ immer noch 10^{16} Schritte

Lösung:

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\longleftarrow a$ $\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich

Immer noch zu langsam?

J	10^{16}	
A	10^{16}	$\longleftarrow a$
B	2	
C	1	

$\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich
$\Rightarrow$ Mache `score(A)`$-$`score(B)` “Superrunden” auf einmal

Immer noch zu langsam?

J	10^{16}	
A	10^{16}	$\longleftarrow a$
B	2	
C	1	

$\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich
$\Rightarrow$ Mache `score(A)`$-$`score(B)` “Superrunden” auf einmal
$\Rightarrow$ (`score(A)`$-$`score(B)`)$\cdot(\lfloor \log_2 a \rfloor + 1)$ Runden

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\longleftarrow a$ $\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich

$\Rightarrow$ Mache `score(A)`−`score(B)` "Superrunden" auf einmal

$\Rightarrow$ (`score(A)`−`score(B)`)$\cdot(\lfloor \log_2 a \rfloor + 1)$ Runden

J	10^{16}
A	10^{16}
B	10^{16}
C	$10^{16}-1$

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\leftarrow a$

$\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich

$\Rightarrow$ Mache `score(A)`$-$`score(B)` "Superrunden" auf einmal

$\Rightarrow$ (`score(A)`$-$`score(B)`)$\cdot(\lfloor \log_2 a \rfloor + 1)$ Runden

J	10^{16}
A	10^{16}
B	10^{16}
C	$10^{16}-1$

- Wird in einer dieser Superrunden Julia überholt?

Immer noch zu langsam?

J	10^{16}
A	10^{16}
B	2
C	1

$\longleftarrow a$

$\Rightarrow$ immer noch 10^{16} Schritte

Lösung: Bis A und B mergen sind alle Runden gleich

$\Rightarrow$ Mache `score(A)`$-$`score(B)` “Superrunden” auf einmal

$\Rightarrow$ $(\texttt{score(A)}-\texttt{score(B)})\cdot(\lfloor\log_2 a\rfloor+1)$ Runden

J	10^{16}
A	10^{16}
B	10^{16}
C	$10^{16}-1$

- Wird in einer dieser Superrunden Julia überholt?
- Was, wenn alle Konkurrenten den gleichen Score haben?

J
5

F	B	C	D	E	A
1	1	1	2	3	4

```
from math import log2

n = int(input())
scores = [int(x) for x in input().split()]
julia = scores[0]
scores = sorted([julia - x for x in scores[1:]])
distance = scores[0]
closest = 0 # anzahl teilnehmer mit meister punktzahl
while closest < n - 1 and scores[closest] == distance:
    closest += 1
almost_closest = 0 # anzahl teilnehmer mit meister punktzahl - 1
while closest + almost_closest < n - 1 and \
        scores[closest + almost_closest] <= distance + 1:
    almost_closest += 1

it = 0
while distance >= 0:
    closest_log = int(log2(closest))
    if distance < closest_log:
        it += distance + 1
        break
    distance -= closest_log
    closest += almost_closest
    it += closest_log + 1
    # finde neue almost_closest
    almost_closest = 0
    while closest + almost_closest < len(scores) and \
            scores[closest + almost_closest] <= distance + it + 1:
        almost_closest += 1
print (it - 1)
```