Problem P: Assistant to the Regional Manager

Bildquelle:
https://tinyurl.com/bildp1

Agenda

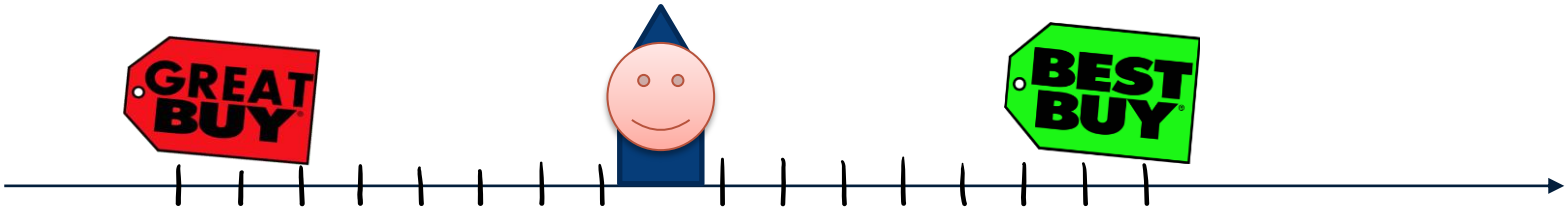
1. Das Problem
2. Unser Ansatz
3. Laufzeitanalyse

Das Problem

Wir befinden uns in der Stadt One-Road City, indem die Supermarktkette *Great Buy* eine Monopolstellung hat. Die Stadt, wie der Name bereits vermuten lässt, besteht dabei nur aus einer Straße, der Smith-Street mit ***k*** Häusern und ***n*** Great Buy Geschäfte entlang dieser Straße.

Als Assistent der Supermarktkette *Best Buy* möchtet ihr die Monopolstellung beenden. Du wirst damit beauftragt Läden mit einer Anzahl ***m*** in der Stadt One-Road zu eröffnen.

Studien zeigen, dass die Kunden faul sind und immer zum nächstgelegenen Geschäft gehen. Falls der Weg gleich lang ist, bevorzugen die Kunden die bekannte Marke Great Buy.



Das Problem

Wir befinden uns in der Stadt One-Road City, indem die Supermarktkette *Great Buy* eine Monopolstellung hat. Die Stadt, wie der Name bereits vermuten lässt, besteht dabei nur aus einer Straße, der Smith-Street mit ***k*** Häusern und ***n*** Great Buy Geschäfte entlang dieser Straße.

Als Assistent der Supermarktkette *Best Buy* möchtet ihr die Monopolstellung beenden. Du wirst damit beauftragt Läden mit einer Anzahl ***m*** in der Stadt One-Road zu eröffnen.

Studien zeigen, dass die Kunden faul sind und immer zum nächstgelegenen Geschäft gehen. Falls der Weg gleich lang ist, bevorzugen die Kunden die bekannte Marke Great Buy.

Deine Aufgabe ist es nun herauszufinden, wie viele Haushalte man mit einer optimalen Platzierung an neuen Geschäften gewinnen kann.

Du darfst deine neuen Läden überall platzieren, also auch auf bereits besetzte oder nicht-integer Koordinaten.

Input

10 2 2
1 2 3 4 6 7 8 9 21 22
5 10

Erste Zeile beinhaltet Integer:
k = **Anzahl** Häuser
n = **Anzahl** Geschäfte Great Buy
m = **Anzahl** neuer Geschäfte Best Buy
wobei (1 ≤ k, n, m ≤ 10^6 and k + n + m ≤ 10^6)

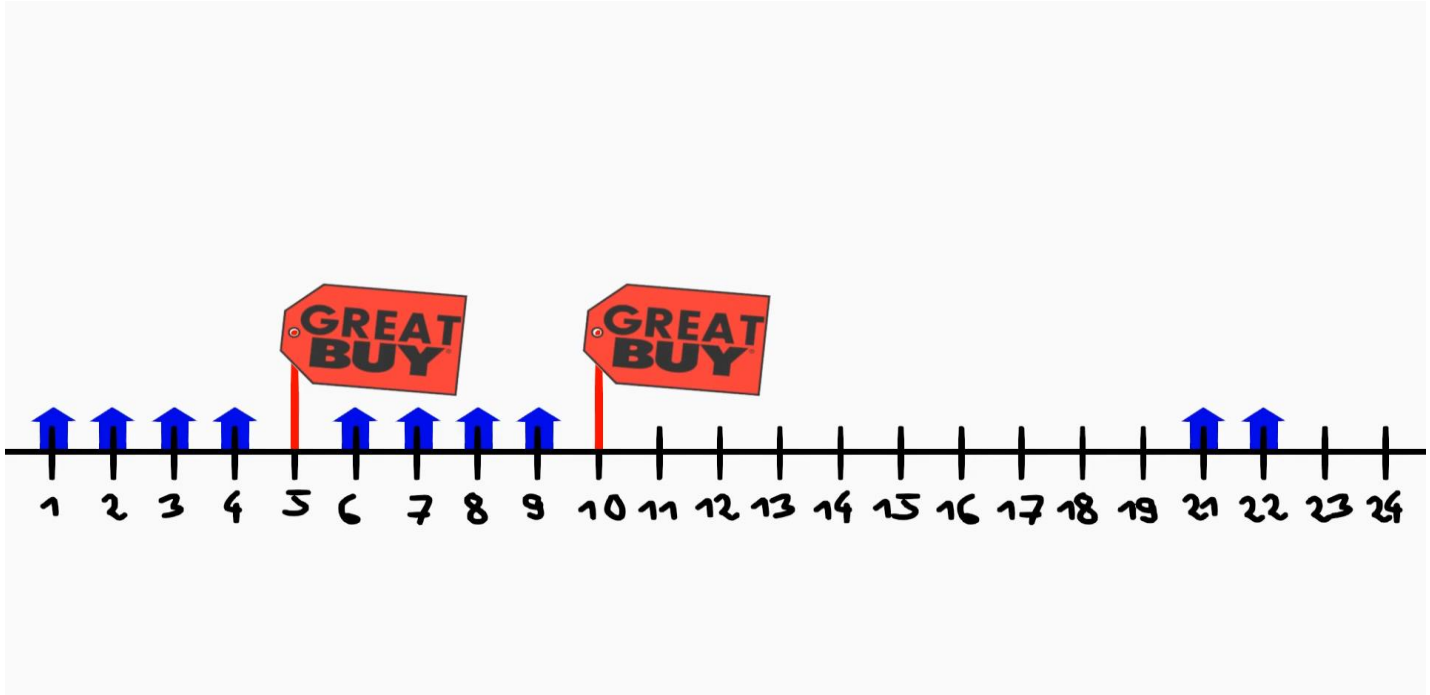
Zweite Zeile beinhaltet Integer:
Koordinaten von k die jeweils in [0;10^6] liegen

Dritte Zeile beinhaltet Integer:
Koordinaten von n die jeweils in [0;10^6] liegen

Häuser und Great Buy Stores sind dabei niemals an derselben Position!

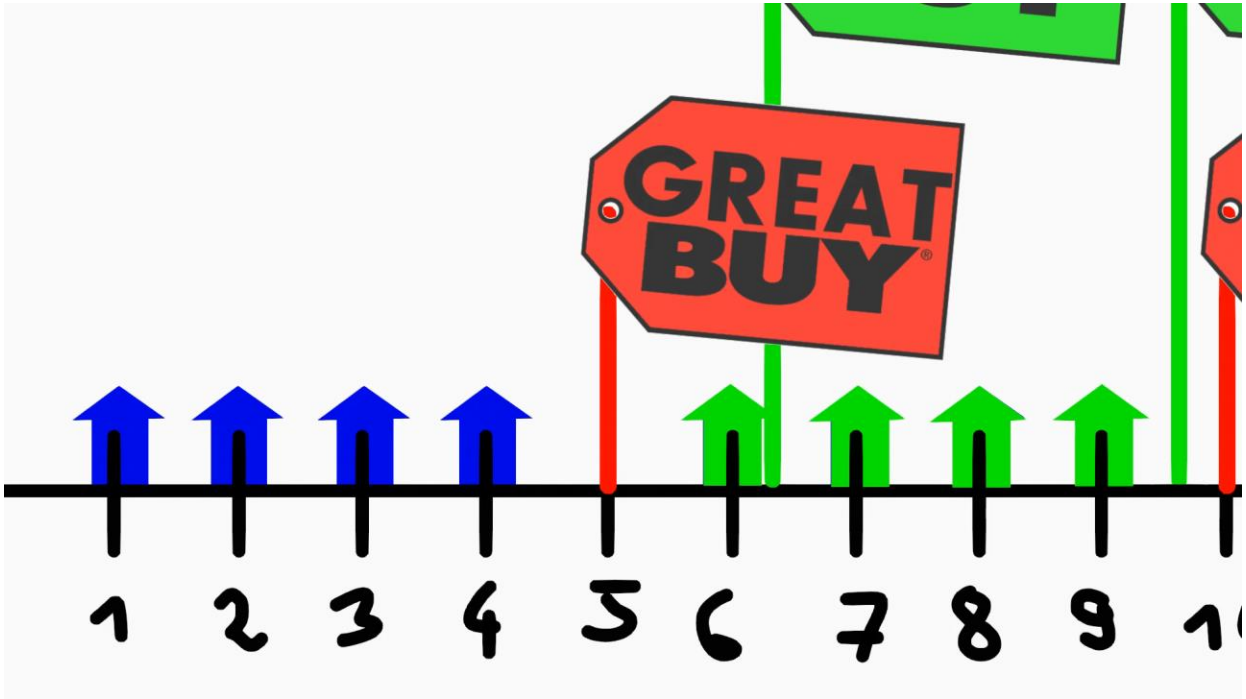
Input visualisiert

10 2 2
1 2 3 4 6 7 8 9 21 22
5 10



Output

Anzahl der Häuser bei optimaler Platzierung neuer Geschäfte, die näher zu Best Buy liegen als zu Great Buy.



Erste Beobachtung:
Links vom linksten Great Buy reicht **ein** Best Buy aus um alle Häuser zu gewinnen (analog zum Rechtesten)

Zweite Beobachtung:
Durch setzen von zwei Best Buys zwischen zwei Great Buys erhält man **alle** Häuser im Intervall

⇒ Man braucht also nicht mehr als zwei Best Buys je Intervall

Brute Force ??

Fragestellung: Könnte Brute Force eine Möglichkeit zur Problemlösung sein?

⇒ Nein, die Kombinatorik zeigt nämlich, dass wir im gesamten Intervall (Länge n+k) insg. n+k über m Möglichkeiten haben, einen Best Buy zu setzen.

⇒ O((n+k)^m) d.h. exponentieller Wachstum

⇒ Das schaffen wir besser ☺

Unser Lösungsansatz

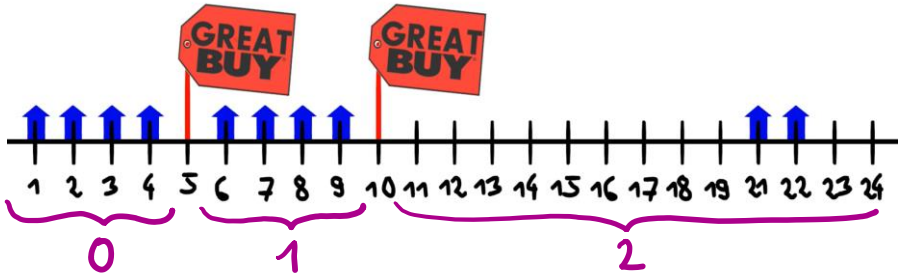
1. Zerlegung des Problems in Intervalle

2. Berechnung der gewonnen Häuser mithilfe Line-Sweap-Algorithmus, wenn i Stores im Intervall j platziert werden

3. Zusammensetzen der Lösungen der Intervalle

Input sortieren
–
spart Zeit bei der Ermittlung des nächsten Great Buy Stores

1. Zerlegung des Problems in Intervalle



Offene Intervalle	Geschlossene Intervalle
bis zum ersten und ab dem letzten Great Buy Store	begrenzt durch zwei Great Buy Stores
Intervalle nummerieren	

Definition B(i,j) und B*(i,j)

- B(i,j) = „Gewinn an Häusern mit j Best Buys in Intervall i“
- B*(i,j) = „Gewinn an Häusern bei Eröffnung von Best Buy Nummer j in Intervall i“

$\Rightarrow$ B*(i,1) = B(i,1) und B*(i,2) = B(i,2) – B(i,1)

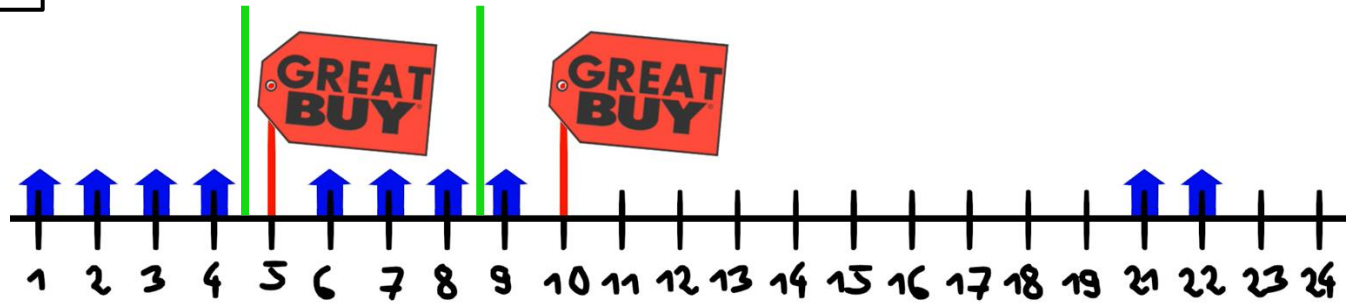
Wir haben bereits gesehen:

$\Rightarrow$ B*(i,j) = 0 für j > 2, da durch zwei Best Buys bereits alle Häuser im Intervall i gewonnen werden.

Sample 2

Sample2.in

```
10 2 2
1 2 3 4 6 7 8 9 21 22
5 10
```



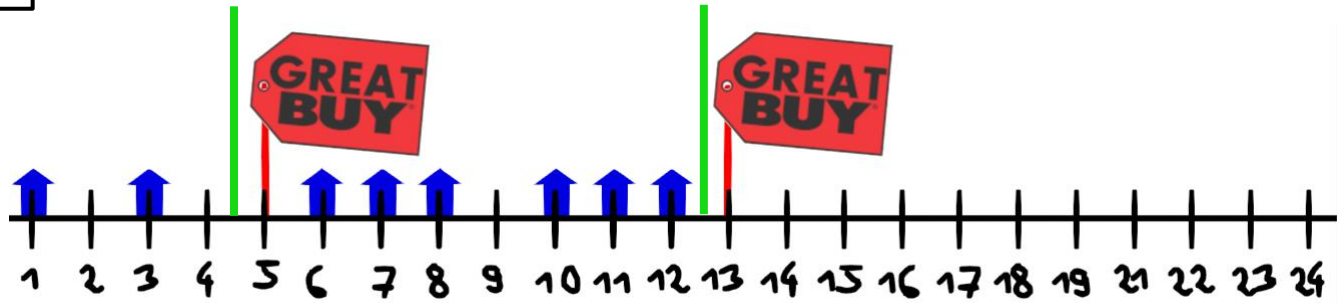
Gewinn an Häusern mit j BestBuys in Intervall i

B(i,1)	4	3	2
B(i,2)	4	4	2

Sample 0

Sample0.in

```
8 2 2
6 1 3 8 7 10 11 12
13 5
```



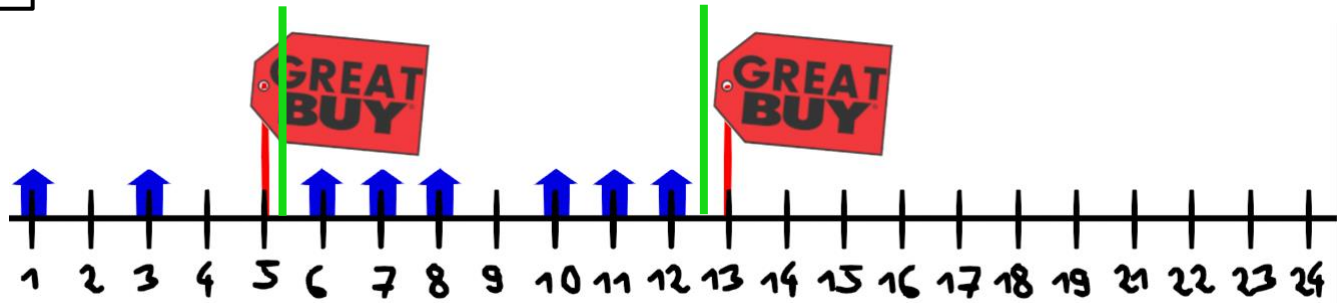
Gewinn an Häusern mit j BestBuys in Intervall i

B(i,1)	2	3	0
B(i,2)	2	6	0

Sample 0

Sample0.in

```
8 2 2
6 1 3 8 7 10 11 12
13 5
```



Gewinn an Häusern mit j BestBuys in Intervall i

B(i,1)	2	3	0
B(i,2)	2	6	0

Sample 2

Sample2.in

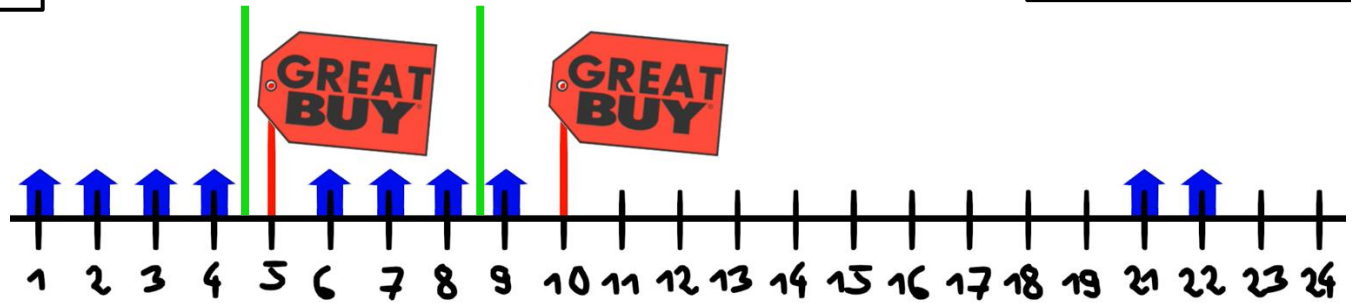
```
10 2 2
1 2 3 4 6 7 8 9 21 22
5 10
```

Sample2.out

```
7
4 3 2
0 1 0
```

Nur zum Debuggen

Gewinn an Häusern bei Eröffnung von BestBuy Nummer j im Intervall i



B*(i,1)	4	3	2
B*(i,2)	0	1	0

Sample 0

Sample0.in

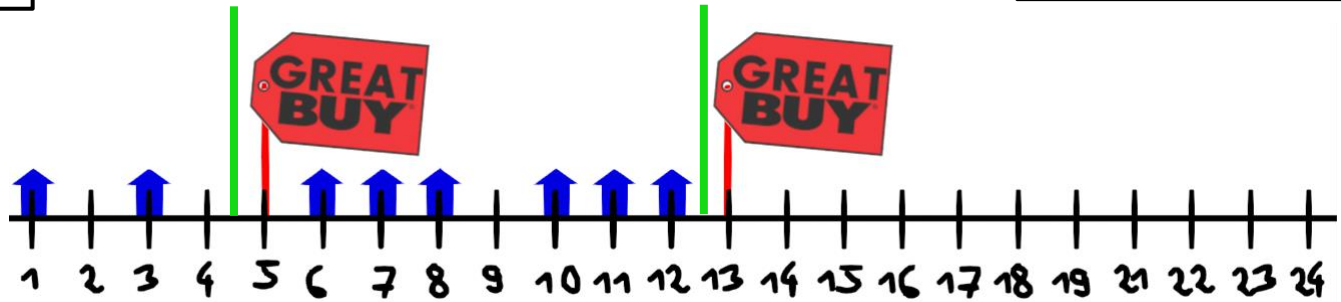
```
8 2 2
6 1 3 8 7 10 11 12
13 5
```

Sample0.out

```
6
2 3 0
0 3 0
```

Nur zum Debuggen

Gewinn an Häusern bei Eröffnung von BestBuy Nummer j im Intervall i



B*(i,1)	2	3	0
B*(i,2)	0	3	0

Sample 0

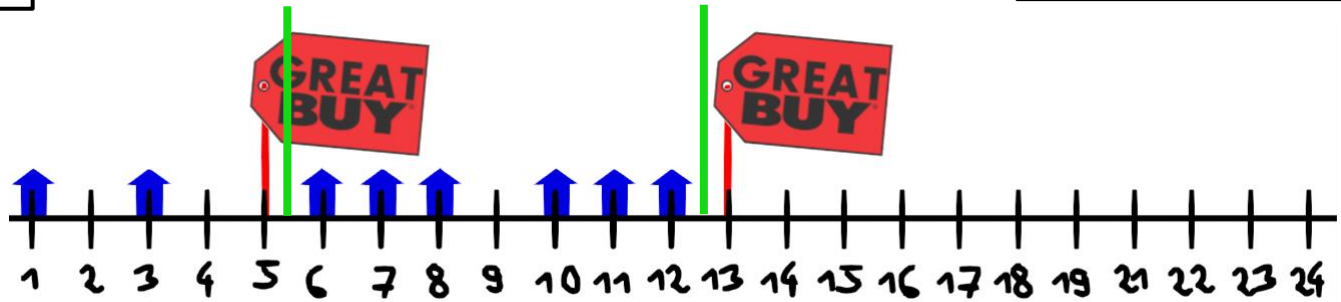
Sample0.in

```
8 2 2
6 1 3 8 7 10 11 12
13 5
```

Sample0.out

```
6
2 3 0
0 3 0
```

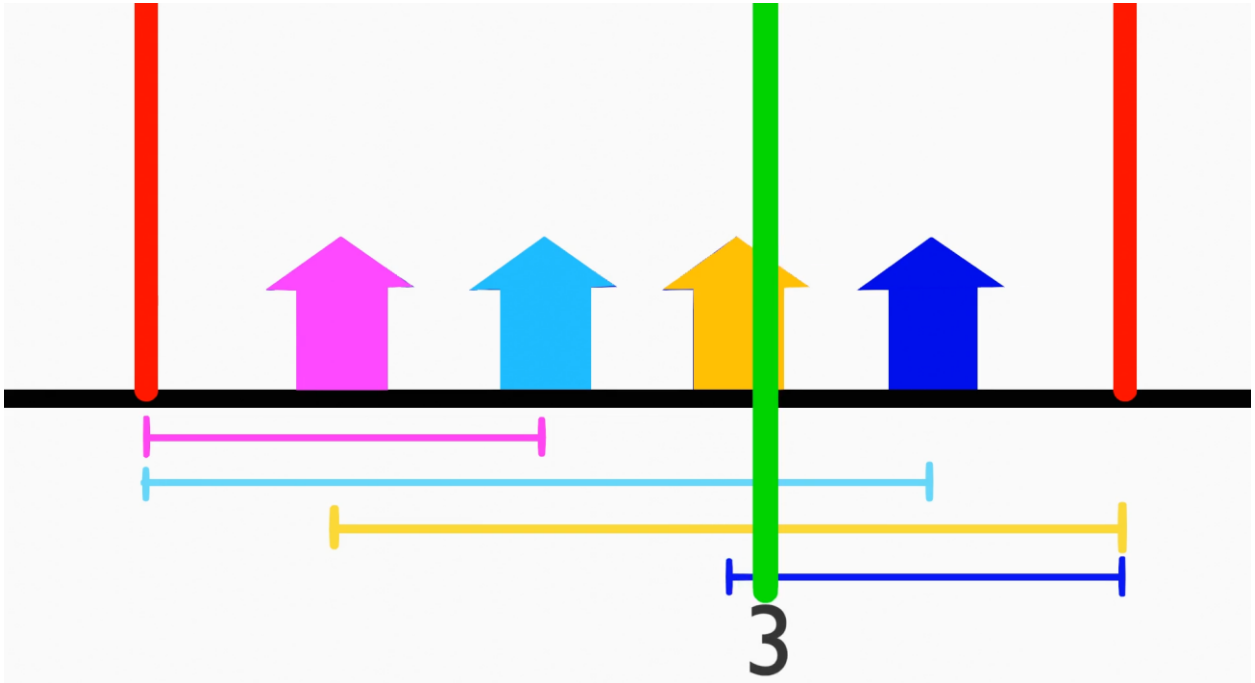
Nur zum Debuggen



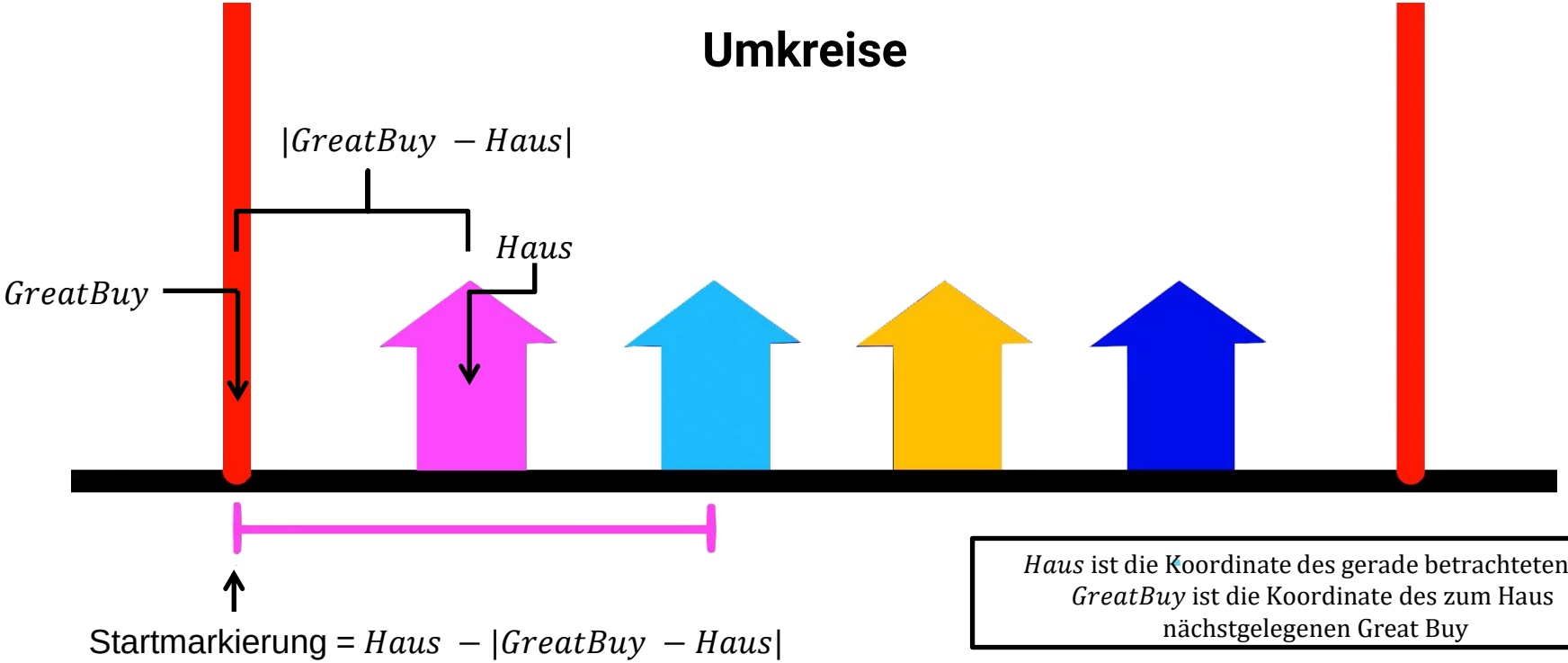
Gewinn an Häusern bei Eröffnung von BestBuy Nummer j im Intervall i

B*(i,1)	2	3	0
B*(i,2)	0	3	0

Line Sweep Algorithmus

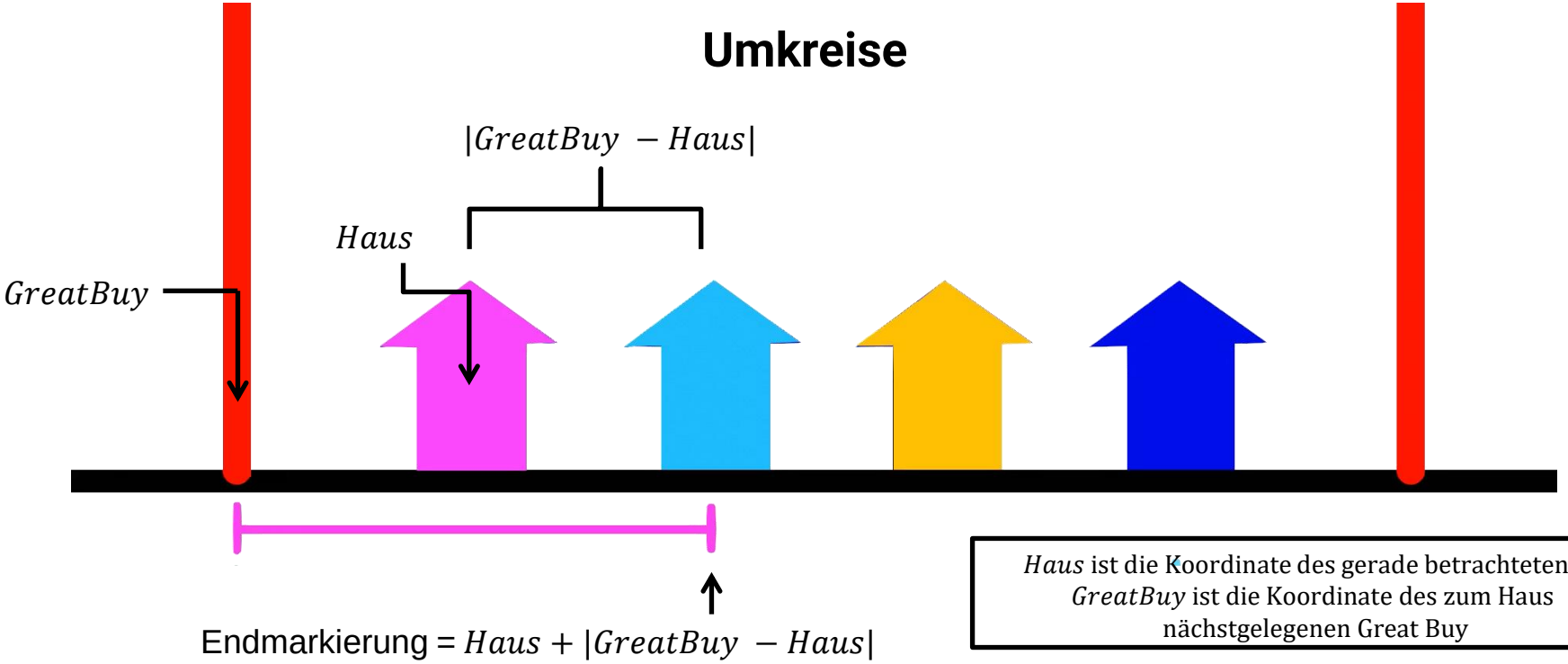


Umkreise



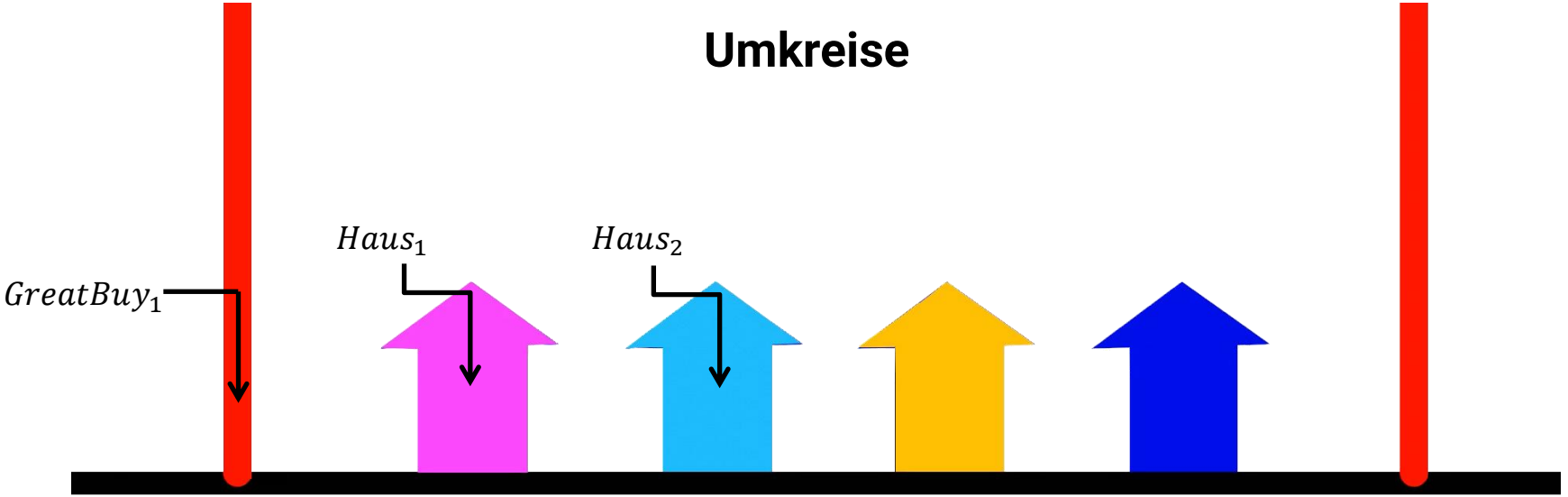
Haus ist die Koordinate des gerade betrachteten Hauses
GreatBuy ist die Koordinate des zum Haus nächstgelegenen Great Buy

Umkreise



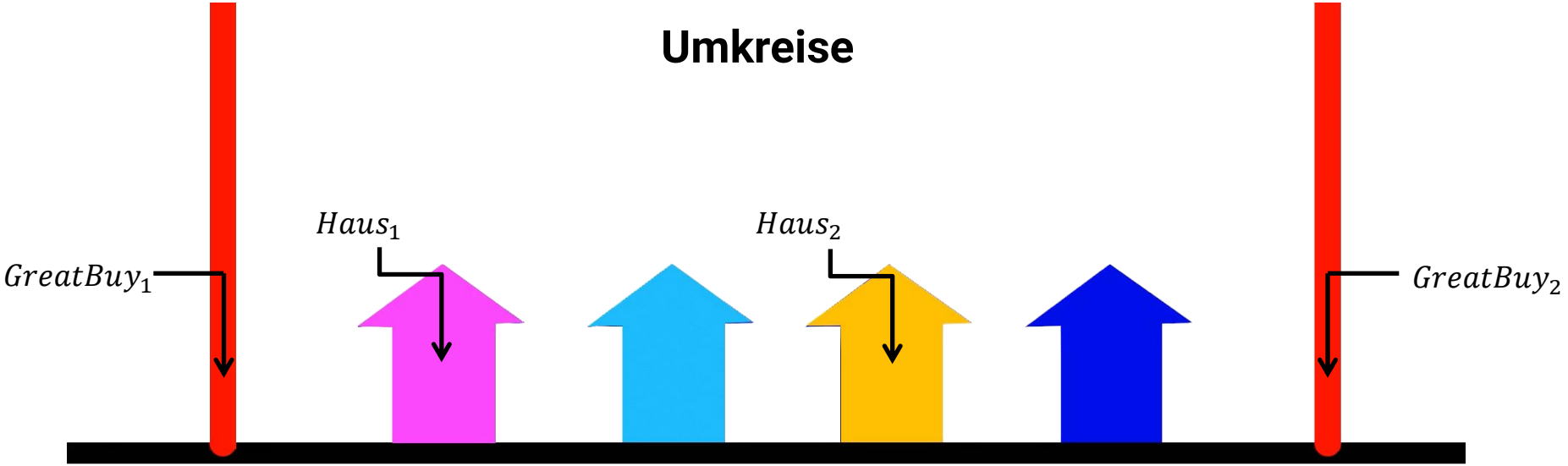
Haus ist die Koordinate des gerade betrachteten Hauses
GreatBuy ist die Koordinate des zum Haus nächstgelegenen Great Buy

Umkreise



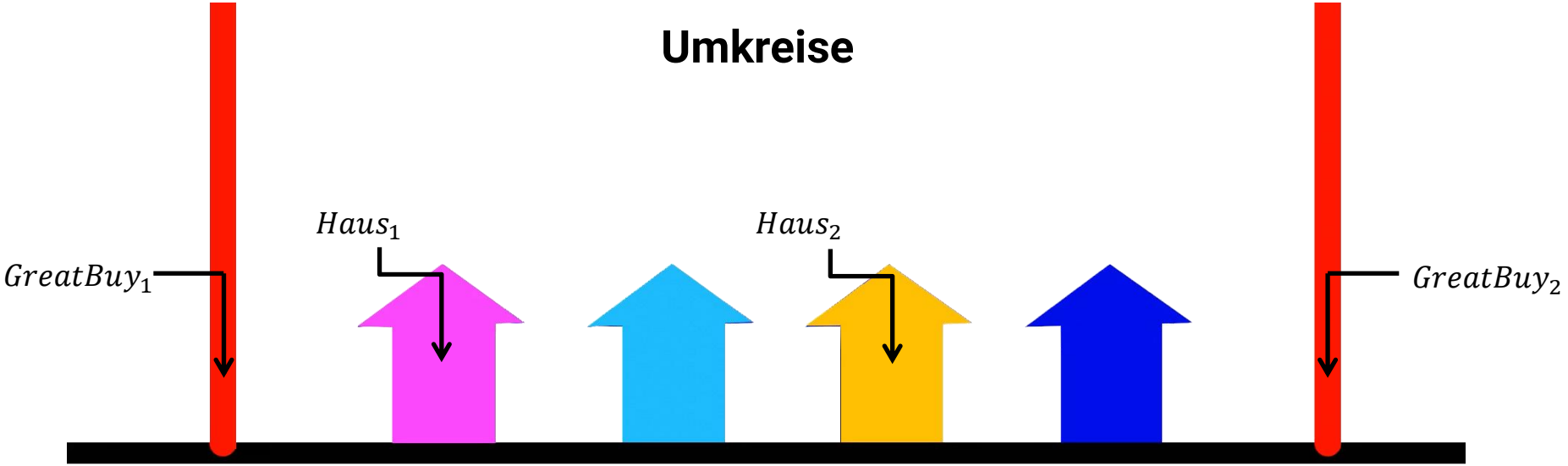
$$Endmarkierung_1 - Endmarkierung_2 = Haus_1 + |GreatBuy_1 - Haus_1| - (Haus_2 + |GreatBuy_1 - Haus_2|)$$
$$= (Haus_1 - Haus_2) + (|GreatBuy_1 - Haus_1| - |GreatBuy_1 - Haus_2|) < 0$$

Umkreise



$$Endmarkierung_1 - Endmarkierung_2 = Haus_1 + |GreatBuy_1 - Haus_1| - (Haus_2 + |GreatBuy_2 - Haus_2|)$$
$$= Haus_1 - GreatBuy_1 + Haus_1 - (Haus_2 + GreatBuy_2 - Haus_2)$$
$$= (Haus_1 - GreatBuy_1) - (GreatBuy_2 - Haus_1) < 0$$

Umkreise



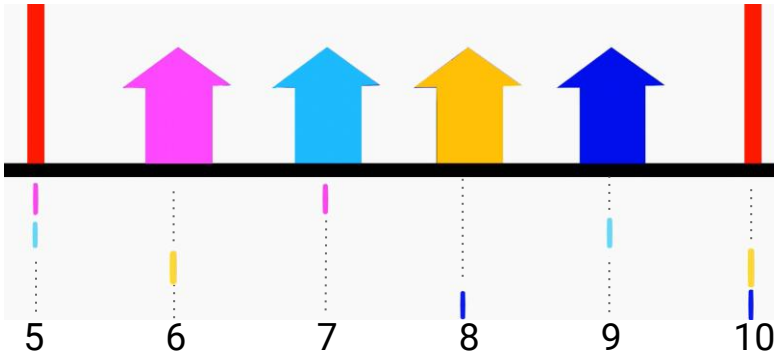
⇒ Wenn man von links nach rechts durch das Intervall durchgeht, dann sind die Startmarkierung und Endmarkierung sortiert

Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10

Markierung	5
Start oder Ende ?	S
Veränderung	+1

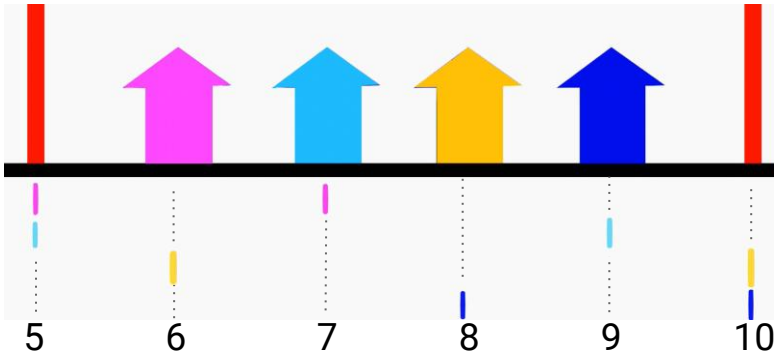


Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10

Markierung	5	5
Start oder Ende ?	S	S
Veränderung	+1	+1

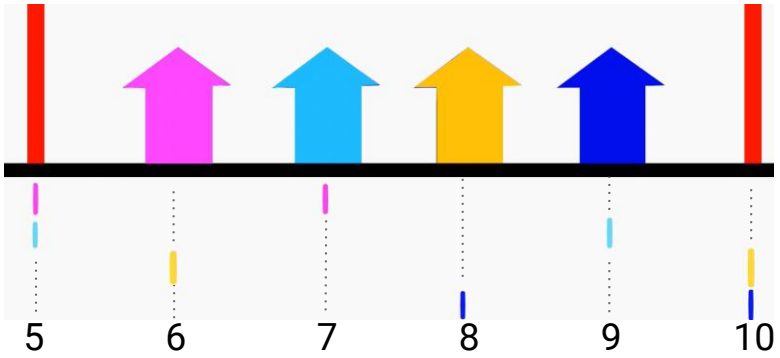


Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10

Markierung	5	5	6
Start oder Ende ?	S	S	S
Veränderung	+1	+1	+1

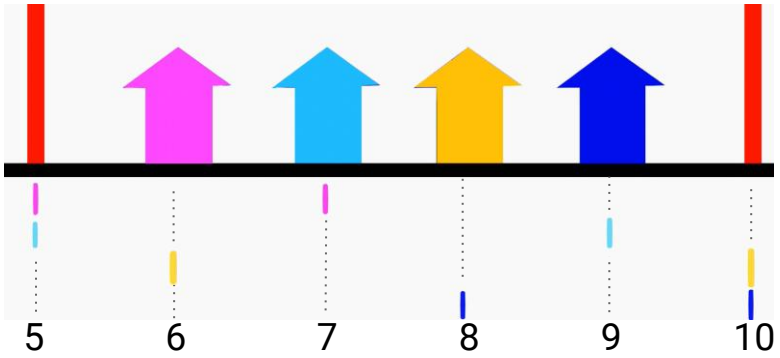


Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10

Markierung	5	5	6	7
Start oder Ende ?	S	S	S	E
Veränderung	+1	+1	+1	-1

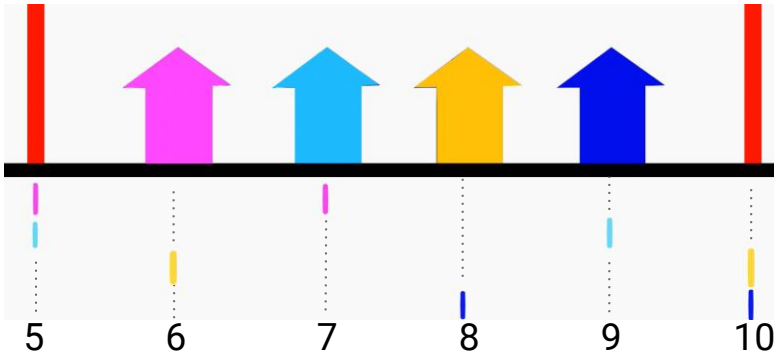


Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10

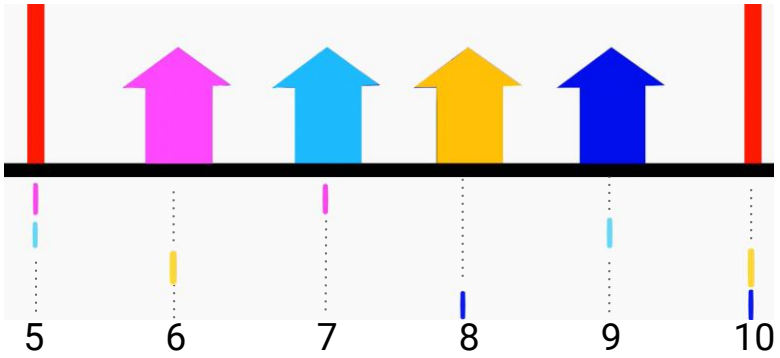
Markierung	5	5	6	7	8
Start oder Ende ?	S	S	S	E	S
Veränderung	+1	+1	+1	-1	+1



Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10



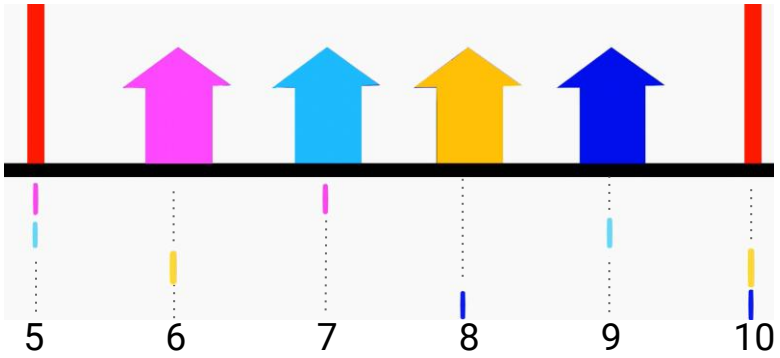
Markierung	5	5	6	7	8	9	10	10
Start oder Ende ?	S	S	S	E	S	E	E	E
Veränderung	+1	+1	+1	-1	+1	-1	-1	-1

Mergen in
O(Anzahl Häuser im Intervall)
=>
für alle Intervall zusammen O(k)

Mergen der Markierungen

Startmarkierung	5	5	6	8

Endmarkierung	7	9	10	10



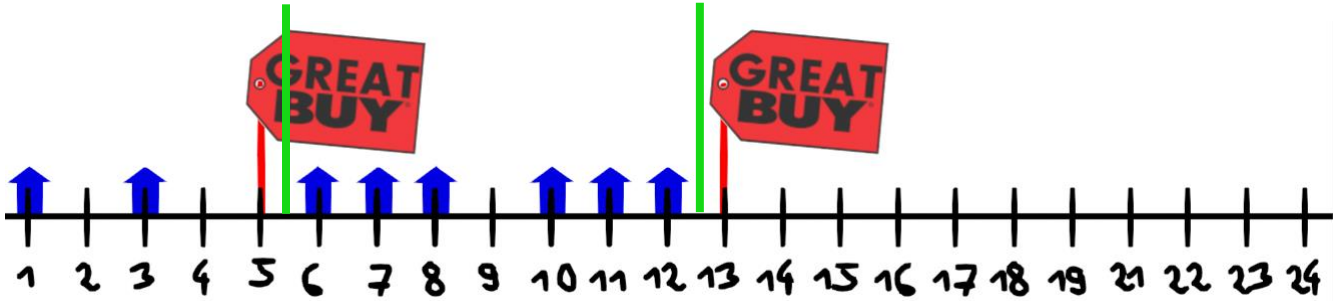
Markierung	5	5	6	7	8	9	10	10
Start oder Ende ?	S	S	S	E	S	E	E	E
Veränderung	+1	+1	+1	-1	+1	-1	-1	-1

Wenn zwei Markierungen an der selben Koordinate liegen, dann muss zuerst die Endmarkierung verarbeitet werden!

Sample 2

B*(i,1)	4	3	2
B*(i,2)	0	1	0

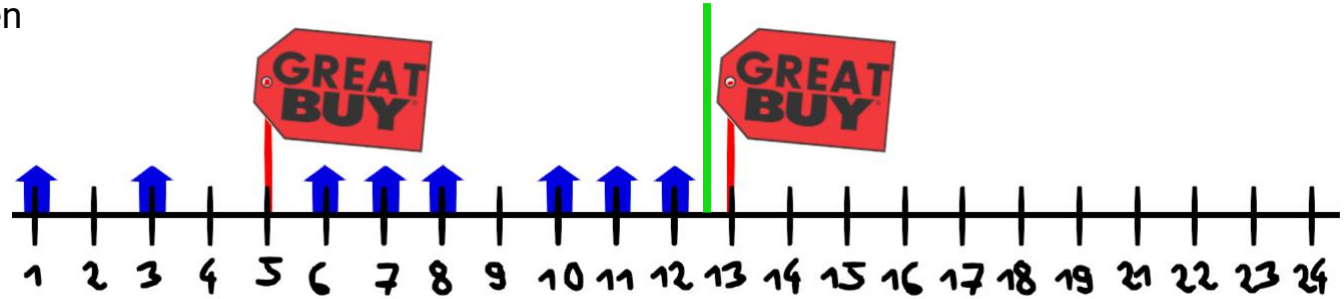
Sample 0



B*(i,1)	2	3	0
B*(i,2)	0	3	0

3. Zusammensetzen der Lösungen der Intervalle

- B*(i,j) gibt an wie viele Häuser mit dem BestBuy Nummer j im Intervall i gewinnt.
- Wenn man m BestBuys platzieren darf, wählt man die m größten B*(i,j), um den Gewinn an Häusern zu maximieren



- Durch paltzieren des BestBuys Nummer 1 in die Hälfte mit mehr Häusern, gewinnt man immer mindestens die Hälfte aller Häuser im Intervall.
- Der BestBuy Nummer 2 kann also nur noch maximal die Hälfte aller Häuser im Intervall Gewinnen
- => $B^*(i,2) \leq B^*(i,1)$ => Es wird nie der BestBuy Nummer 2 ausgewählt, bevor der BestBuy Nummer 1 ausgewählt wurde

- Nur für i∈ $\{0, \dots, n\}$ und j∈ $\{1,2\}$ ist B*(i,j) ungleich 0
- Berechne B*(i,j) für i∈ $\{0, \dots, n\}$ und j∈ $\{1,2\}$ mit Hilfe des LineSweep Algorithmus

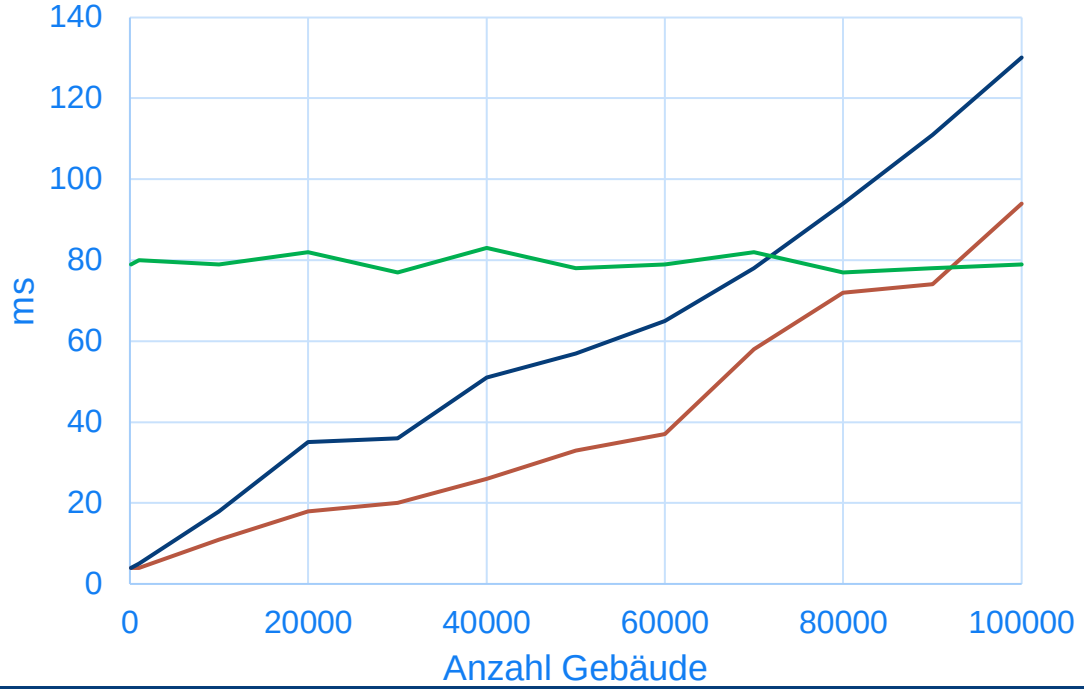
- Sortiere die B*(i,j) um die m größten B*(i,j) addieren zu können
- Die Summe entspricht der maximalen Anzahl an Häusern die man mit m BestBuys gewinnen kann

Laufzeit-Theorie

1. Inputs einlesen und sortieren — O(k log k) + O(n log n)
2. Anzahl der gewonnen Häuser mit einem bzw. zwei BestBuys pro Intervall — O(k)
   i. Vor dem ersten GreatBuy
   ii. Zwischen jeweils zwei GreatBuys
   iii. Hinter dem letzten GreatBuy
3. Sortieren der Ergebnisse der Intervall nach gewonnenen Häusern — O(n log n)
4. Addieren der m größten Ergebnisse — O(min{m,n})

=> O(k log k + n log n)

Laufzeit-Praxis



Laufzeit für jeweils zwei konstante Parameter und einer Variable. Festen Parameter haben den Wert 1000 Gebäuden.

- k ⇒ k Variabel, n und m fest
- n ⇒ n Variabel, k und m fest
- m ⇒ m Variabel, k und n fest