Binary Search

Everything

Célian “Cc” Raimbault

Binary Search

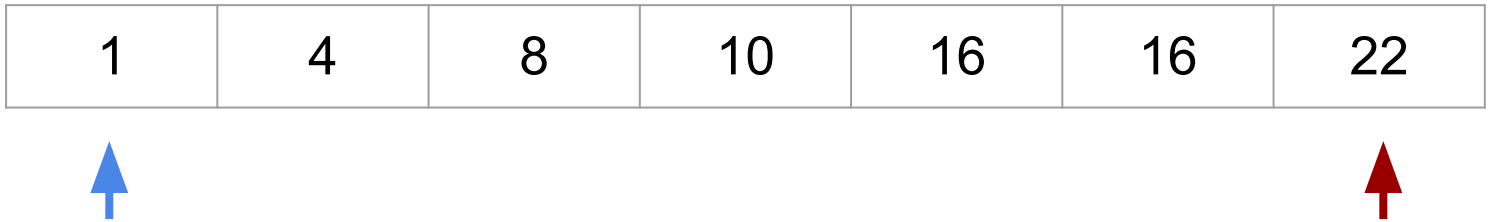
- Find a value in an ordered array
- **Fast** : O(log N) => ~30 queries for 1 000 000 000 items !
- Can even be used without arrays…

Binary Search

1	4	8	10	16	16	22

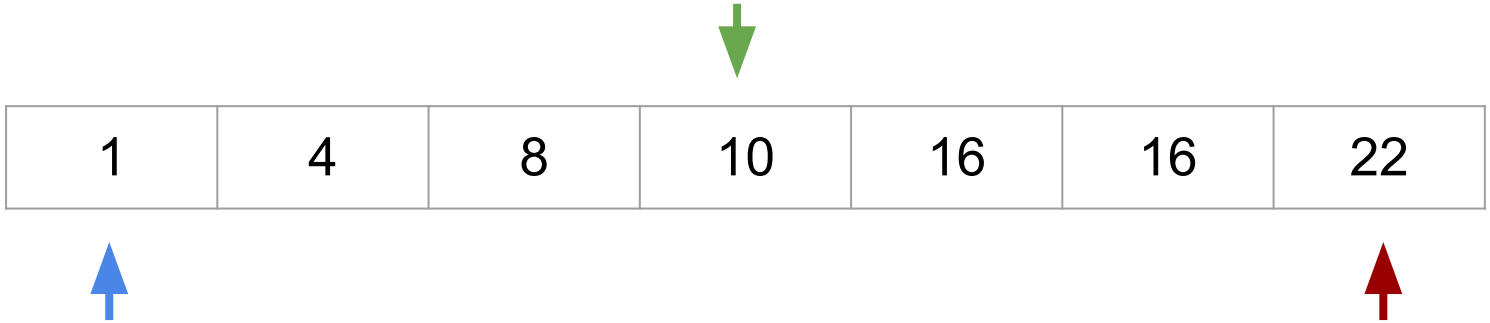
- Find 8 using binary search

Binary Search



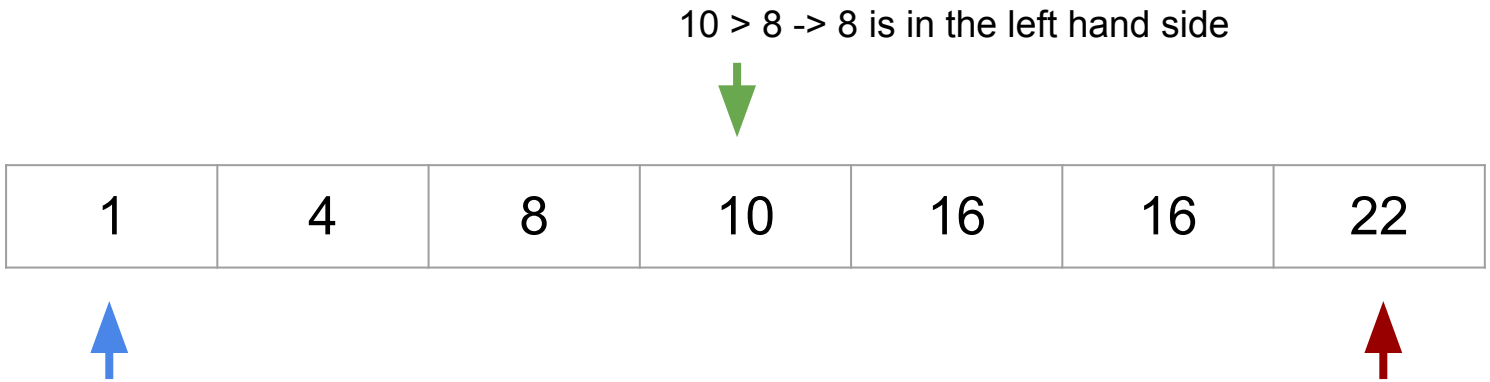
- Find 8 using binary search

Binary Search



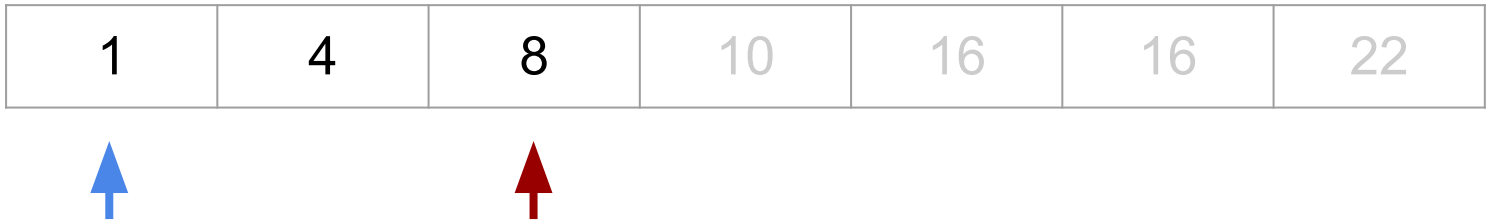
- Find 8 using binary search

Binary Search



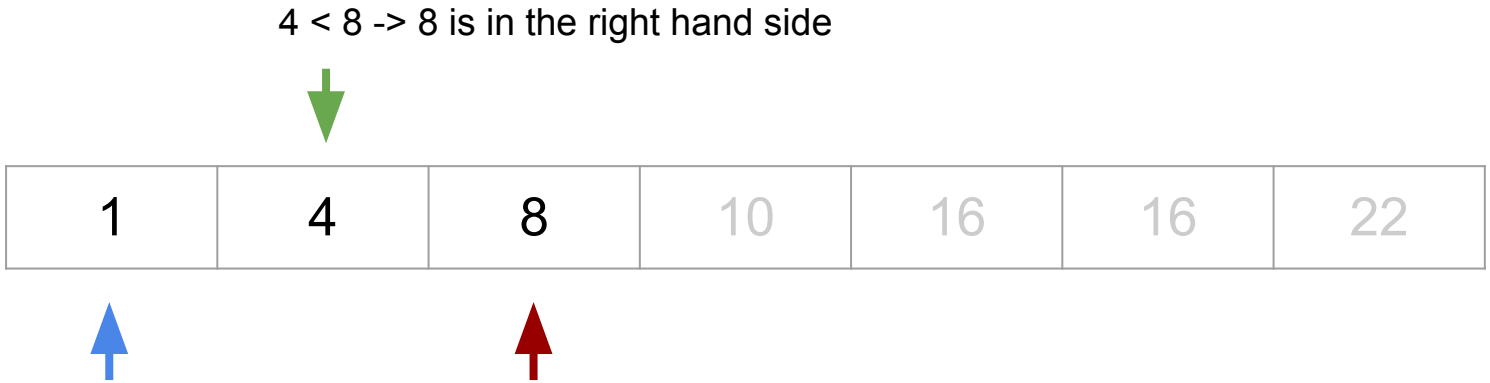
- Find 8 using binary search

Binary Search



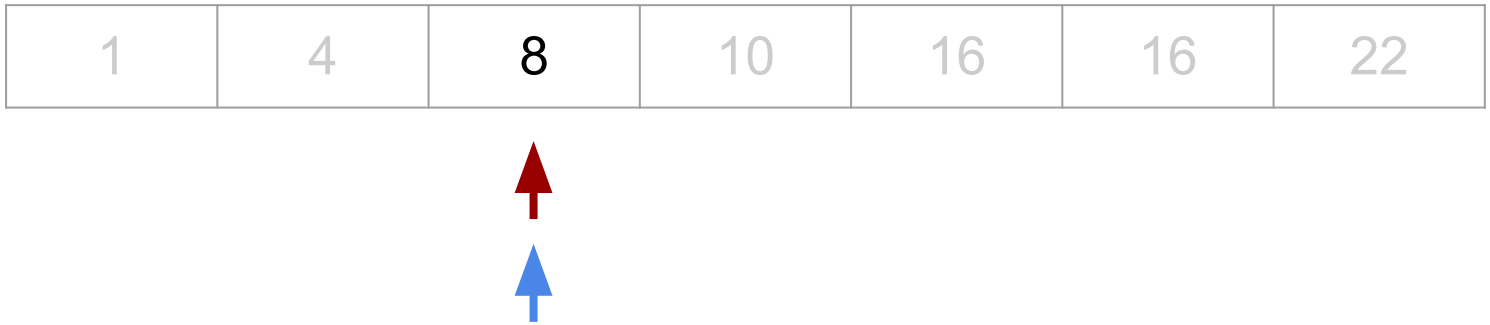
- Find 8 using binary search

Binary Search



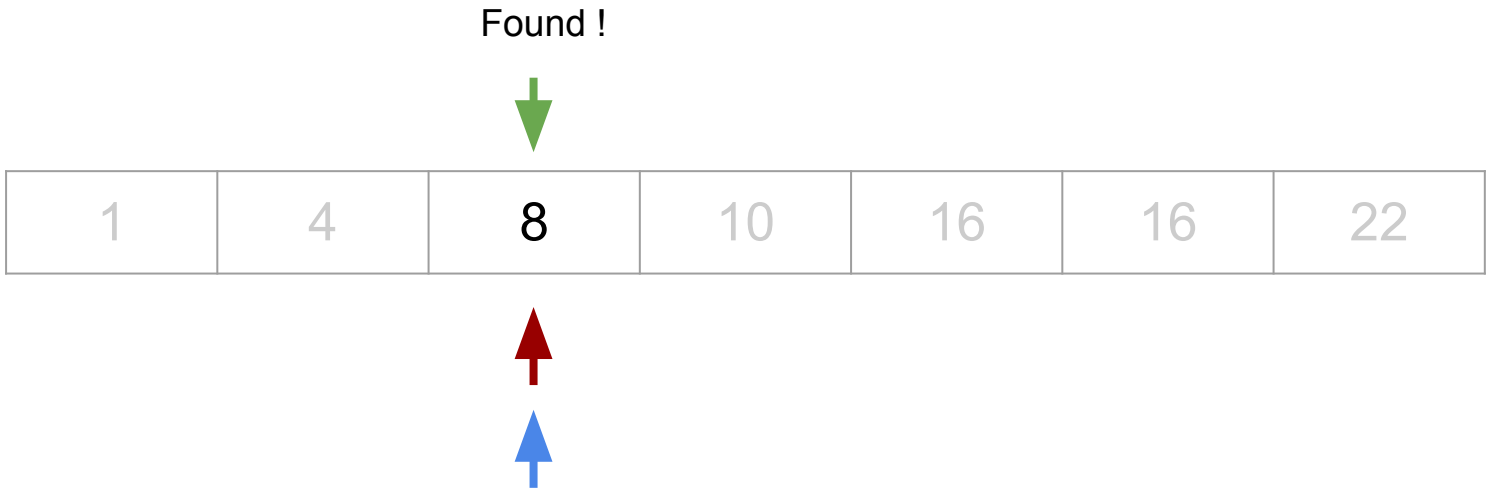
- Find 8 using binary search

Binary Search



- Find 8 using binary search

Binary Search



- Find 8 using binary search

Coding time !

- Website : https://cc618.github.io/Binary-Search-Everything
- Code stubs : https://github.com/Cc618/Binary-Search-Everything

Sqrt

- Given any integer **N**, can you find its integer square root using binary search ?
- In other words, compute *floor(sqrt(N))*

Sqrt

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt

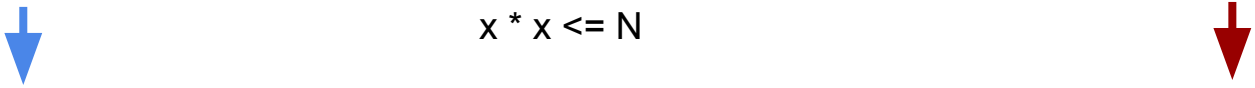
0 <= sqrt(N) <= N

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt

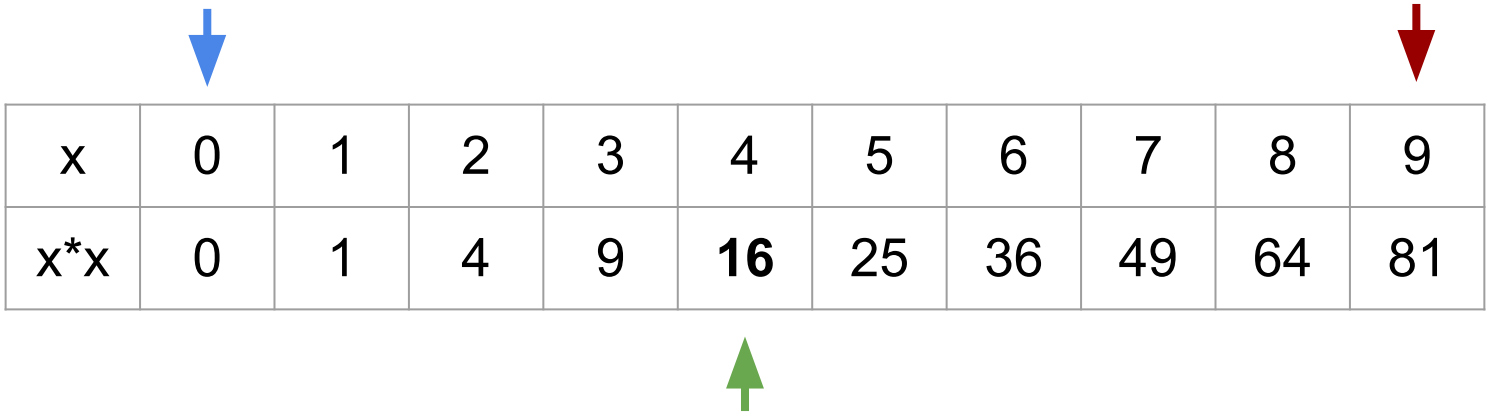
Find the last x such that
x <= sqrt(N)
that is :
x * x <= N



x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt



x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	**16**	25	36	49	64	81

16 > 9 -> Too high !

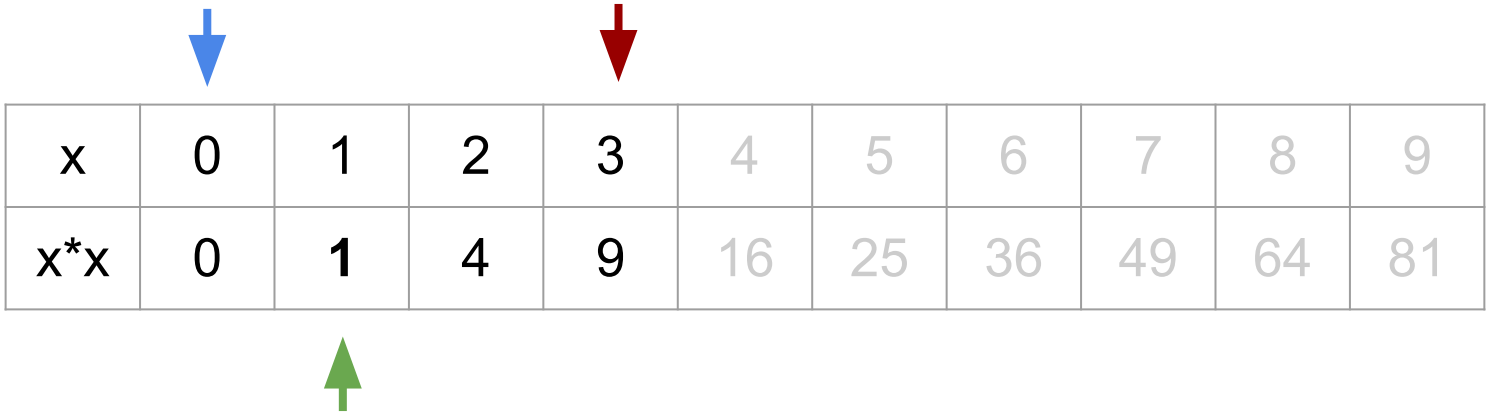
- Find sqrt(9) using binary search

Sqrt

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt



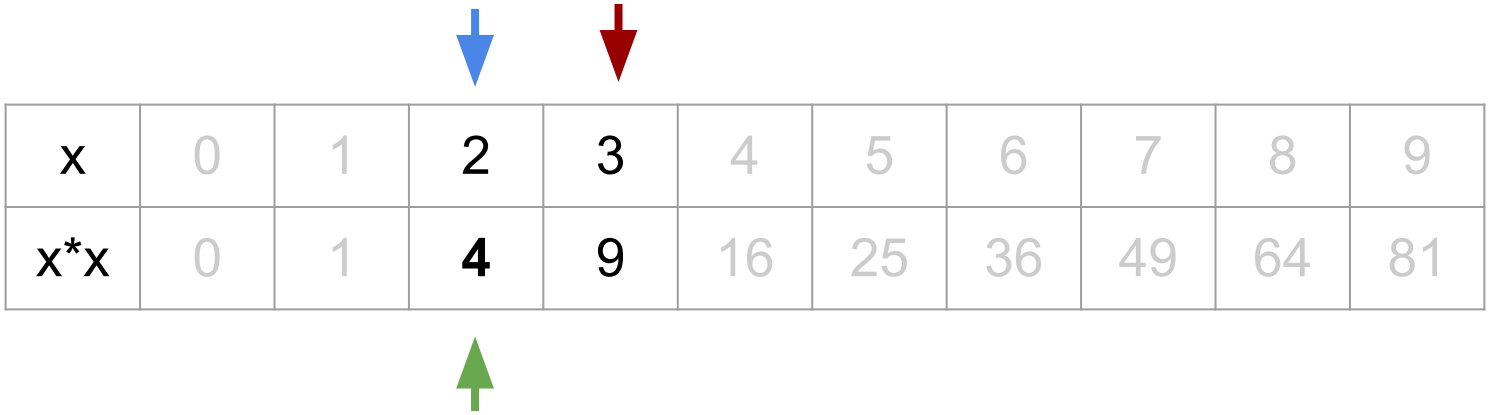
- Find sqrt(9) using binary search

Sqrt

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt



- Find sqrt(9) using binary search

Sqrt

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	9	16	25	36	49	64	81

- Find sqrt(9) using binary search

Sqrt

x	0	1	2	3	4	5	6	7	8	9
x*x	0	1	4	**9**	16	25	36	49	64	81

9 = 9 -> Found !

- Find sqrt(9) using binary search

Coding time !

- Website : https://cc618.github.io/Binary-Search-Everything
- Code stubs : https://github.com/Cc618/Binary-Search-Everything

Sticks

- There are **N** columns and a magic bag at the top of each column
- Each magic bag has a size

- You can generate **K** sticks in total from these bags
- At each generation, you choose a bag, take a stick and place it in its column
- The stick has the same size as the bag

- The score is the minimum of every column size
- Can you deduce the maximum possible score ?

Sticks

Sticks

Bags: 3 5 6

K = 5

Score = 0

Columns: **0** **0** **0**

Sticks

Bags: 3 5 6

K = 4

Score = 0

Columns: 3 **0** **0**

Sticks

Bags: 3 5 6

K = 3

Score = 0

Columns: 3 5 **0**

Sticks

Bags: 3 5 6

K = 2

Score = 3

Columns: **3** 5 6

Sticks

Bags: 3 5 6

K = 1

Score = 5

Columns: 6 **5** 6

Sticks

Bags: 3 5 6

K = 0

Score = 6

Columns: **6** 10 **6**

Sticks

Let’s break this problem down !

1. Binary search the maximum possible score
2. Create the “query” function -> Is it possible to achieve a score *x* ?
3. Solve the problem

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- It is always possible to have a score of 0
- Maximum score = max(Bags) * K *

** The maximum score is 9 instead of 30 for clarity here*

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- Find the last 1

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- Find the last 1

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	**1**	1	1	0	0	0

- Find the last 1

possible(4) = True

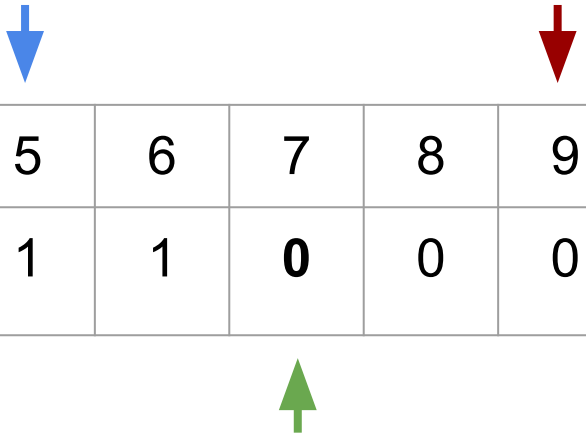
-> The score might be higher

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- Find the last 1

Sticks - Binary Search



Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	**0**	0	0

- Find the last 1

possible(7) = False

-> The score is lower

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- Find the last 1

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	**1**	1	0	0	0

- Find the last 1

possible(5) = True

-> The score might be higher

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	1	0	0	0

- Find the last 1

Sticks - Binary Search

Score	0	1	2	3	4	5	6	7	8	9
Is it possible to achieve this score ?	1	1	1	1	1	1	**1**	0	0	0

- Find the last 1

possible(6) = True

-> Found the highest score !

Sticks - Query function

Bags: 3 5 6

K = 0

Score = 0

- Is it possible to have score >= 4 ?

Columns: **0** **0** **0**

Sticks - Query function

Bags: 3 5 6

K = 1

Score = 0

- Is it possible to have score >= 4 ?

Columns: 3 **0** **0**

Sticks - Query function

Bags: 3 5 6

K = 2

Score = 0

- Is it possible to have score >= 4 ?

Columns: 6 **0** **0**

Sticks - Query function

Bags: 3 5 6

K = 3

Score = 0

- Is it possible to have score >= 4 ?

Columns: 6 5 **0**

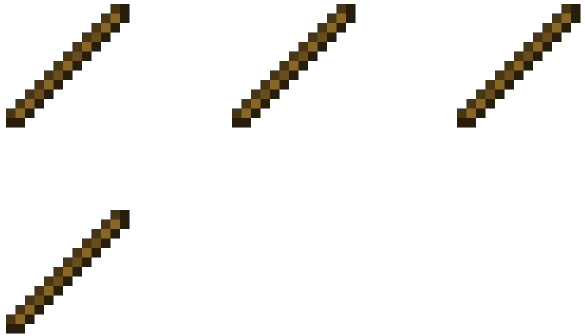
Sticks - Query function

Bags: 3 5 6

K = 4

Score = 5

- Is it possible to have score >= 4 ?



Columns: 6 **5** 6

Sticks - Query function

Bags: 3 5 6

K = 4

Score = 5

- Is it possible to have score >= 4 ?

-> Yes, we used 4 < 5 generations to have a score >= 4

Columns: 6 **5** 6

Sticks - Query function

Bags: 3

- How many generations for each column ?
- ceil(score / bag_size)

-> How many sticks to generate to have a column size >= score

- ceil(4 / 3) = ceil(1.333) = 2

-> We need 2 generations of bag 3 to have a column size >= 4

- Do this for each column, if we need less or equal than K generations, it is possible
- Done in O(N)

Columns: 6

Coding time !

- Website : https://cc618.github.io/Binary-Search-Everything
- Code stubs : https://github.com/Cc618/Binary-Search-Everything

Tournament (Piscine Tycoon)

- 2 players, students team and assistants team
- Final score is (students points + assistants points) / 2
- Students and assistants points are NOT correlated

-> A students team player will always have the same points no matter which assistant

- In the tournament, every students players will play with every assistants players
- Scores are ordered in a list
- You want to find the ***K***-th score

Tournament - Let’s reformulate

- You are given an integer **K** and two integer arrays **A** and **B**
- The score list contains all averages given all pairs from **A** and **B** in order

Example :

- A = [1, 4, 3]
- B = [5, 2, 4]
- K = 5

Score list : [1.5, 2.5, 2.5, 3, **3**, 3.5, 4, 4, 4.5]

^ **5**th one

Tournament - Let’s reformulate

Example :

- A = [1, 4, 3]
- B = [5, 2, 4]
- K = 5

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

Tournament - Intuition

Score list : [1.5, 2.5, 2.5, 3, **3**, 3.5, 4, 4, 4.5] (**K**=5)

- What does the **K**-th value mean ?
- Let's call **x** the **K**-th value, here **x**=3

-> There are < **K** values < **x**, here 3 values: 1.5, 2.5, 2.5

If we find a way to count how many values are less than any **x**,
we can directly binary search the result

-> Find the last **x** such that there are less than **K** values < **x**

Tournament

Let’s break this problem down !

1. Binary search **x**, the **K**-th value
2. Create the “query” function -> How many values are less than a given **x**
3. Solve the problem

Tournament - Binary search

- The minimum value is *(min(A) + min(B)) / 2*
- The maximum value is *(max(A) + max(B)) / 2*

Tournament - Query function

- Let's call it *count_lower*
- count_lower(x) -> How many average values are < x ?
- We can binary search *again* the first value >= x for each row

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

Tournament - Query function

Example : How many values < 3 ?

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

Tournament - Query function

Example : How many values < 3 ?

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

Tournament - Query function

Example : How many values < 3 ?

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

Tournament - Query function

Example : How many values < 3 ?

Ordered A / B	**2**	**4**	**5**
1	1.5	2.5	3
3	2.5	3.5	4
4	3	4	4.5

3 values in total :

2 values

1 value

No values

Tournament

- The main binary search function is O(log(max(A, B)) * O(count_lower))
- O(count_lower) = O(N * log(M))
- Total time complexity : O(N * log(M) * log(max(A, B)))

That’s all folks !

- Binary search 🤗

You like problem solving ?

- Codeforces : https://codeforces.com/
- Google Kickstart : https://codingcompetitions.withgoogle.com/kickstart
- Prologin : https://prologin.org/

Thanks !

Sources

- Graphics :
- www.pinclipart.com
- www.minecraft.gamepedia.com

- Inspiration :
- www.codeforces.com (ITMO Academy: pilot course)