競プロ勉強会04

elzup

二分探索

- 非常に高速
  - 線形探索 n -> 二分探索 log(n)
- 要素がソートされているリストが前提

まずは線形探索

- AOJ: Linear Search
- http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=ALDS1_4_A&lang=jp

線形探索

S1, S2, S3, S4, S5, S6 … Sn

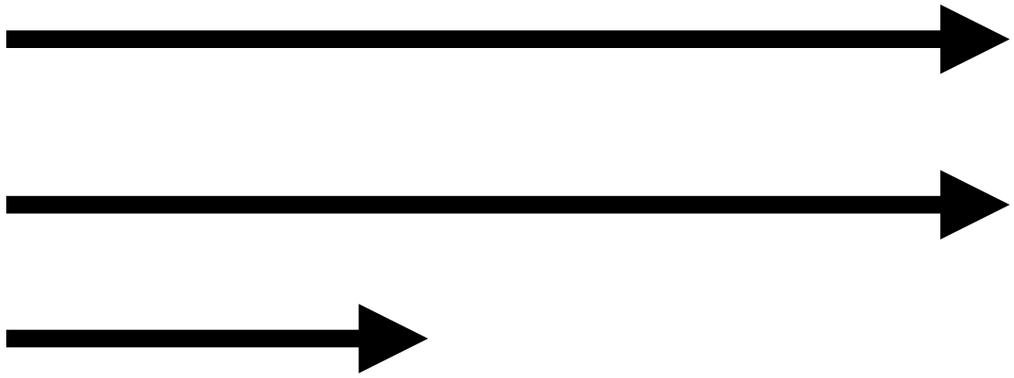
T1

T2

T3

…

Tq



n <= 10000, q <= 500

n x q = 5000000

線形探索: 実装

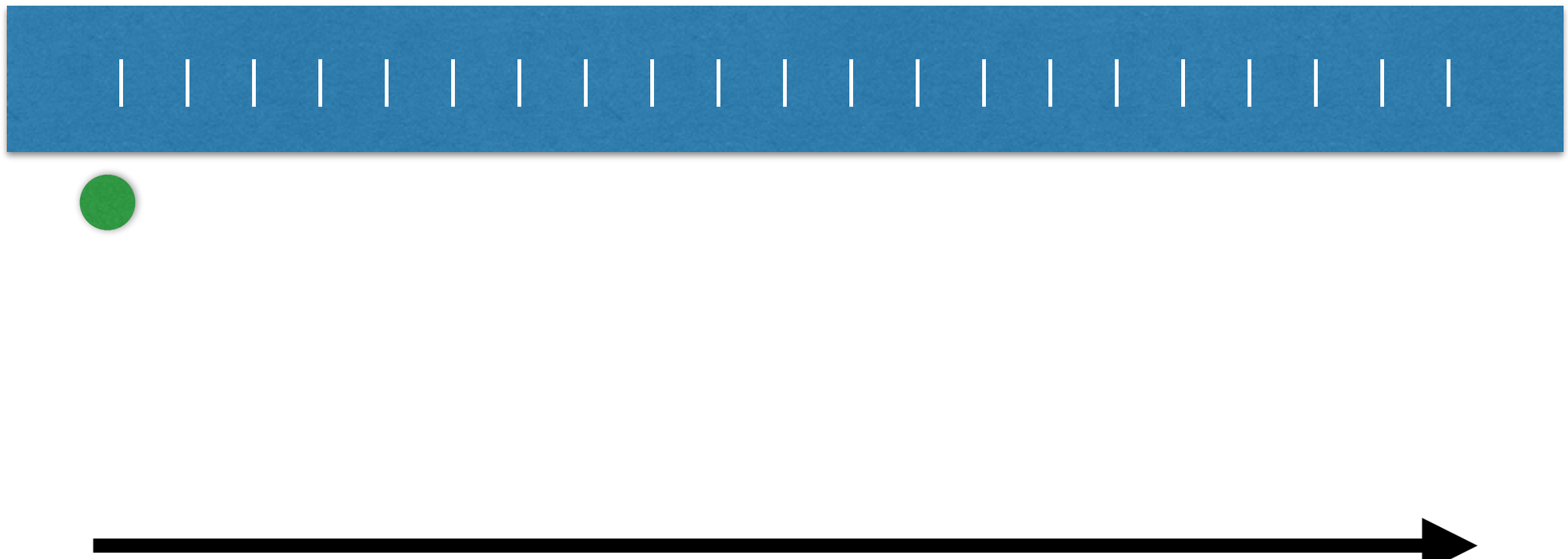
```
for (int i = 0; i < q; i++) {
    for (int j = 0; j < n; j++) {
        if (t[i] == s[j]) {
            c++;
            break;
        }}}
```

二分探索問題

- AOJ: Binary Search
- http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=ALDS1_4_B&lang=jp

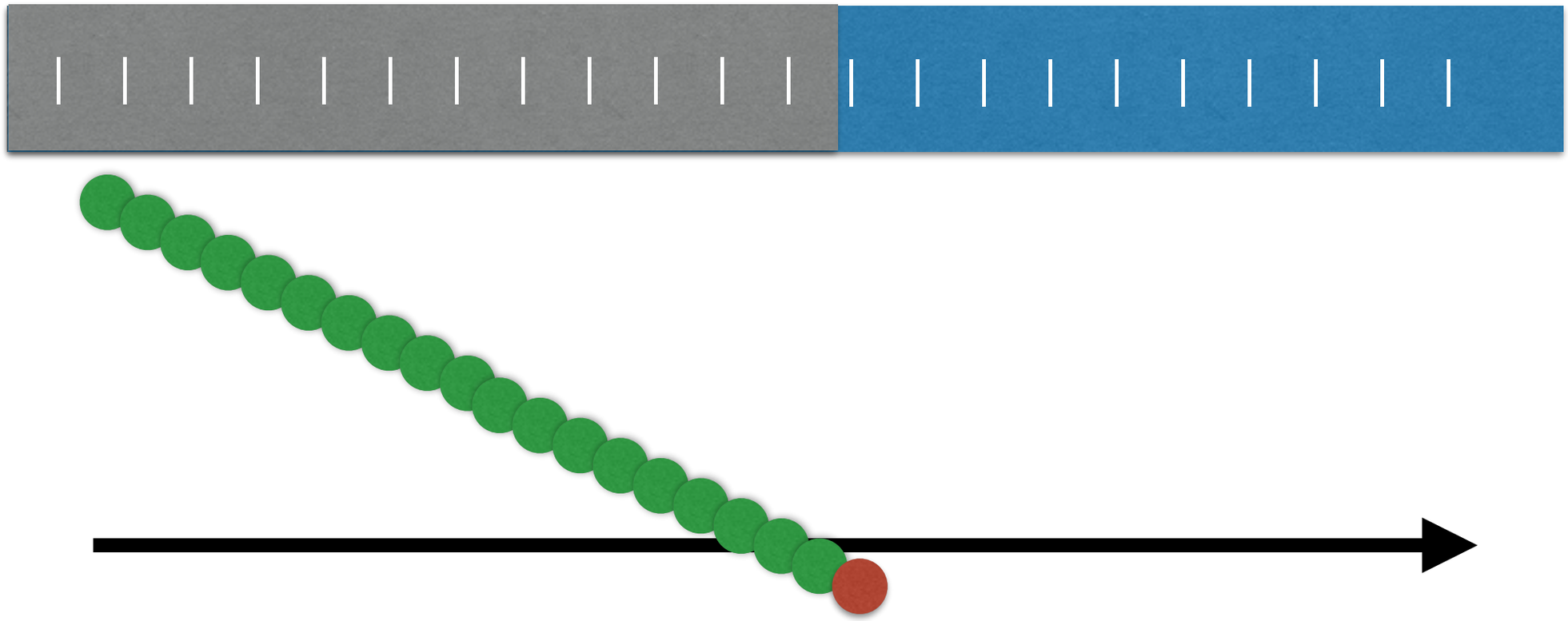
線形探索アルゴリズム

線形



線形探索アルゴリズム

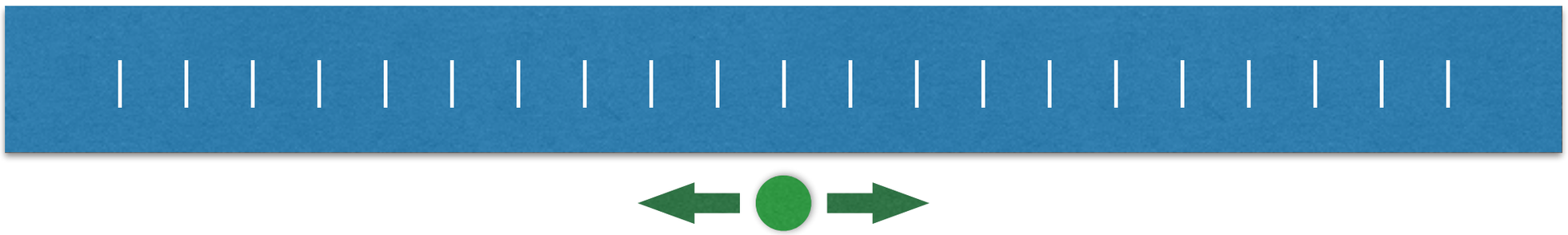
線形



二分探索アルゴリズム

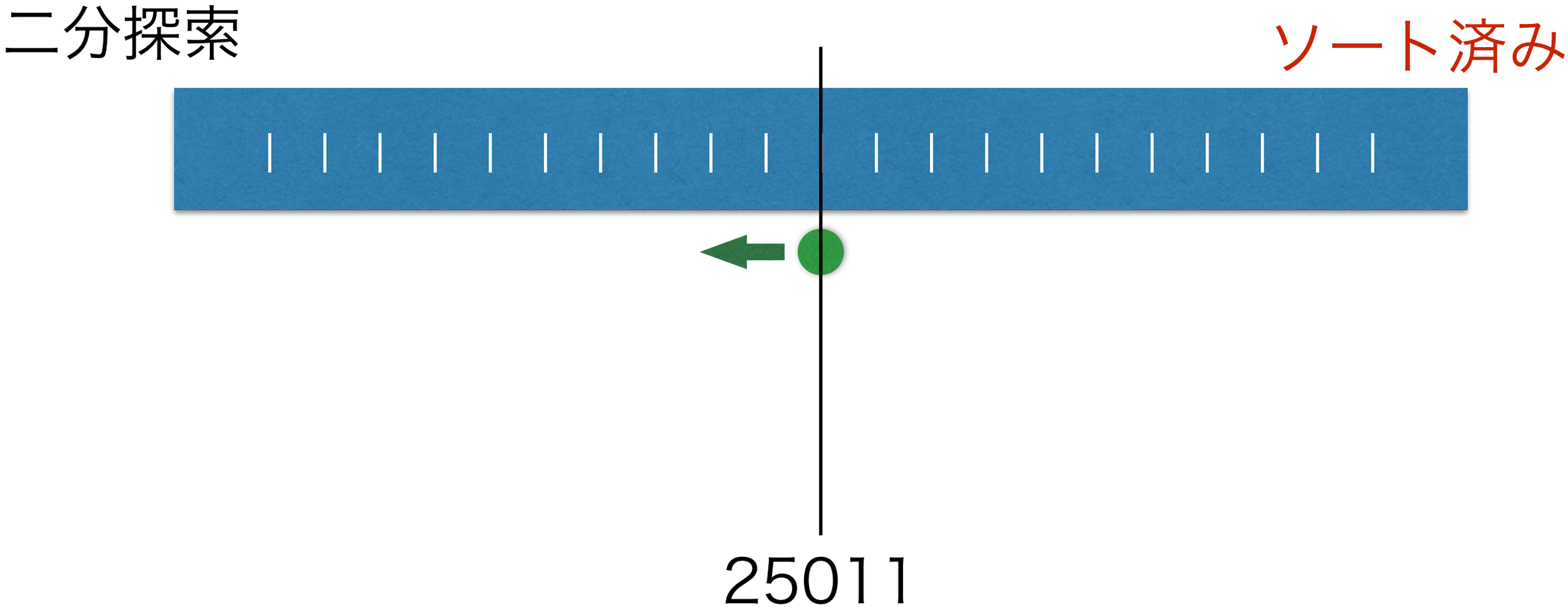
二分探索

ソート済み



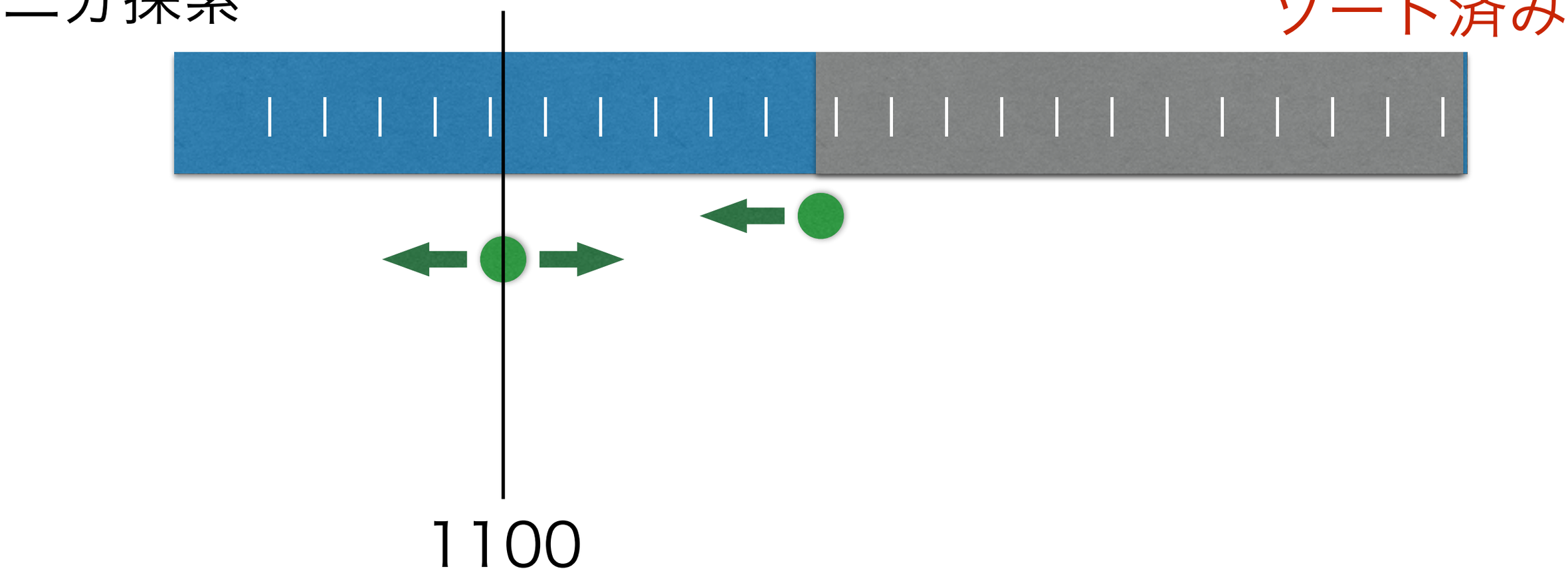
二分探索アルゴリズム

二分探索



二分探索アルゴリズム

二分探索

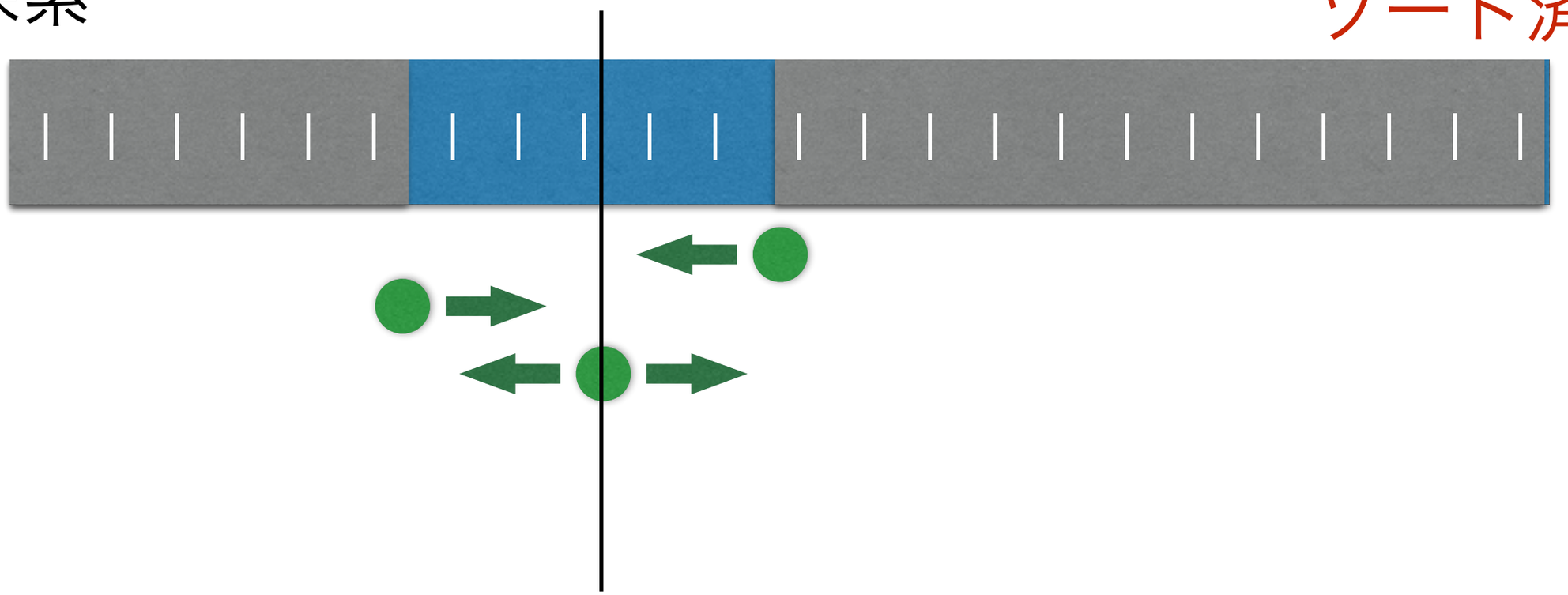


3333 < x　　3333?　　3333 < x

二分探索アルゴリズム

二分探索

ソート済み



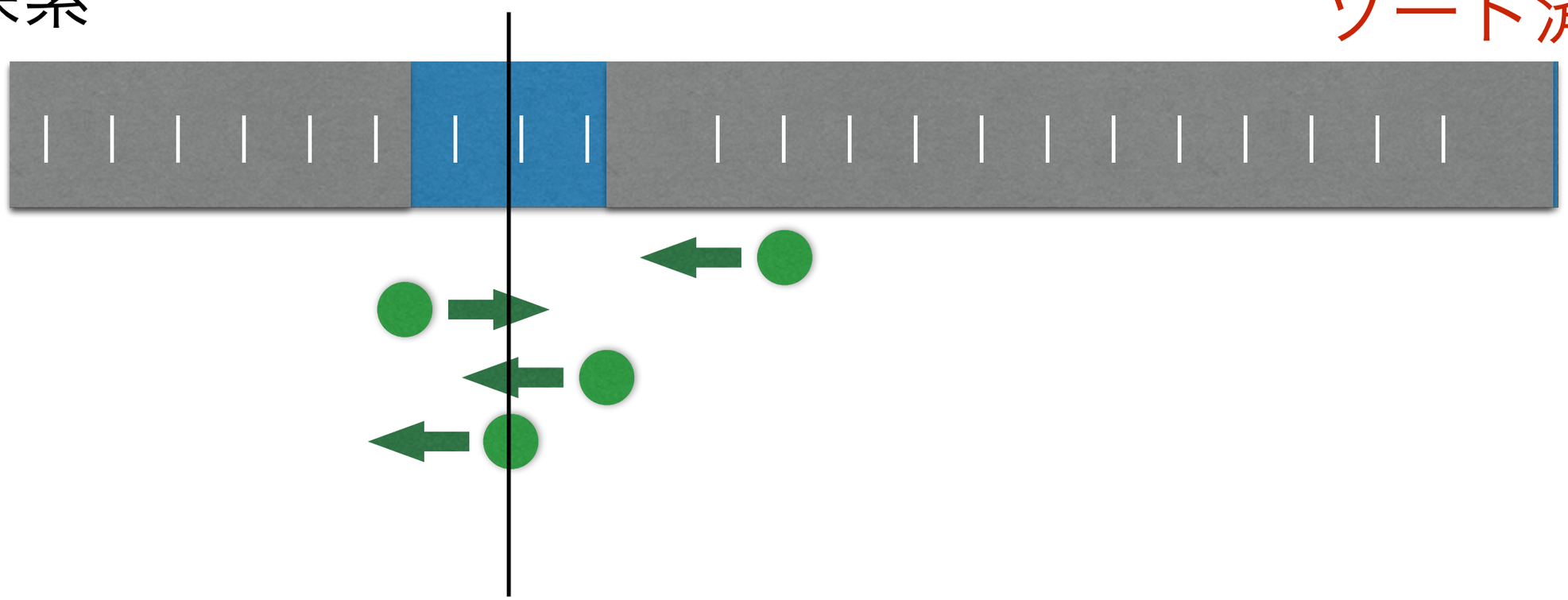
4000

3333?

二分探索アルゴリズム

二分探索

ソート済み



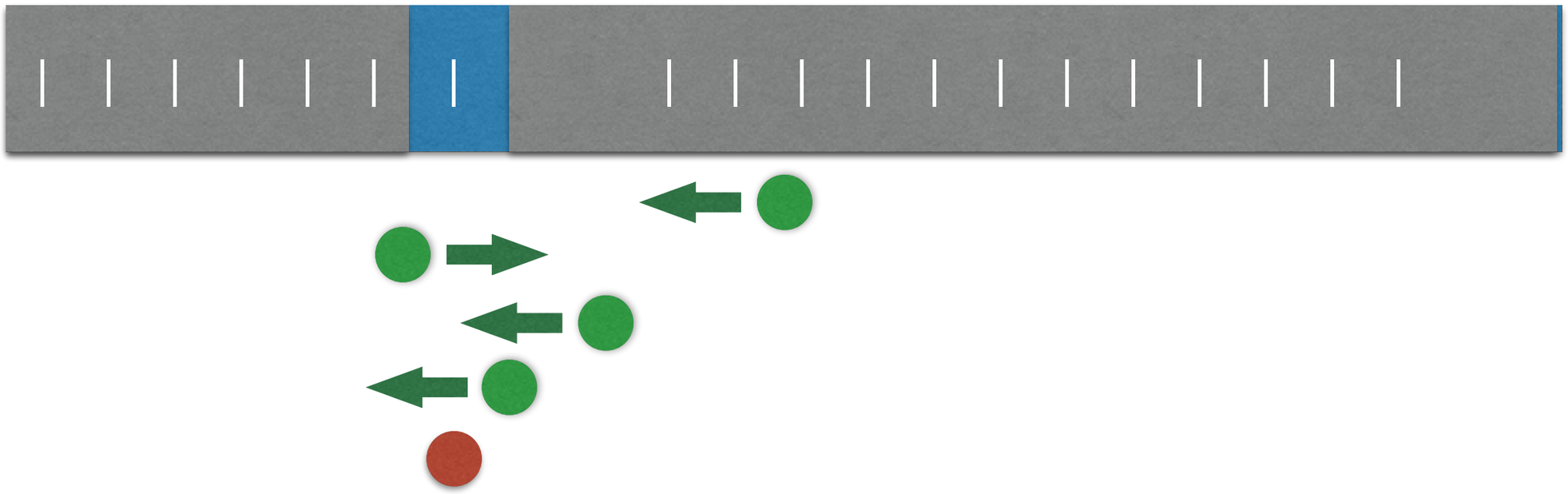
3700

3333?

二分探索アルゴリズム

二分探索

ソート済み

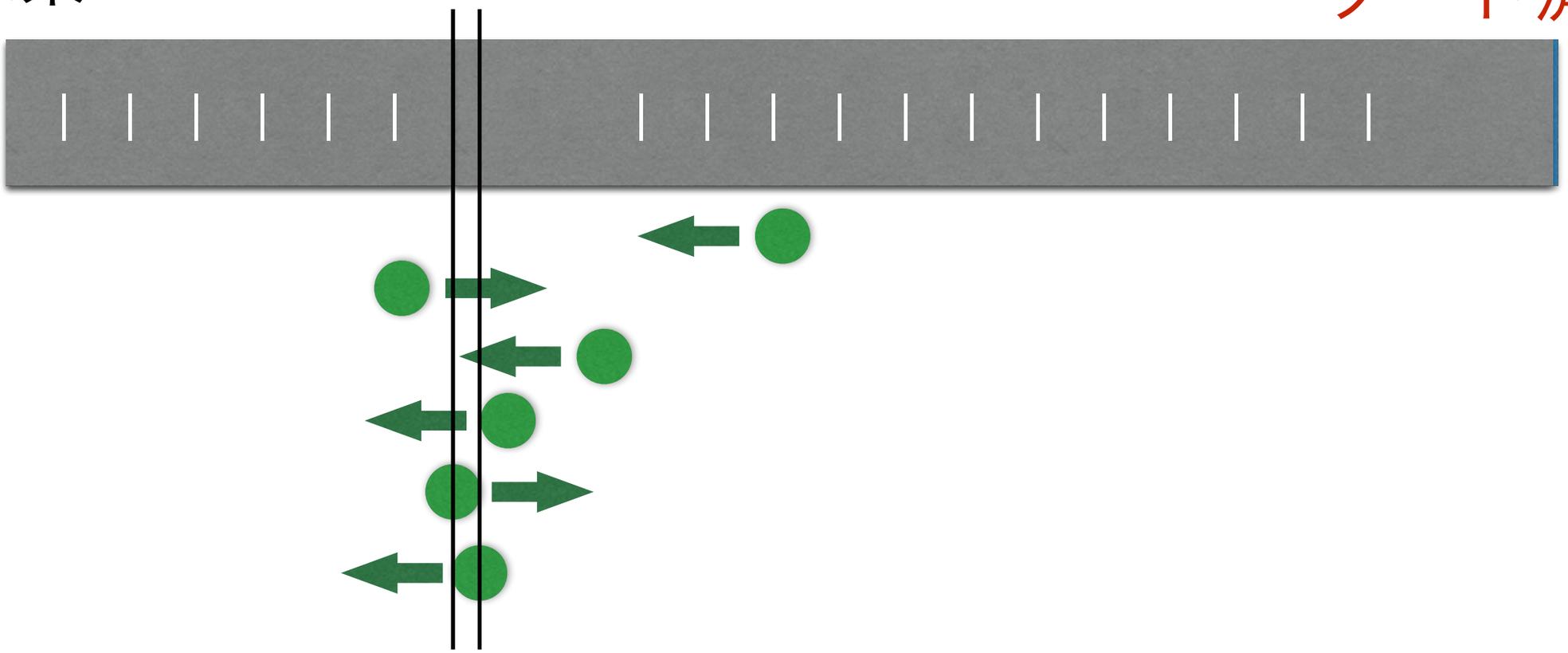


3333!

二分探索アルゴリズム
ヒットしない場合

二分探索

ソート済み



3330 3340

3333?

実装

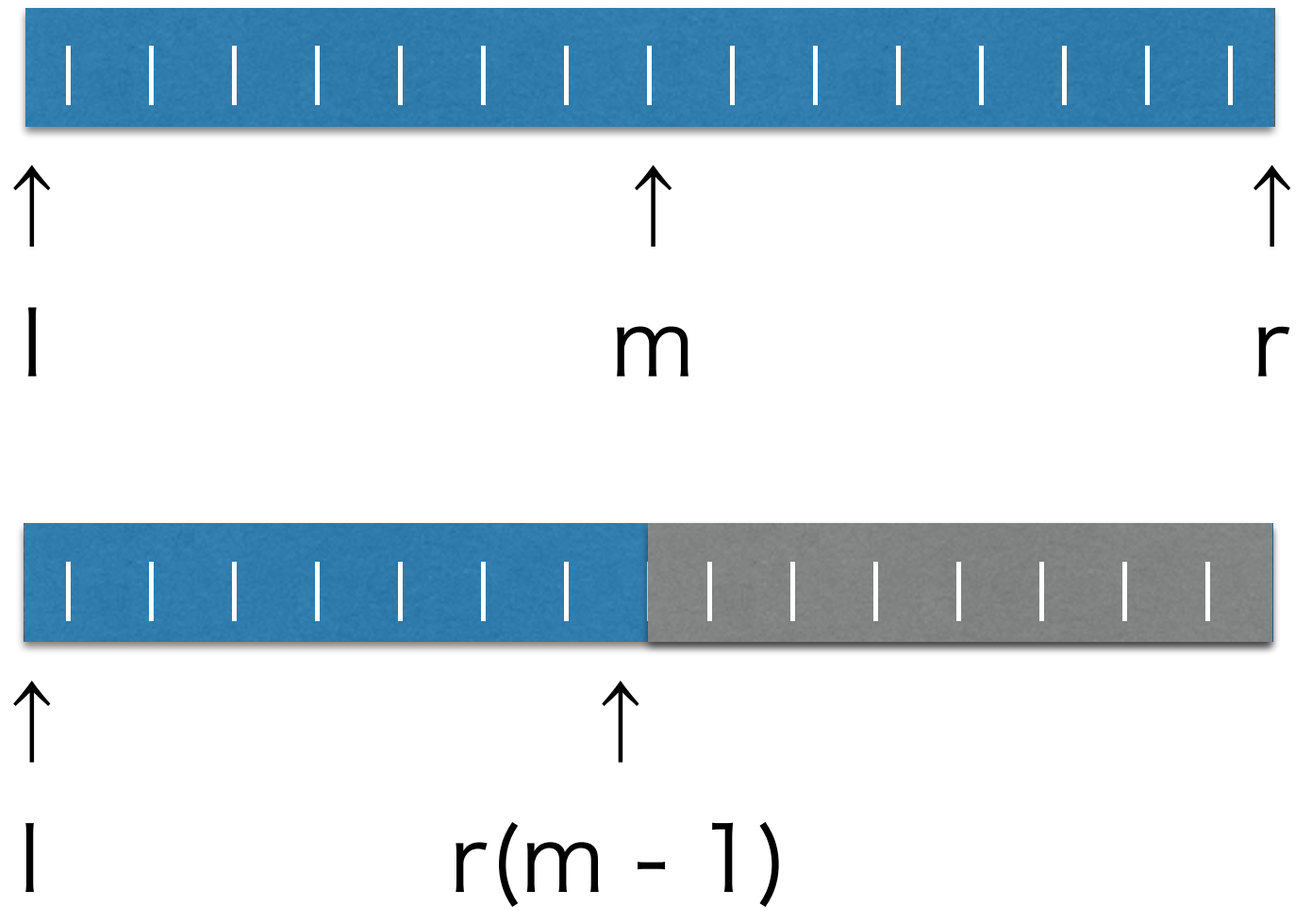
- // 初期化
- int l = 0; // left
- int r = a.length - 1; // right

実装

- while (l <= r) {
- 　　int m = (r + l) / 2;　　// 中心値の決定
- 　　if (a[m] == v) {　　// 探索した値と一致したら true で終了
- 　　return true;
- 　　}
- 　　…
- }

実装

```
while (l <= r) {
    ...
    if (a[m] > v) {
        r = m - 1;
    } else {
        l = m + 1;
    }
}
```



標準ライブラリ(java)

- Arrays.binarySearch(int[], int)
- Collections.binarySearch(List, <>)
    - あり -> index
    - なし -> ~index (bit反転)
- arrayA.retain(arrayB).size
    - -> arrayA で arrayB が含まれるものの個数

標準ライブラリ(c++)

- binary_search(int[], int) -> int

他の二分探索例

- データ内からの探索ではなく値を求める
- 円周率を求める
- 関数の境界値を求める

平方根を求める

- // どこまでの精度で求めるか
- double accuracy = 0.00000000001;
- while (r - l > accuracy) { }
- -> Order: log(10^11)

標準ライブラリ

- Java
  - Math.sqrt(double) -> double
- C++
  - sqrt(double) -> double

要素の数

- １ / 精度 に刻んでいると考えられる
- 精度: 0.001
  - 0.001, 0.002, 0.003 … 0.999, 1.000
  - から探索していると考えられる

3上根や４乗根

- double t = m * m;
- -> double t = m * m * m;
- -> double t = m * m * m * m;