# アルゴリズム講座

## 第2回 〜二分探索編〜

2020/5/16 13:00~

LeadHACK
@utsubo_21 高澤 栄一

# Lesson 2

# 今回の内容

- 予定時間割
  - 二分探索 (45分)
  - （Parametric Search (5分)）
- 備考
  - 前回とはある程度独立しているのでここからでも大丈夫だと思います.

# 講座の具体的な内容

- 下記のアルゴリズム・データ構造を10回程度に分けて１つずつ解説

Lesson1
- 全探索

Lesson2
- 二分探索
- 深さ優先探索
- 幅優先探索
- 動的計画法
- 累積和
- グラフ・木
- ダイクストラ法
- ワーシャルフロイド法
- クラスカル法
- Union-Find

参考：レッドコーダーが教える、競プロ・AtCoder上達のガイドライン【中級編：目指せ水色コーダー！】
https://qiita.com/e869120/items/eb50fdaece12be418faa

# 二分探索

# 早速ですが問題です

# 問題

- 昇順にソート済みの長さNの配列 a と整数 Wが与えられる
- 配列 a に整数 Wが含まれるか答えよ

# 問題

- 昇順にソート済みの長さNの配列 a と整数 Wが与えられる
- 配列 a に整数 Wが含まれるか答えよ

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W :　16**

# 問題

- 昇順にソート済みの長さNの配列 a と
  整数 Wが与えられる

- 配列 a に整数 Wが含まれるか答えよ

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W : 16**

➡ **16はあるので, Yes**

# 問題

- 昇順にソート済みの長さNの配列 a と
  整数 Wが与えられる

- 配列 a に整数 Wが含まれるか答えよ

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W : 16**

**どのように解くか考えてみましょう！**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W： 16**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W :　16**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W :　16**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**整数 W： 16**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

あった！

**整数 W： 16**

# 解法（全探索）

- まずは全探索してみます
- 配列 a 内の各要素を見ていき，
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**あった！**

**整数 W :　16**

**計算量はO(N)**

**Wがa内にない場合，N回比較が必要**

# 解法（全探索）

- まずは全探索してみます

- 配列 a 内の各要素を見ていき,
  それぞれWと一致するか確認します

**配列 a :**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 |

**あった！**

**整数 W： 16**

**計算量をもっと減らせる？**

**→「ソート済み」であることを活かせてない**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**① まずは真ん中の要素を確認**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**① まずは真ん中の要素を確認**

**「6 (検索対象) < 31 (5番目の要素)」**

**→ ソート済みなため，31より右には31以上しかない**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**① まずは真ん中の要素を確認**

**「6 (検索対象) ＜ 31 (5番目の要素)」**

→ ソート済みなため，**31より右には31以上しかない**

**つまり，31より左を探せば良い！**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**② 絞りこんだ範囲の真ん中を確認**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**② 絞りこんだ範囲の真ん中を確認**

**「6 (検索対象) ＜ 11 (2番目の要素)」**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**② 絞りこんだ範囲の真ん中を確認**

**「6 (検索対象) ＜ 11 (2番目の要素)」**

**→ ソート済みなため，11より右には11以上しかない**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**③ 絞りこんだ範囲の真ん中を確認**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**③ 絞りこんだ範囲の真ん中を確認**

**「2 (0番目の要素) < 6 (検索対象)」**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**③ 絞りこんだ範囲の真ん中を確認**

**「2 (0番目の要素) ＜ 6 (検索対象)」**

**→ ソート済みなため，2より左には2以下しかない**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

**④ 絞りこんだ範囲の真ん中を確認**

**ない！**

# 二分探索で解いてみる

**W： 6 があるか探す** **N=10**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 11 | 16 | 20 | 31 | 43 | 51 | 54 | 69 |

**④ 絞りこんだ範囲の真ん中を確認**

**ない！**

**計算量を見積もってみよう！**

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - 例を見ると，1回の比較で探索範囲を半分に絞れている

| 比較回数 | 探索範囲の長さ |
|---|---|
| 1 | 10 |
| 2 | 5 |
| 3 | 2 |
| 4 | 1 |

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - 例を見ると，1回の比較で探索範囲を半分に絞れている

| 比較回数 | 探索範囲の長さ |
|---|---|
| 1 | N |
| 2 | N / 2 |
| 3 | N / 4 |
| i | N / 2 ^ (i - 1) |

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - i 回比較した時，探索範囲は N / 2^(i -1)
  - i が何回で探索範囲が絞りきれるか？

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - i 回比較した時，探索範囲は N / 2^(i -1)
  - i が何回で探索範囲が絞りきれるか？

**探索範囲が1まで絞れれば良いので,**

$$\frac{N}{2^{(i-1)}} = 1$$

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - i 回比較した時，探索範囲は N / 2^(i -1)
  - i が何回で探索範囲が絞りきれるか？

**探索範囲が1まで絞れれば良いので,**

$$\frac{N}{2^{(i-1)}} = 1$$

$$\log N = \log 2^{(i-1)}$$

$$i = \frac{\log N}{\log 2} + 1$$

# 二分探索の計算量

- 何回の比較が必要か見積もる
  - i 回比較した時，探索範囲は N / 2^(i -1)
  - i が何回で探索範囲が絞りきれるか？

**探索範囲が1まで絞れれば良いので,**

$$\frac{N}{2^{(i-1)}} = 1$$

$$\log N = \log 2^{(i-1)}$$

$$i = \frac{\log N}{\log 2} + 1$$

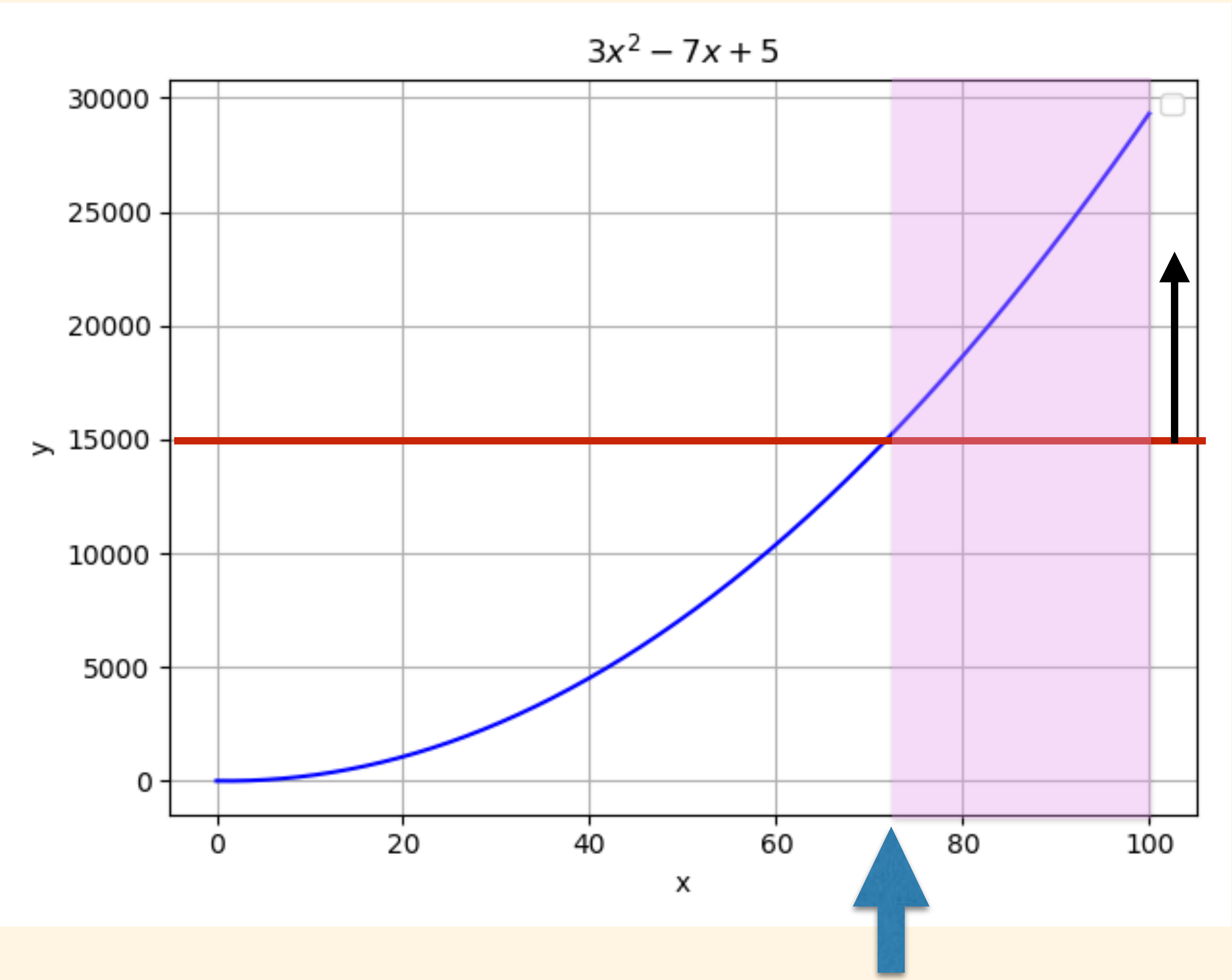
➡ **O(logN)**

# 二分探索の応用例

- 概要
  - 単調増加の関数f(x)の出力が基準値以上になる最小の入力xを得る

- 例
  - 単調増加の関数： $f(x) = 3x^2 - 7x + 5 \quad (0 \leq x \leq 100)$
  - $f(x) \geq 15000$ となる最小のxを探す

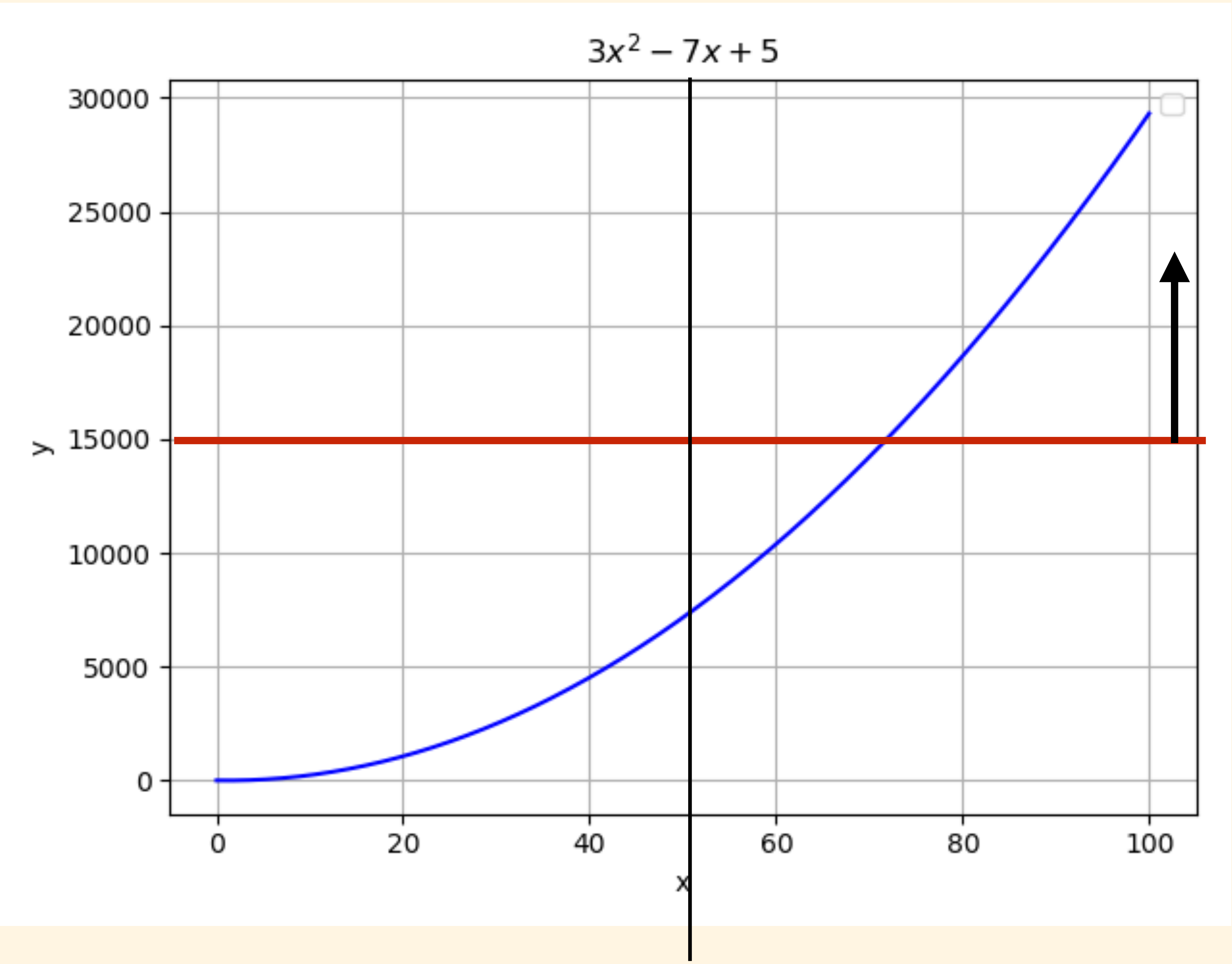
# 二分探索の応用例




基準値(15000)
以上の範囲

条件の満たし始めの部分を知りたい

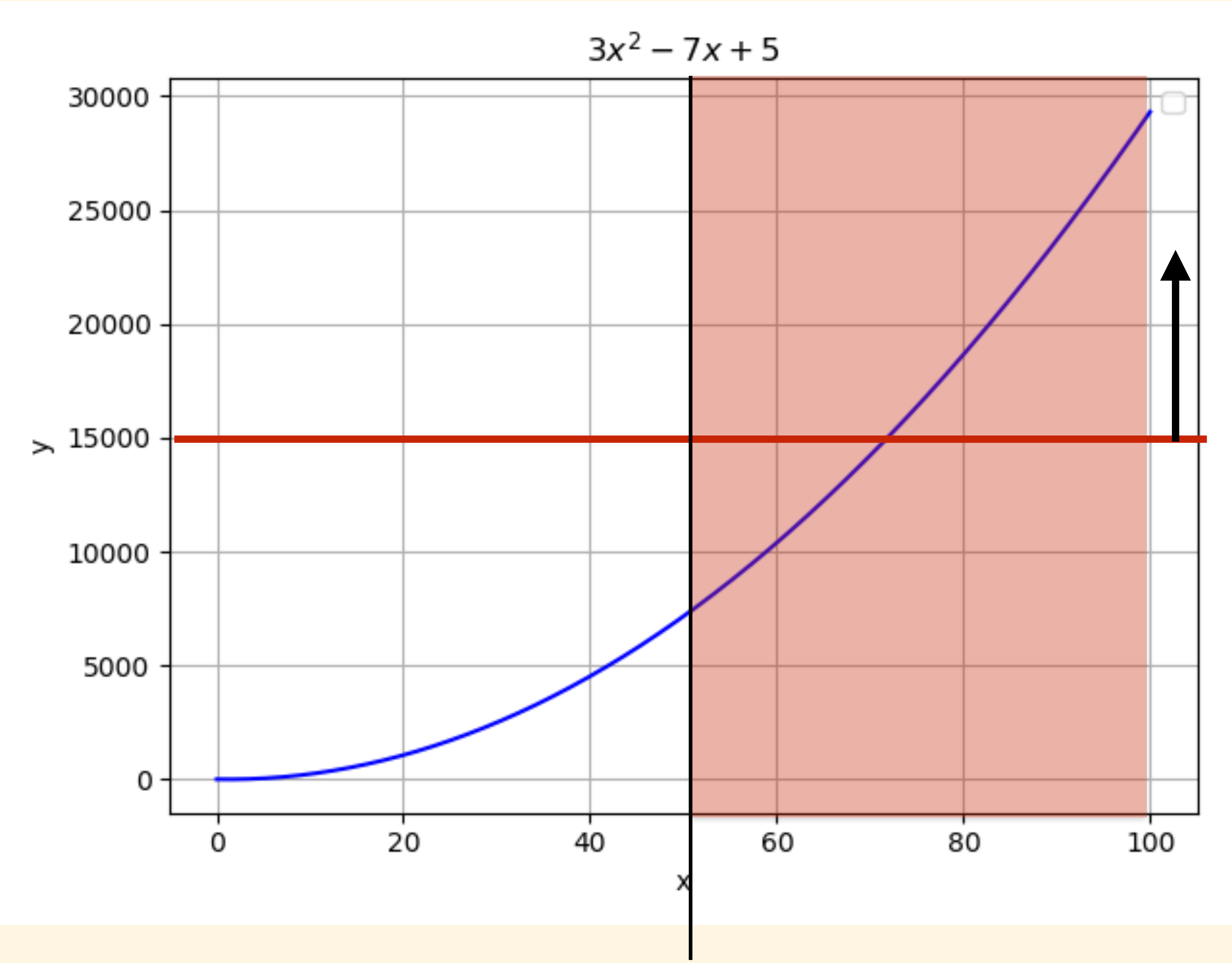
# 二分探索の応用例




**基準値(15000)**
**以上の範囲**

$$f(50) = 7155$$

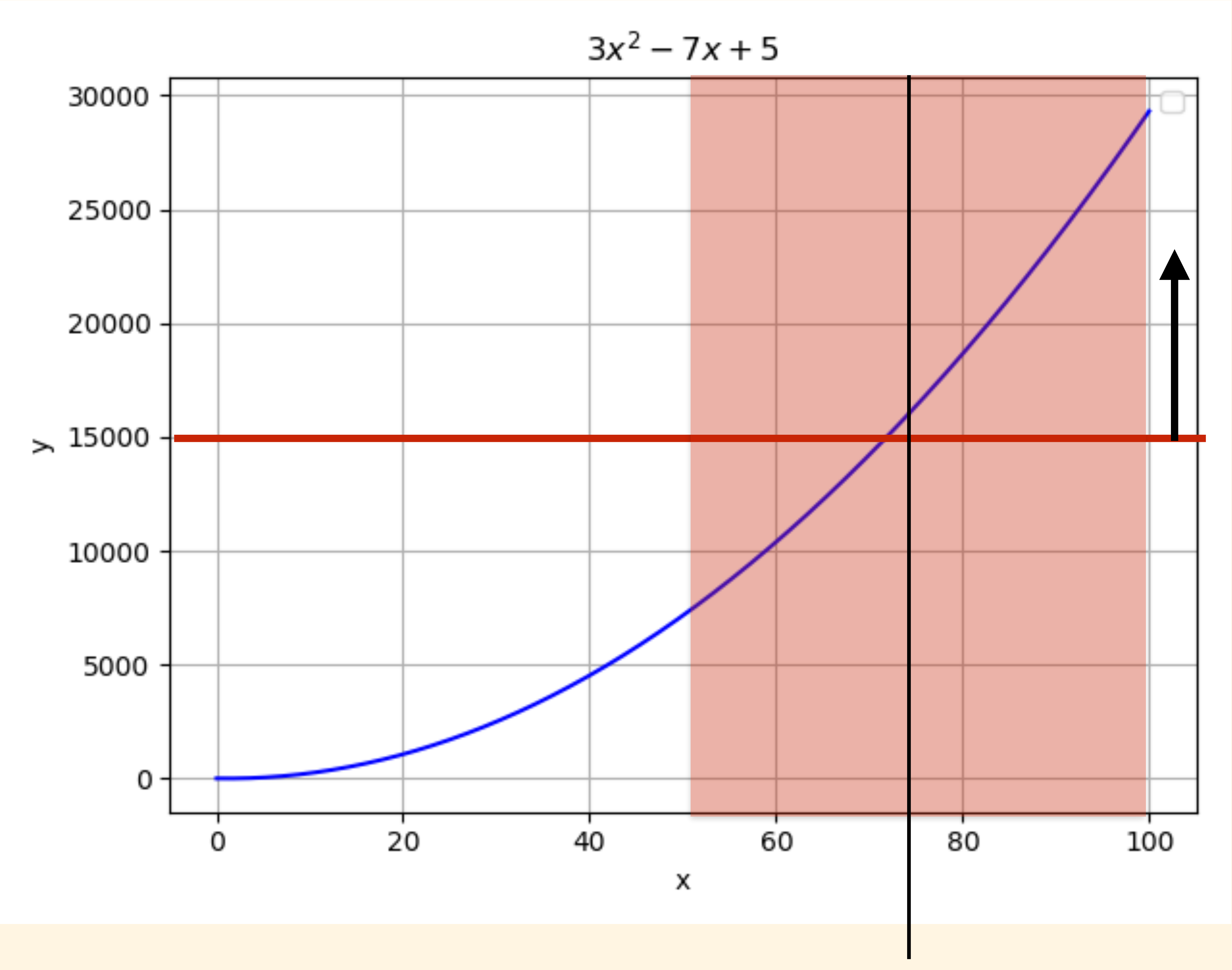
# 二分探索の応用例




基準値(15000)
以上の範囲

**f(x)≧15000を最初に満たすxはこれより右**

# 二分探索の応用例

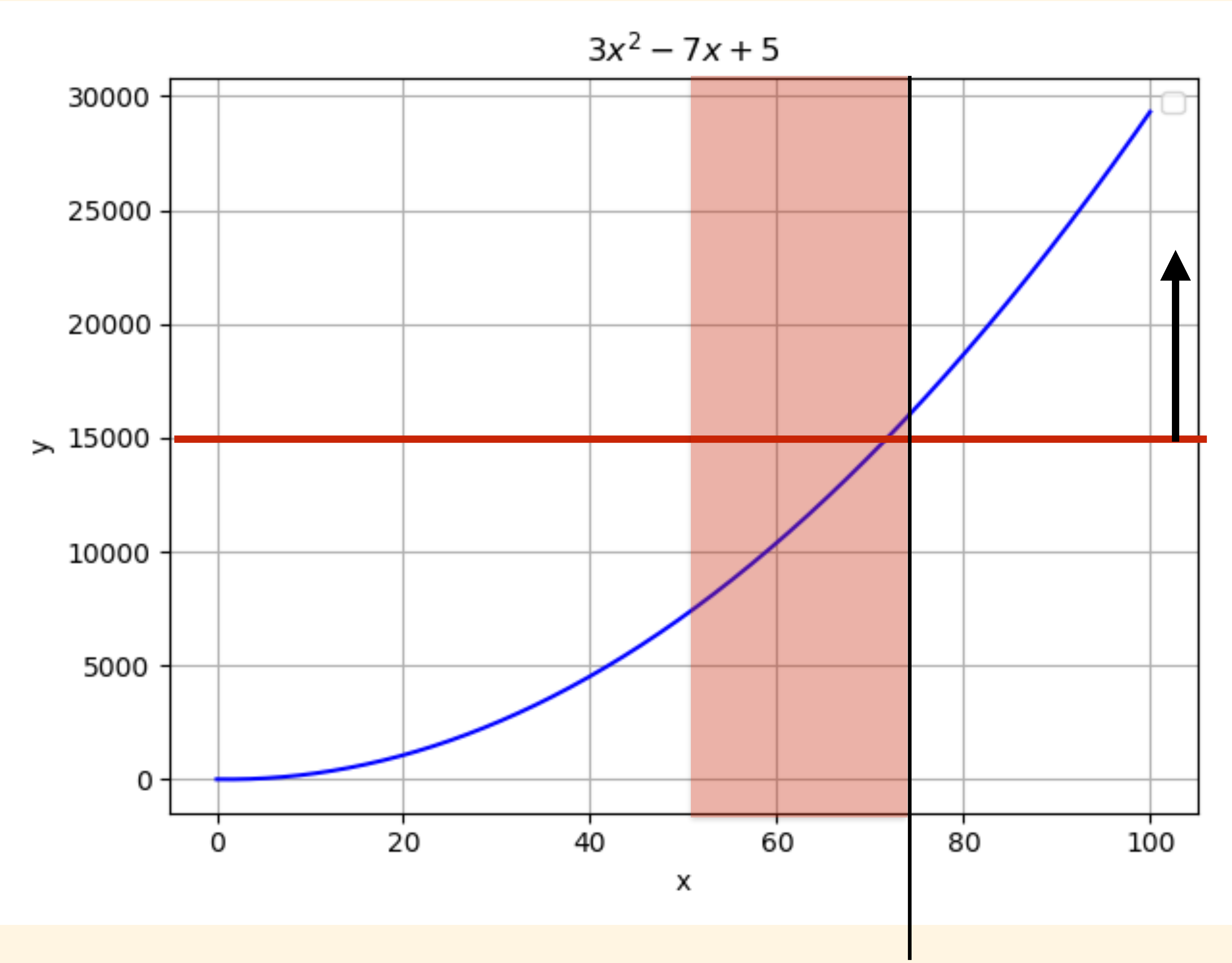



**基準値(15000)**

**以上の範囲**

$f(75) = 16355$

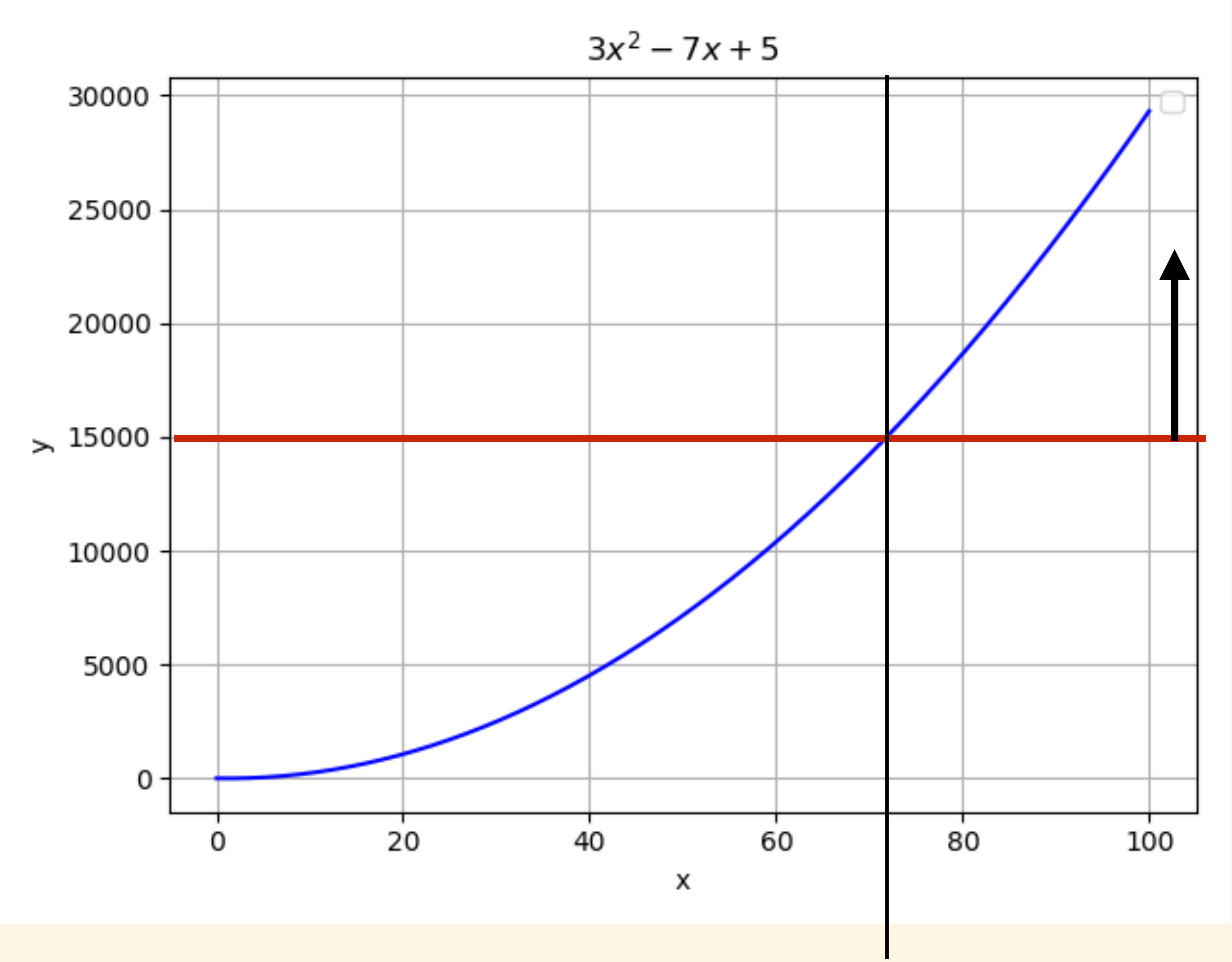
# 二分探索の応用例




基準値(15000)
以上の範囲

f(x)≧15000を最初に満たすxはこれより左

# 二分探索の応用例

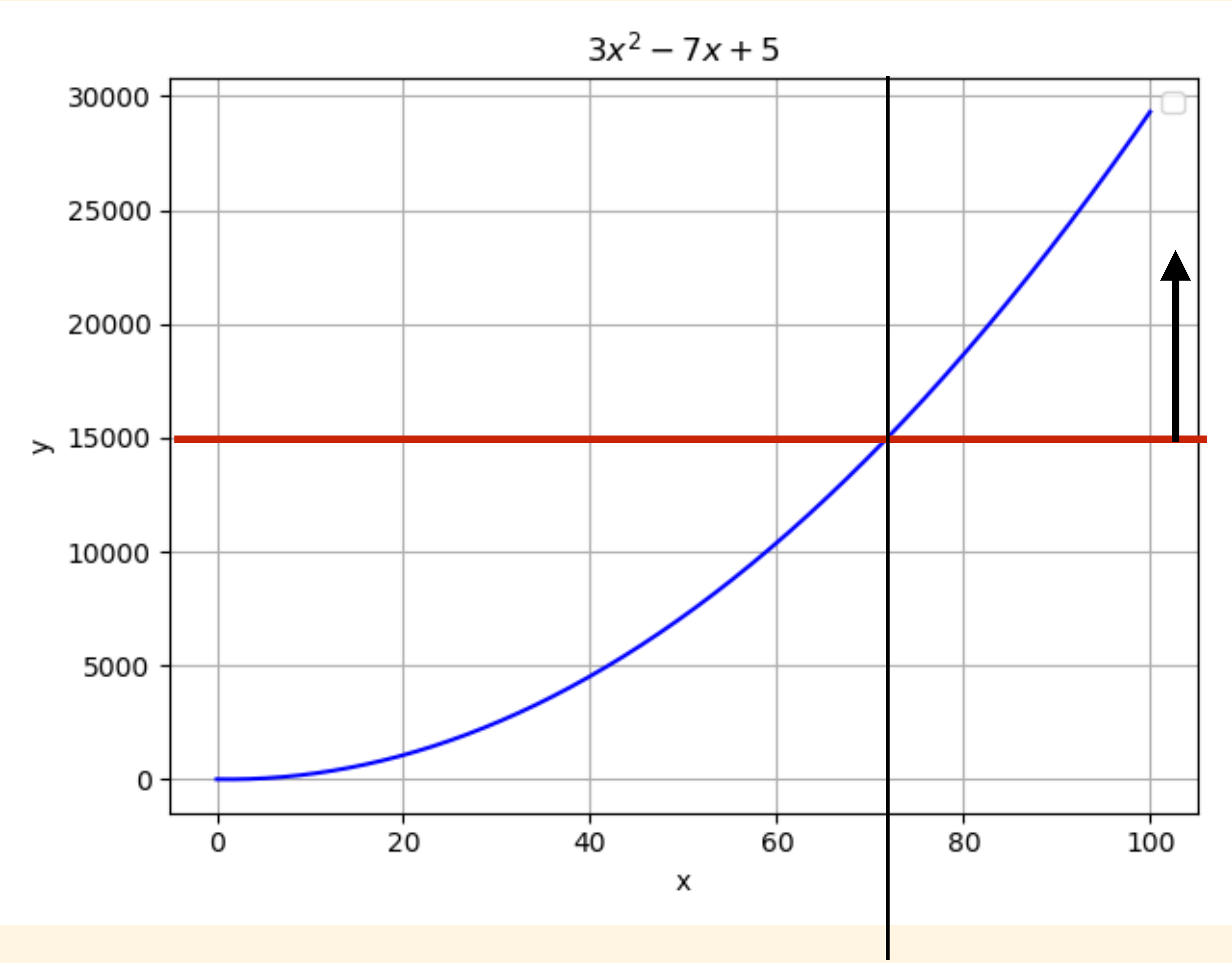



基準値(15000)

以上の範囲

十分に繰り返せば，f(x) ≧ 15000 となる最小のxが分かる

# 二分探索の応用例




**基準値(15000)以上の範囲**

**終了条件は，|f(x) - 15000| < ε のようにしておく**

# 二分探索の実装法

```
eps = 1e-6      // 誤差の設定
left = 0        // 探索範囲の設定
right = 100

while (true) {
  mid = (left + right) / 2; // 探索範囲の中点を取る
  if (f(mid) < 15000)
    left = mid;  // 左側を狭める
  else
    right = mid; // 右側を狭める
  // 誤差未満になったら終了
  if (abs(f(mid) - 15000) < eps) return mid;
}
```

# 二分探索の実装法

```
eps = 1e-6     // 誤差の設定
left = 0       // 探索範囲の設定
right = 100

while (true) {
  mid = (left + right) / 2; // 探索範囲の中点を取る
  if (f(mid) < 15000)
    left = mid;  // 左側を狭める
  else
    right = mid; // 右側を狭める
  // 誤差未満になったら終了
  if (abs(f(mid) - 15000) < eps) return mid;
}
```

# 二分探索の実装法

```
eps = 1e-6      // 誤差の設定
left = 0        // 探索範囲の設定
right = 100

while (true) {
  mid = (left + right) / 2; // 探索範囲の中点を取る
  if (f(mid) < 15000)
    left = mid;  // 左側を狭める
  else
    right = mid; // 右側を狭める
  // 誤差未満になったら終了
  if (abs(f(mid) - 15000) < eps) return mid;
}
```

# 二分探索の実装法

```
eps = 1e-6     // 誤差の設定
left = 0       // 探索範囲の設定
right = 100

while (true) {
  mid = (left + right) / 2; // 探索範囲の中点を取る
  if (f(mid) < 15000)
    left = mid;  // 左側を狭める
  else
    right = mid; // 右側を狭める
  // 誤差未満になったら終了
  if (abs(f(mid) - 15000) < eps) return mid;
}
```

# 二分探索の応用例

- 問題
  - 赤い花R本，青い花B本持っている
  - 下記の2種類の花束を作ることが出来る
    - ①: **x**本の赤い花　&　1本の青い花
    - ②: 1本の赤い花　&　**y**本の青い花
  - 最大いくつの花束を作ることが出来るか？
- 制約
  - R, B ≦ 10^18
  - x, y ≦ 10^9

# 二分探索の応用例

- 入力例
  - 赤い花**5**本，青い花**5**本持っている
  - 下記の2種類の花束を作ることが出来る
    - ①: **3**本の赤い花　&　1本の青い花
    - ②: 1本の赤い花　&　**4**本の青い花
  - 最大いくつの花束を作ることが出来るか？
- 解答
  - 2
    - ①：1個 (赤: 3本, 青:1本)
    - ②：1個 (赤: 1本, 青: 4本)

# 二分探索の応用例

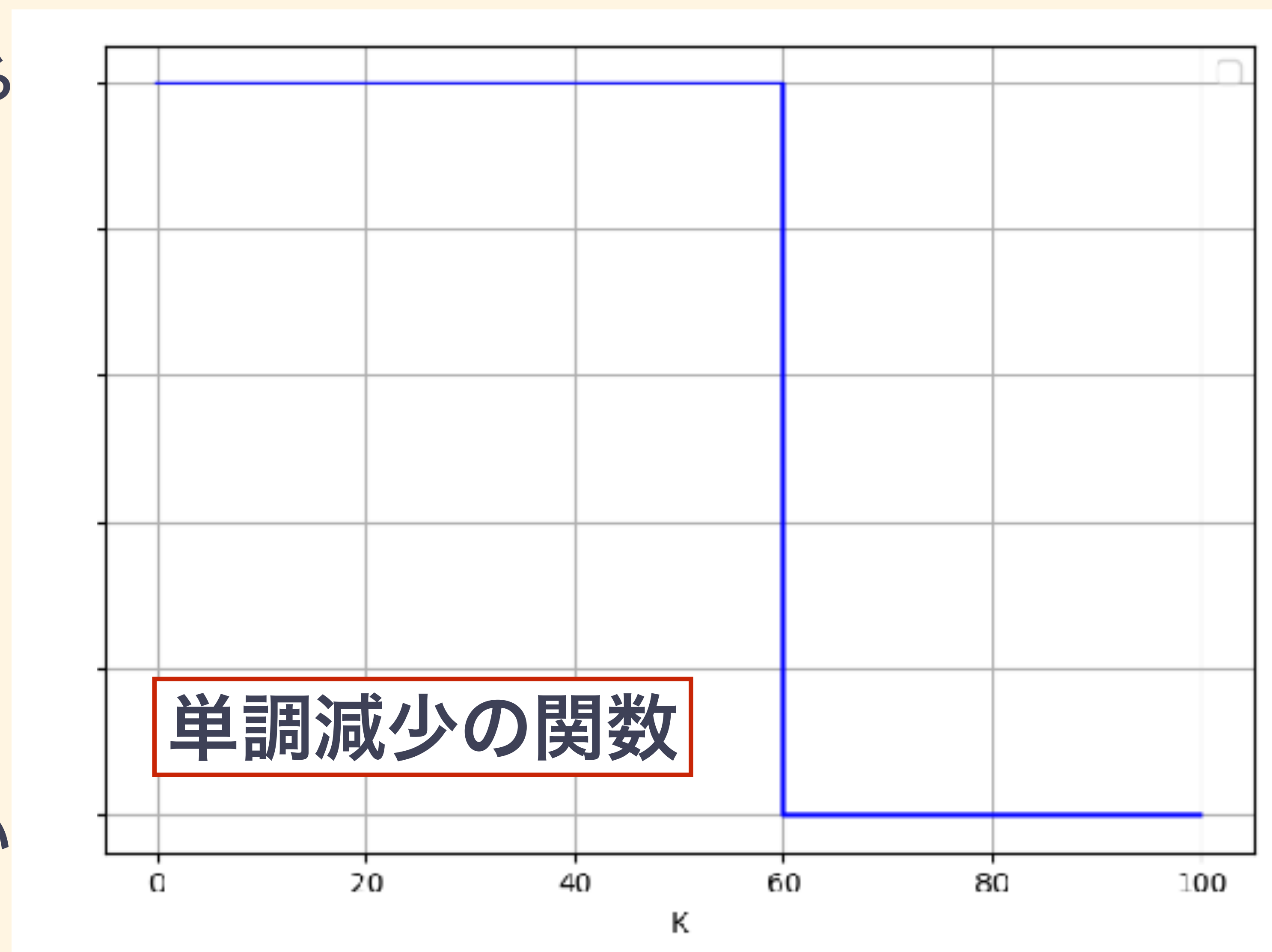
- 考察
  - 「K個の花束を作ることが出来るか？」という Yes / No問題を考える
    - 花束①，花束②どちらを作るとしても，赤い花，青い花が1本ずつ必要
    - 残りの花の数は，赤: R - K本，青: B - K本
    - 花束①を作ることが出来る個数は (R - K) / (x - 1)個
    - 花束②を作ることが出来る個数は (B - K) / (y - 1)個
    - (R - K) / (x - 1) + (B - K) / (y - 1) ≧ Kであれば，Yes
  - この問題であれば，高速に答えられそう！

# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」という Yes / No問題を考える
  - 「K個の花束が作ることが出来る」とき, 「K個以下の花束も作ることが出来る」
  - 「K個の花束を作ることが出来るか？」が Yesとなる最大のKを探す

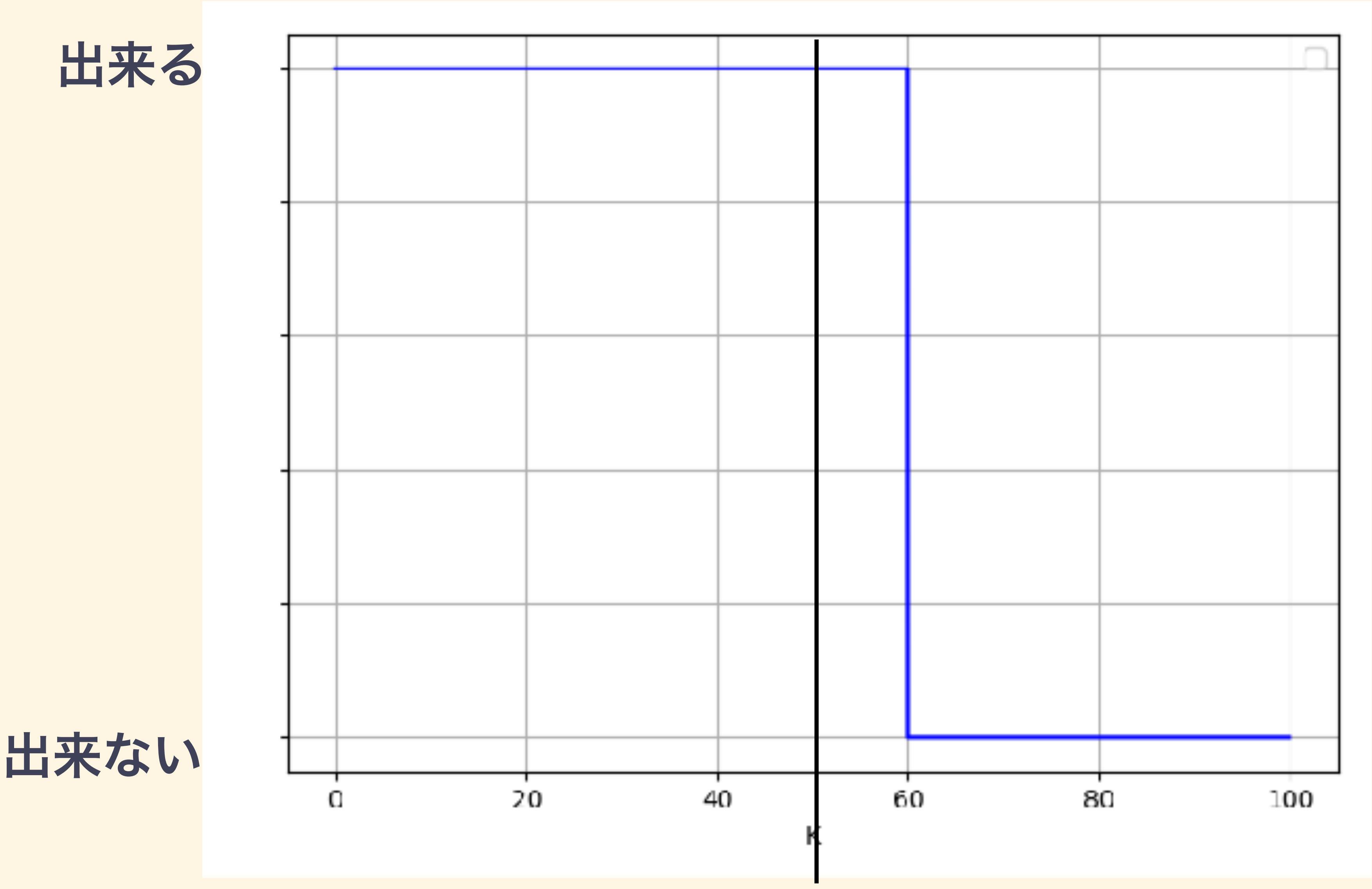
# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す



# 二分探索の応用例

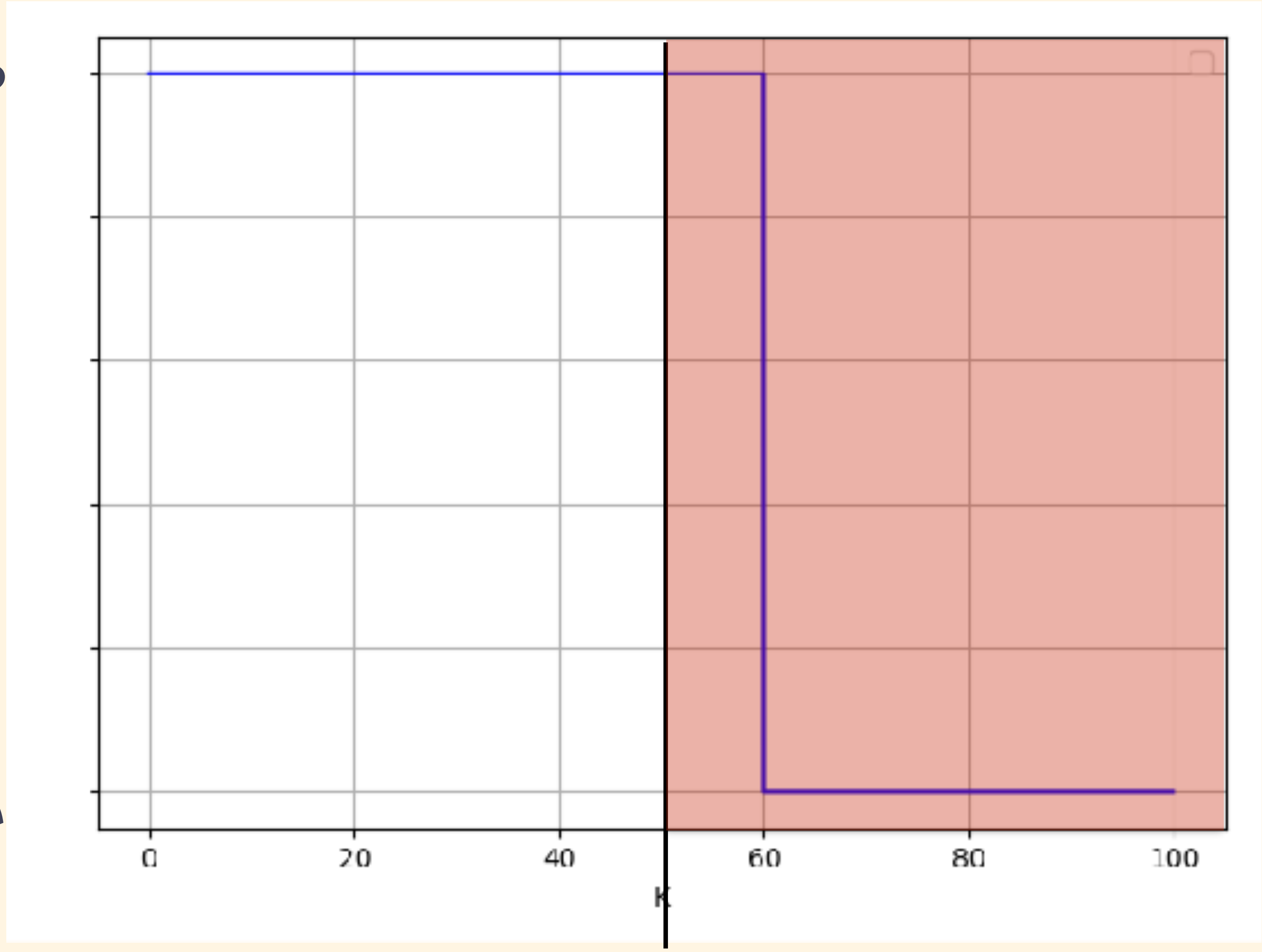
- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す



# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す

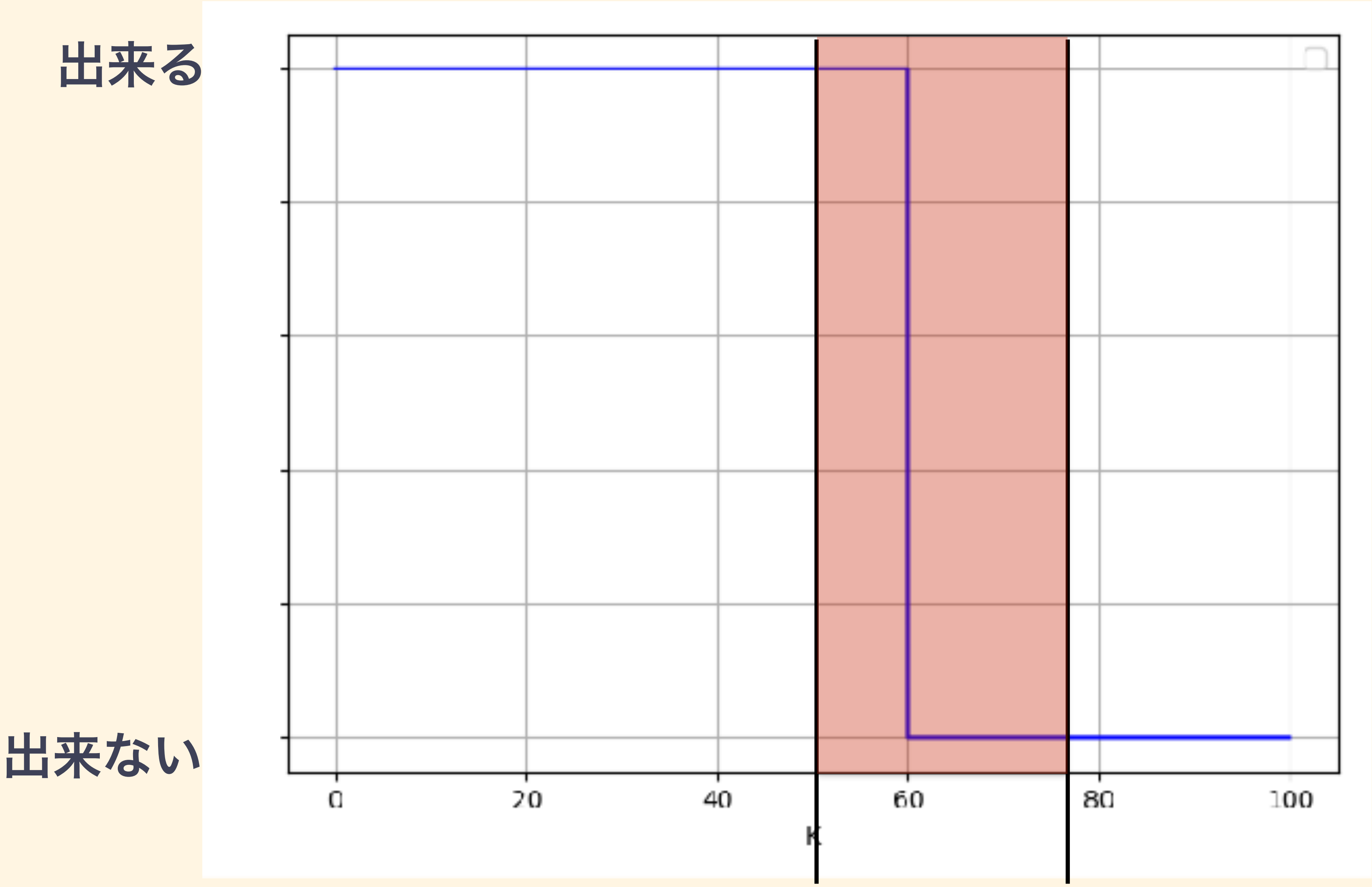
出来る




出来ない

# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す

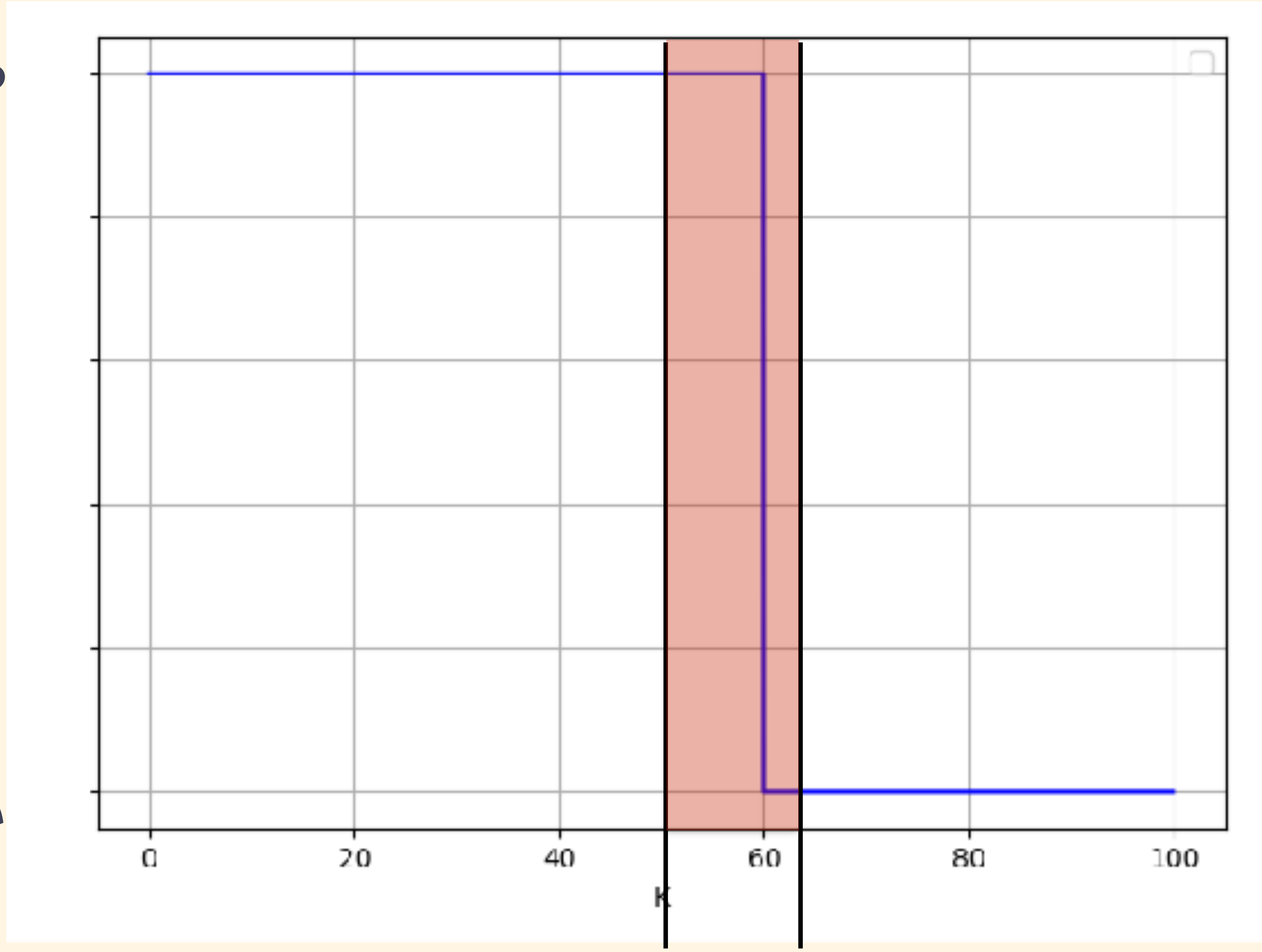


# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す

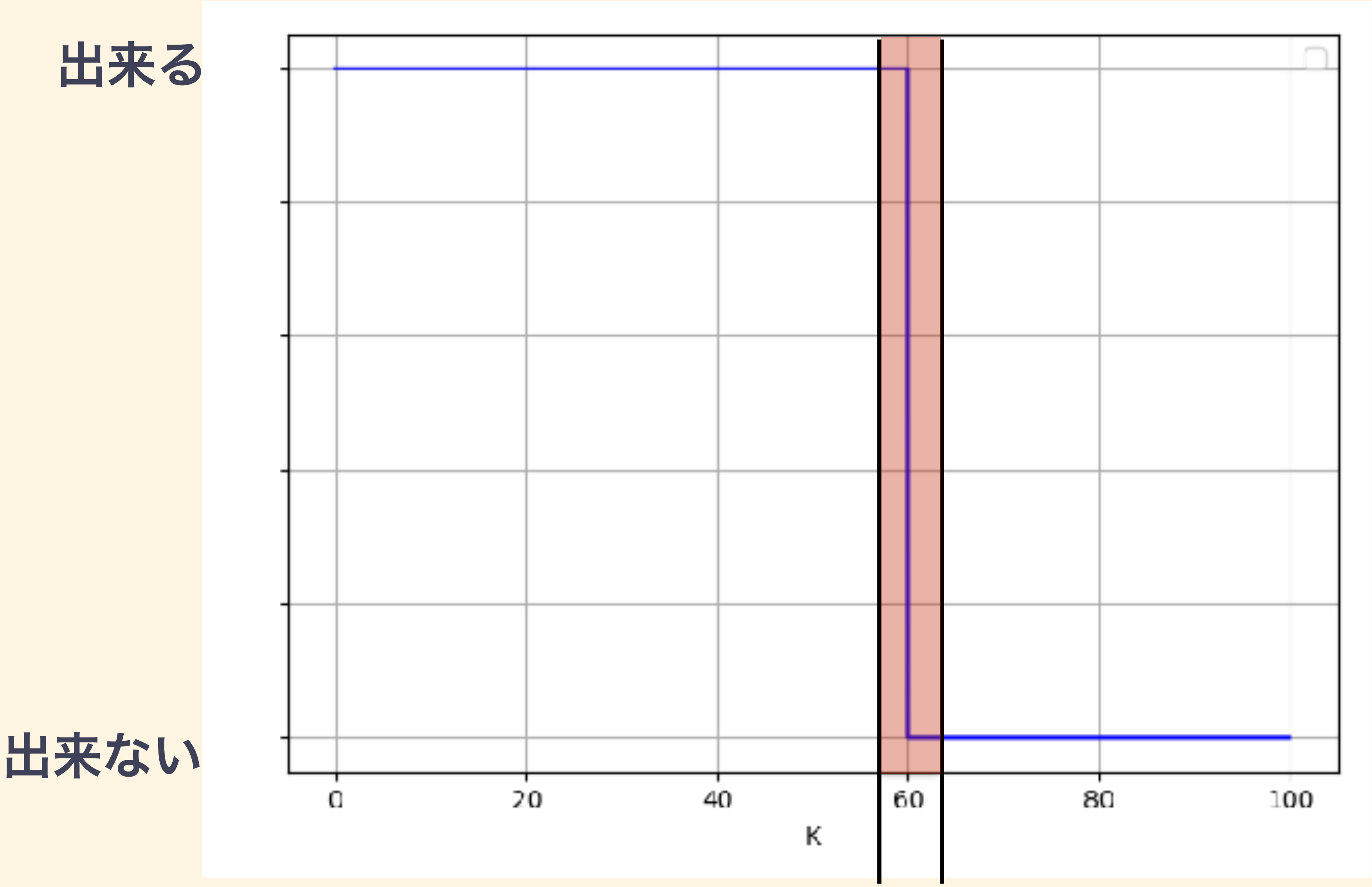
出来る

出来ない



# 二分探索の応用例

- 考察
  - 「K個の花束を作ることが出来るか？」がYesとなる最大のKを探す



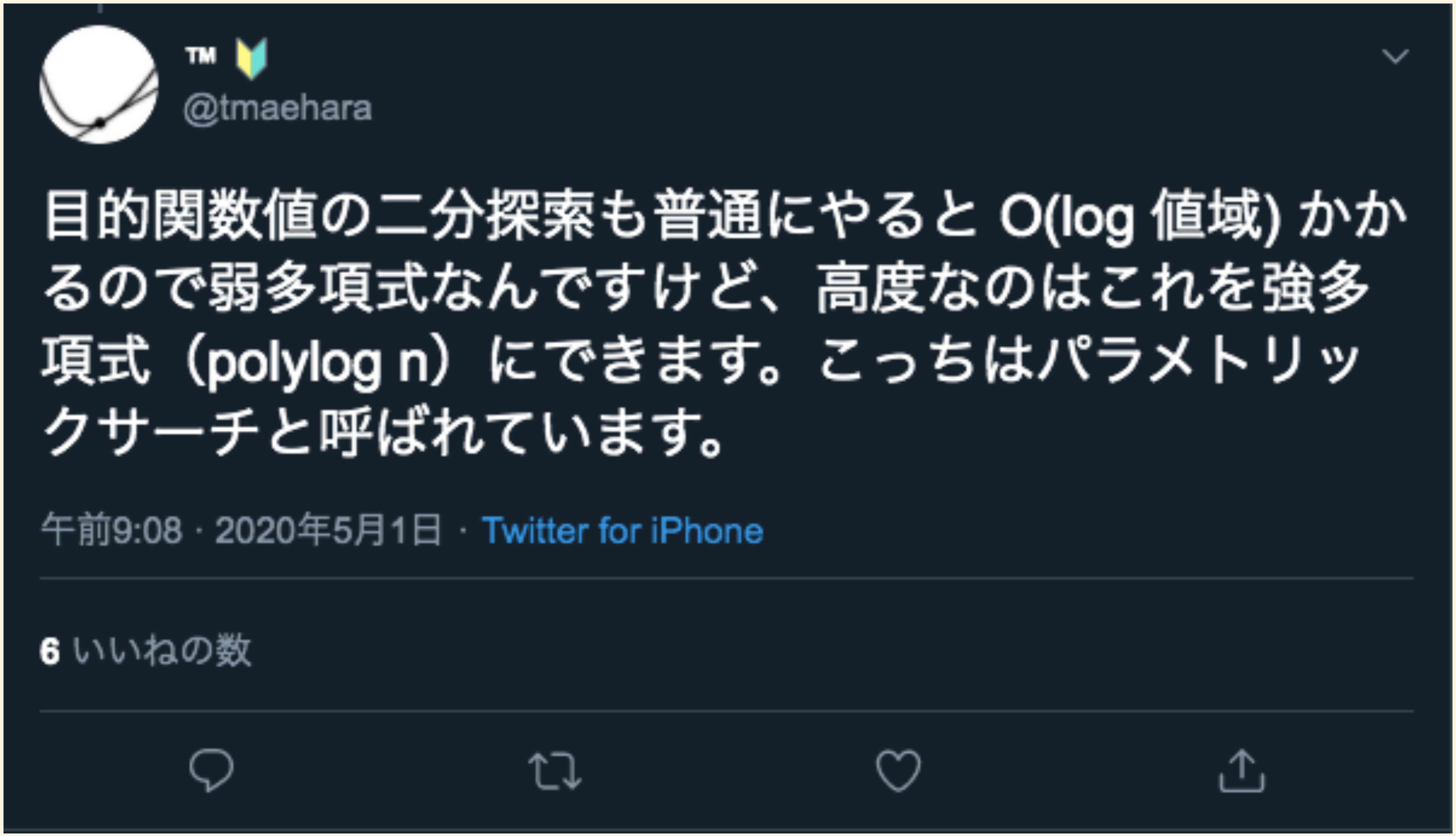
# まとめ

- 概要
  - O(logn)で高速なアルゴリズム
  - 応用が効く
- 二分探索を利用して解く手順
  - 二分探索出来る問題の性質を持っているか確認
    - 適当に解の値を決めた場合,<br>それを満たす解が存在するか素早く判定できる
    - 単調性（ある地点より右はずっとYes）がある
  - 判定問題 (Yes/Noで答える問題) にする
  - 二分探索でYesとなる最小(最大)の解の値を求める

# 自由課題

- C - Garden：https://atcoder.jp/contests/cf17-relay-open/tasks/relay2_c
  - ちょっと難しいかもしれません.
    解説も参照してみてください.
  - 時間があれば一緒に解いていきます

# Appendix.
# Parametric Search



TM
@tmaehara

目的関数値の二分探索も普通にやると O(log 値域) かかるので弱多項式なんですけど、高度なのはこれを強多項式（polylog n）にできます。こっちはパラメトリックサーチと呼ばれています。

午前9:08 · 2020年5月1日 · Twitter for iPhone

6 いいねの数

# 弱多項式時間って？

- 強多項式時間
  - O()の中が配列の長さNなど入力の「個数」の多項式
- 擬多項式時間
  - O()の中が整数Aなど入力の「数値」の多項式
- 弱多項式時間
  - O()の中が整数Aなど入力の「数値」のlogの多項式

**イメージ的には，強多項式＞弱多項式＞擬多項式 の順で速い**

参考：計算量オーダーの求め方を総整理！ 〜 どこから log が出て来るか 〜 https://qiita.com/drken/items/872ebc3a2b5caaa4a0d0

11. 動的計画法と擬多項式時間アルゴリズム http://www.akita-pu.ac.jp/system/elect/ins/kusakari/japanese/teaching/InfoMath/2007/note/11.pdf

# Parametric Search

- Megiddoによって開発
- 実用上は二分探索で十分でパラメトリック探索は理論上のものという話があったり
- 主に計算幾何学で使われる模様
- 並列アルゴリズムを逐次的に実行

  並列処理部分を単調性を利用して一括処理して効率化する（のだと思う...）

# Parametric Search

- 例題がここに載っていたので,
  興味ある方はみて下さい
  - 計算幾何学と並列アルゴリズム
    http://www.orsj.or.jp/~archive/pdf/bul/Vol.37_08_386.pdf
  - まとめようと思いましたが気合と時間が足りませんでした．例題部分は辛うじて理解しやすかったです．

# Parametric Search

- 例題
  - $f_i(x) = a_i x + b_i \quad (0 \le i \le n)$ とし,
  - $f(x) = \mathrm{median}\{f_i(x), f_2(x), \ldots, f_n(x)\}$ と定義する.
  - $f(x) = 0$ となる $x$ を求めよ.

**@TODO**
**一般的な枠組みから**
**説明しないと誤解を生む**

- 考察とアイデア
  - $f(x)$ は単調増加関数である
  - $f(x) = 0$ となる $x$ を $x^*$ と置く.
  - $x^*$を知らない状態で $f_i(x^*), f_2(x^*), \ldots, f_n(x^*)$ を並び替える
    - この順番が分かれば $f_i(x^*) = f(x^*)$ が分かる → 解ける

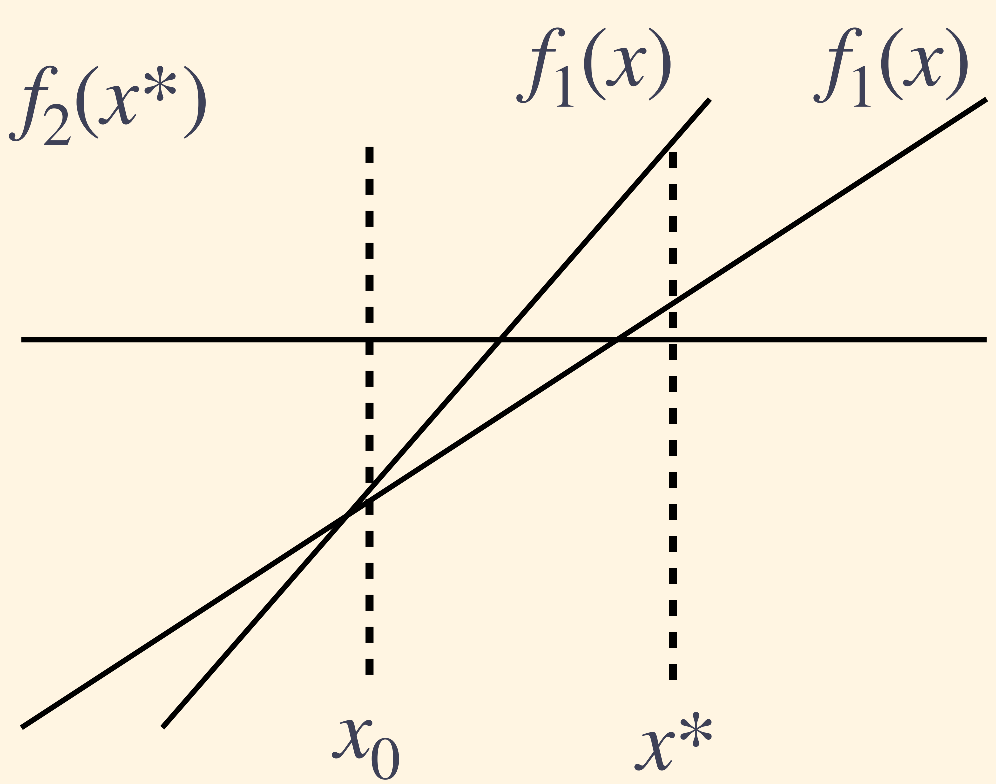
# Parametric Search

- 解法
  - $x^*$ を知らない状態で $f_i(x^*), f_2(x^*), \ldots, f_n(x^*)$ を並び替える
  - $f_1(x^*)$ と $f_2(x^*)$ を比較したい $(a_1 > a_2)$
  - $f_1(x^*)$ と $f_2(x^*)$ の交点を $x^*$ とすると,

$$x^* > x_0 \quad \textbf{の時} \quad f_1(x^*) > f_2(x^*)$$

$$x^* < x_0 \quad \textbf{の時} \quad f_1(x^*) < f_2(x^*)$$

  - $x^*, x_0$ の大小関係が分かれば良い

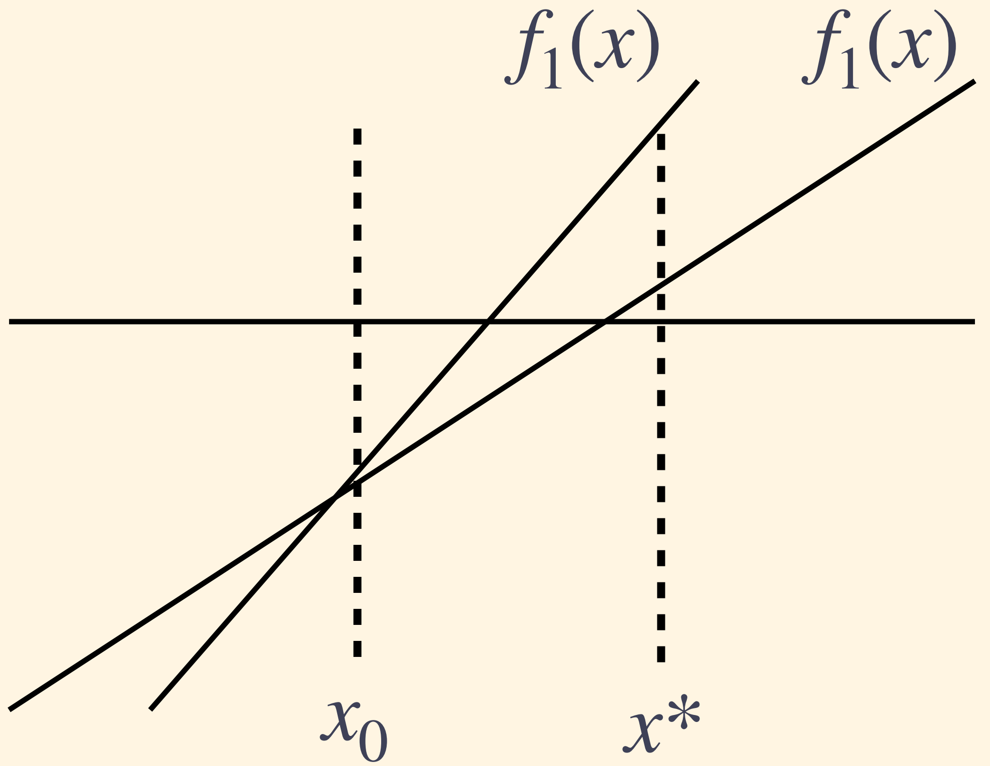


# Parametric Search

- 解法
  - $x^*, x_0$ の大小関係が分かれば良い
  - これは下記のように分かる

$$f(x_0) > 0 \quad \textbf{であれば} \quad x_0 > x^*$$

$$f(x_0) < 0 \quad \textbf{であれば} \quad x_0 < x^*$$




  - 比較にはf(x)を求める時間
    O(n)掛かる