プログラミング及び実習2

第 6 回 二重ループ、配列の応用（統計量・ソート・数え上げ）

佐野 彰

本日の内容

1. 二重の繰り返しループ（復習）
2. 配列変数の応用（統計量・並べ替え・数え上げ）
3. プログラミング試験（第08回）の準備

1. 二重の繰り返しループ

二重の繰り返し処理

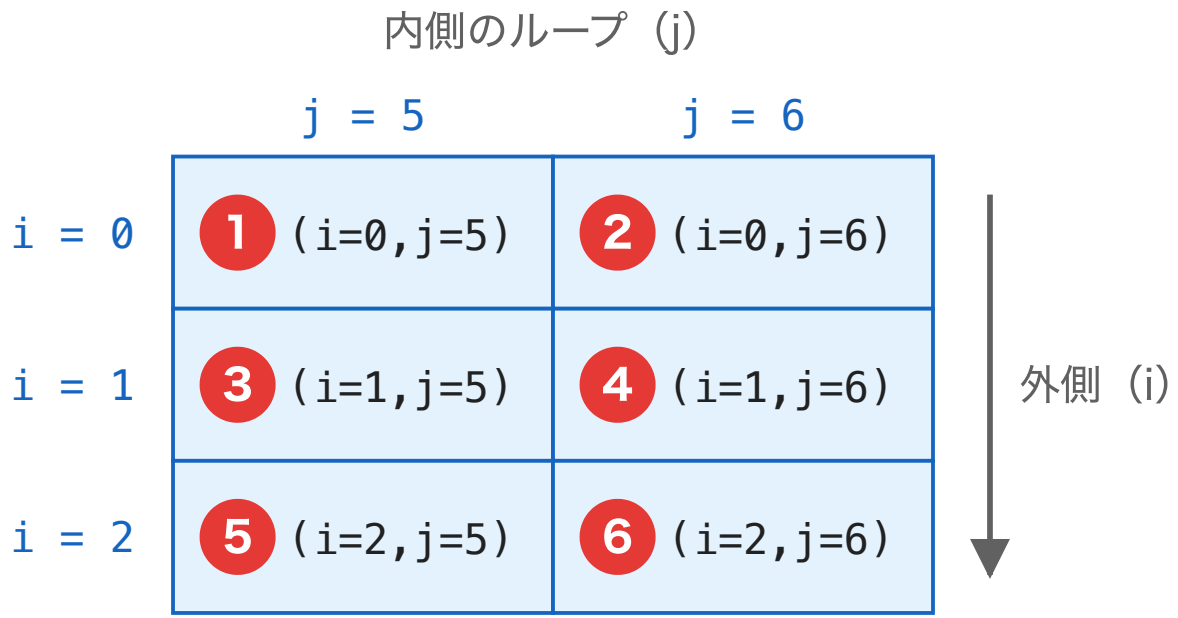
```c
int i, j;
for (i = 0; i < 3; i++) {
    for (j = 5; j < 7; j++) {
        printf("(i=%d,j=%d)\n", i, j);
    }
}
```

実行例

```
(i=0,j=5)
(i=0,j=6)
(i=1,j=5)
(i=1,j=6)
(i=2,j=5)
(i=2,j=6)
```

`i` に関する `for` ループの中に、さらに `j` に関する `for` ループがあります。
`(j = 5,6) × (i = 0,1,2)` で、計 6 回の `printf()` が実行されます。

二重ループの実行順



外側の `i` を 1 つ進める前に、内側の `j` が 5 から 6 まで一回り動きます。
表の左上から、1 行ずつ右へ進む順に `printf()` が実行されます。

復習: 掛け算九九表（ex05 の p6）

```c
for (i = 1; i <= 9; i++) {        // 行
    for (j = 1; j <= 9; j++) {    // 列
        printf("%3d", i * j);
    }
    printf("\n");
}
```

実行例

```
  1  2  3  4  5  6  7  8  9
  2  4  6  8 10 12 14 16 18
  3  6  9 12 15 18 21 24 27
  4  8 12 16 20 24 28 32 36
  5 10 15 20 25 30 35 40 45
  6 12 18 24 30 36 42 48 54
  7 14 21 28 35 42 49 56 63
  8 16 24 32 40 48 56 64 72
  9 18 27 36 45 54 63 72 81
```

外側のループ（`i`）が「行」、内側のループ（`j`）が「行の中の列」に対応します。

`printf("%3d", ...)` の `3` は、第 4 回で学んだ変換指定の表示幅です。

内側のループの回数を外側の変数で変える

```c
for (i = 1; i <= 9; i++) {
    for (j = 1; j <= i; j++) {   // 上限が i
        printf("%3d", i * j);
    }
    printf("\n");
}
```

内側のループの条件を `j <= i` にすると、`i` 行目には `i` 個だけ表示されます。九九表の左下の三角形の部分になります。

実行例

```
  1
  2  4
  3  6  9
  4  8 12 16
  5 10 15 20 25
  6 12 18 24 30 36
  7 14 21 28 35 42 49
  8 16 24 32 40 48 56 64
  9 18 27 36 45 54 63 72 81
```

内側のループの範囲が外側の変数で決まる形は、このあとの並べ替え（選択ソート）でも使います。

2. 配列変数の応用

データ統計量: 最大値

```c
#define N 10

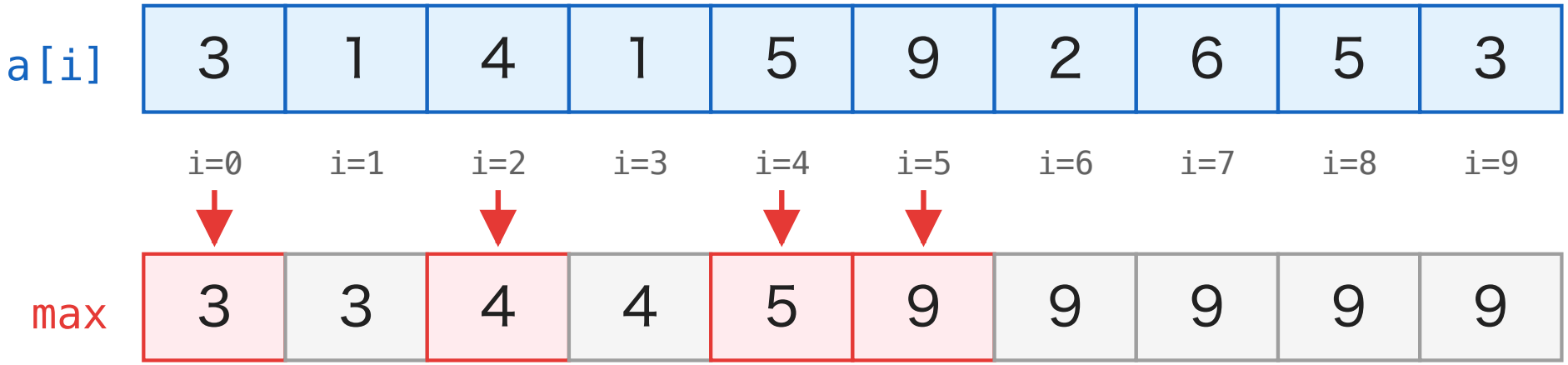
int a[N];
int max, i;
for (i = 0; i < N; i++) {
    scanf("%d", &a[i]);
    if (i == 0 || max < a[i]) {
        max = a[i];
    }
}
printf("最大値 %d\n", max);
```

実行例

```
$ ./max
3 1 4 1 5 9 2 6 5 3
最大値 9
```

入力した値が最初の値であるか、それまでの最大値より大きければ、`max` を更新します（最小値は、不等号を逆にすれば同じように求まります）。

最大値の更新の様子



`a[i]` を左から順に見て、それまでの `max` より大きい値（赤）が来たときだけ `max` を更新します。
最後の `max` の 9 が全体の最大値です。

データ統計量: 分散と標準偏差

```c
#include <math.h>
#define N 5

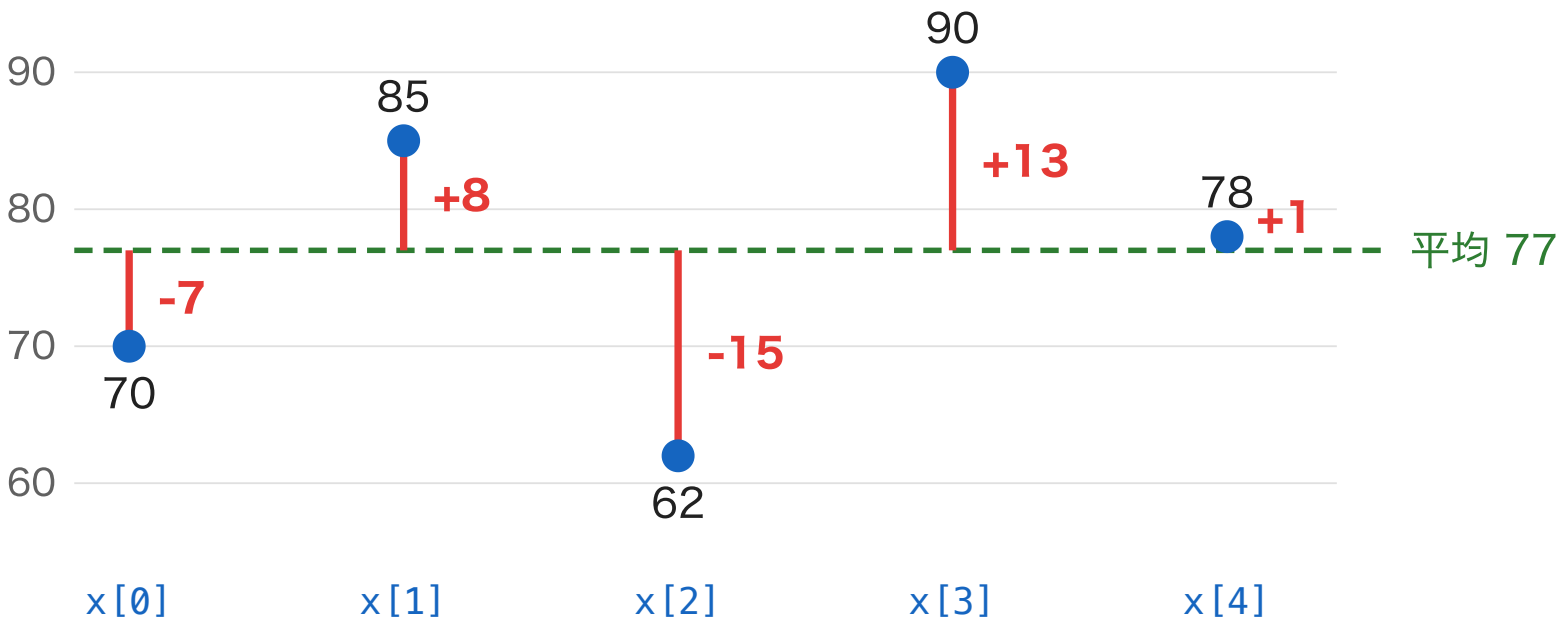
double x[N], sum = 0.0, avg, var = 0.0;
int i;
for (i = 0; i < N; i++) {
    scanf("%lf", &x[i]);
    sum += x[i];
}
avg = sum / N;
for (i = 0; i < N; i++) {
    var += (x[i] - avg) * (x[i] - avg);
}
var /= N;
printf("平均 %.1f, 分散 %.2f, 標準偏差 %.2f\n",
       avg, var, sqrt(var));
```

実行例

```
$ ./var
70 85 62 90 78
平均 77.0, 分散 101.60, 標準偏差 10.08
```

- 平均は第 5 回の例 1 と同じ求め方です
- 分散は「平均との差」（第 5 回の例 2）の 2 乗の平均です。標準偏差は分散の平方根です
- 平均は全員分を読み込むまで分からないので、入力を配列に覚えておきます

平均との差を図で見る



赤い線の長さが平均との差です。分散は、この差を 2 乗して平均したものです。

$((-7)^2 + 8^2 + (-15)^2 + 13^2 + 1^2) / 5 = 508 / 5 = 101.6$

配列変数の並べ替え（選択ソート）

まず、`a[0]` の値が全体の最大値になるように入れ替えます:

```
#define N 10

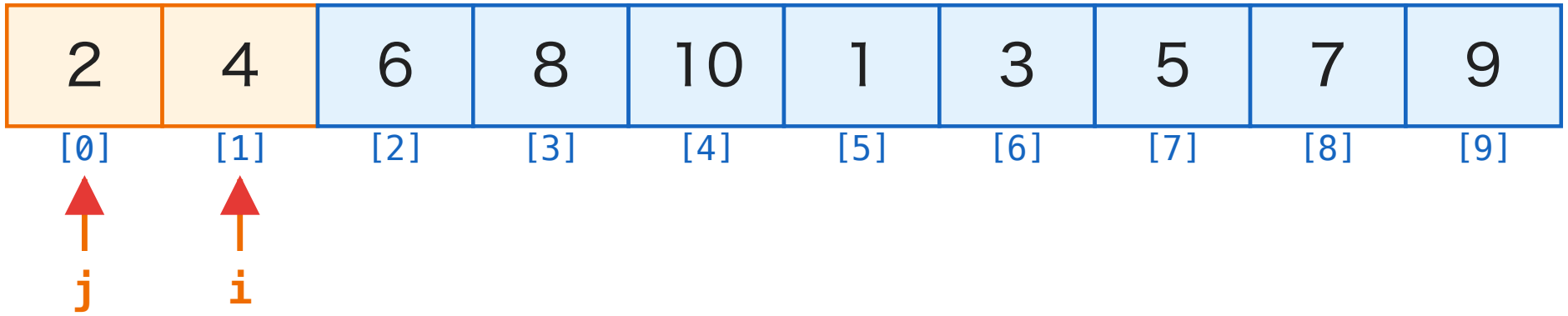
int a[N] = {2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
int i, tmp;
for (i = 1; i < N; i++) {
    if (a[0] < a[i]) {
        tmp = a[0];   // tmp を使って
        a[0] = a[i];  // a[0] と a[i] を
        a[i] = tmp;   // 入れ替える
    }
}
for (i = 0; i < N; i++) {
    printf("%d ", a[i]);
}
printf("\n");
```

実行例

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

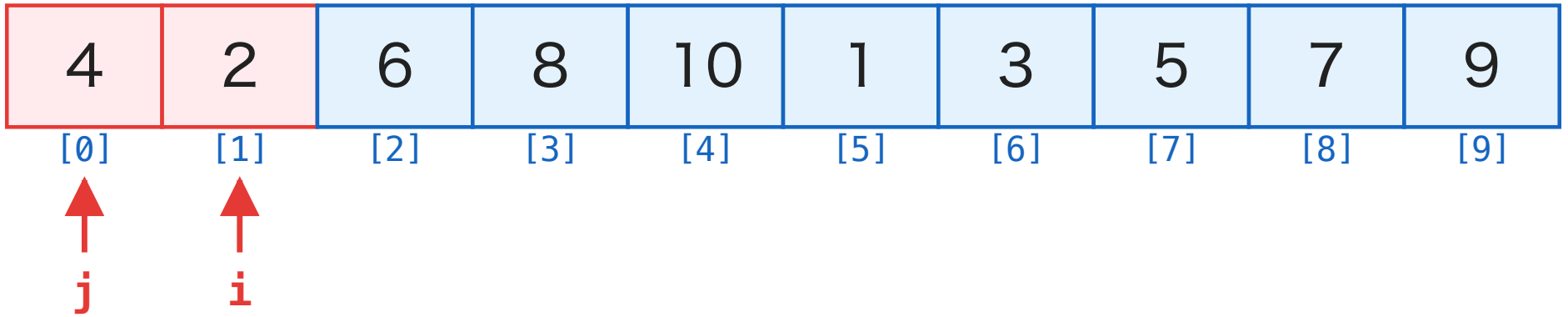
最大値の 10 が `a[0]` に来ます。ほかの要素の並びは、入れ替えのたびに少しずつ動きます。

選択ソートの動き（j = 0 の段 1/6）



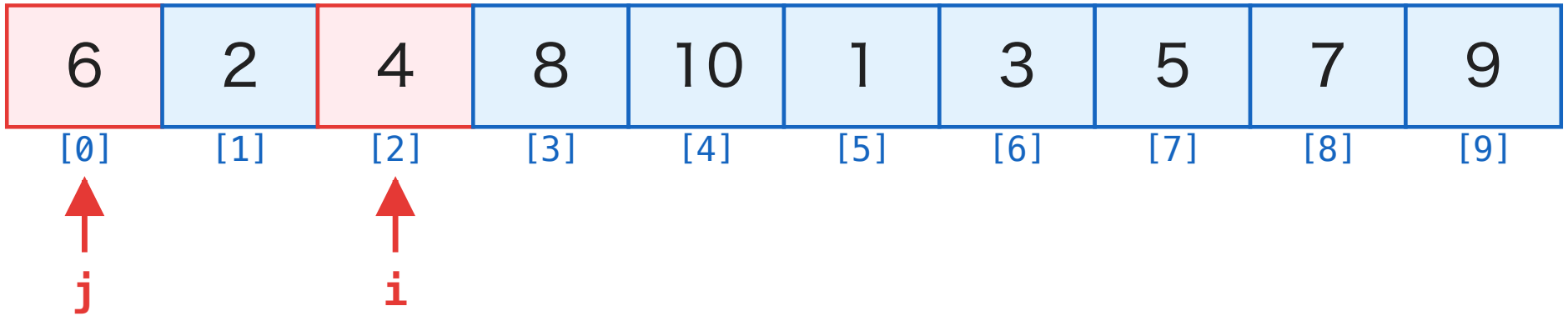
`j = 0` の段では、`a[0]` と `a[1]` 〜 `a[9]` を順に比べます。まず `a[0]`（2）と `a[1]`（4）を比べます。

選択ソートの動き（j = 0 の段 2/6）



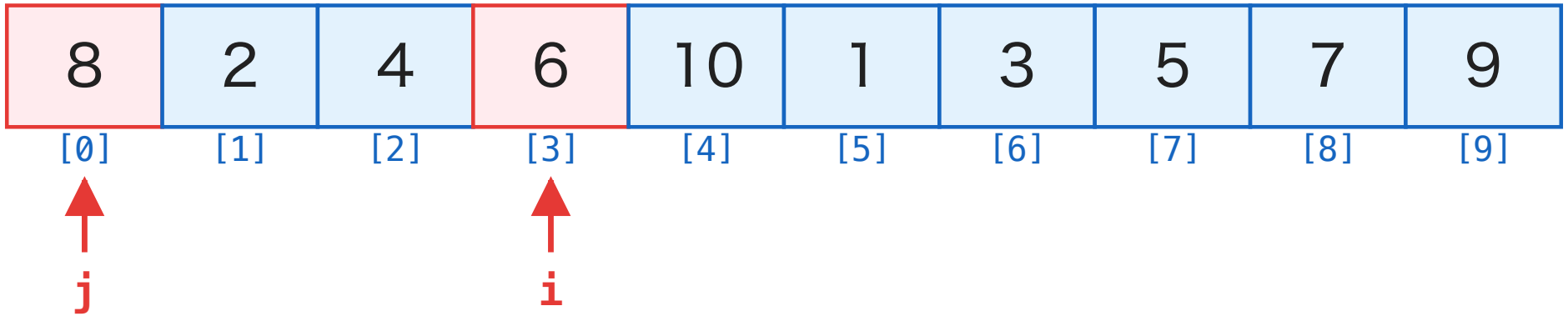
`i = 1`: `a[0]` の 2 より `a[1]` の 4 が大きいので、入れ替えます。`a[0]` は 4 になります。

選択ソートの動き（j = 0 の段 3/6）



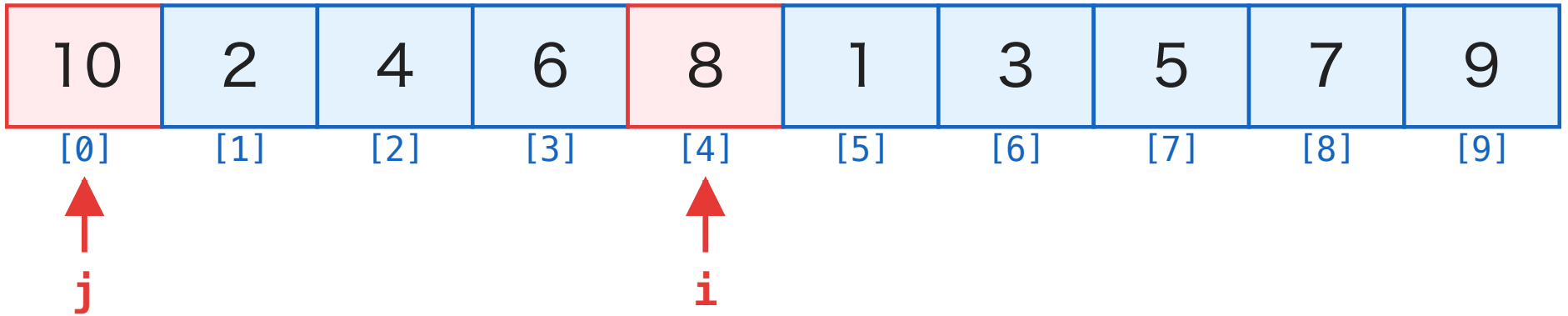
`i = 2`: 4 より 6 が大きいので入れ替えます。`a[0]` は 6 になります。

選択ソートの動き（j = 0 の段 4/6）



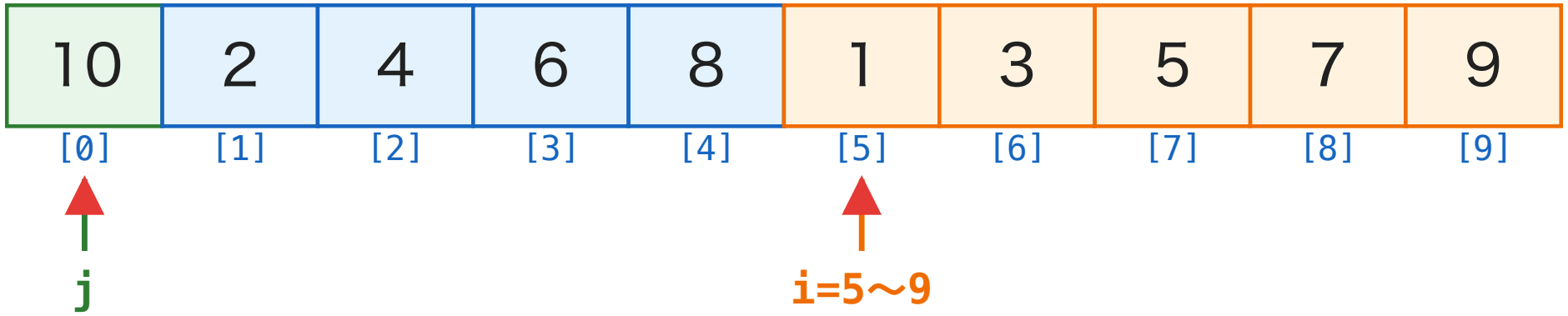
`i = 3`: 6 より 8 が大きいので入れ替えます。`a[0]` は 8 になります。

選択ソートの動き（j = 0 の段 5/6）



`i = 4`: 8 より 10 が大きいので入れ替えます。`a[0]` は 10 になります。

選択ソートの動き（j = 0 の段 6/6）



`i = 5` 〜 `9`：10 より大きい値はないので、入れ替えません。
この段が終わると、`a[0]` には全体の最大値 10 が入ります（緑）。

選択ソートの完成

`j` を 1 つずつずらしながら、`a[j]` の値が `a[j]` 以降の最大値になるように入れ替えます。

```
int a[N] = {2, 4, 6, 8, 10, 1, 3, 5, 7, 9};
int i, j, tmp;
for (j = 0; j < N - 1; j++) {
    for (i = j + 1; i < N; i++) {  // j で決まる
        if (a[j] < a[i]) {
            tmp = a[j];
            a[j] = a[i];
            a[i] = tmp;
        }
    }
}
for (i = 0; i < N; i++) {
    printf("%d ", a[i]);
}
printf("\n");
```

実行例

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

`a[]` が降順（大きい順）に並びます。

最大値の位置を覚えておき、内側のループが終わってから 1 回だけ入れ替える形が、本来の選択ソートです。

段ごとの並びの変化

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
初め	2	4	6	8	10	1	3	5	7	9
j = 0 の後	10	2	4	6	8	1	3	5	7	9
j = 1 の後	10	9	2	4	6	1	3	5	7	8
j = 2 の後	10	9	8	2	4	1	3	5	6	7
j = 3 の後	10	9	8	7	2	1	3	4	5	6
j = 4 の後	10	9	8	7	6	1	2	3	4	5
j = 5 の後	10	9	8	7	6	5	1	2	3	4
j = 6 の後	10	9	8	7	6	5	4	1	2	3
j = 7 の後	10	9	8	7	6	5	4	3	1	2
j = 8 の後	10	9	8	7	6	5	4	3	2	1

`j` の段が終わるたびに、残りの最大値が `a[j]` に入り、先頭から確定します（緑）。

データの数え上げ（ヒストグラム）

第 5 回の例 3（`days[m - 1]`）とは逆に、今回は**値を添字に使って**数えます。
`a[]` の値が 0〜9 の整数なら、数え上げ用の配列 `b[10]` を用意して、次のように数えられます:

```c
int a[N] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};
int b[10] = {0};   // すべて 0 で始める
for (i = 0; i < N; i++) {
    if (a[i] >= 0 && a[i] <= 9) {  // 範囲を確認
        b[a[i]]++;                 // 1 を加える
    }
}
for (i = 0; i < 10; i++) {
    printf("%d ", b[i]);
}
printf("\n");
```

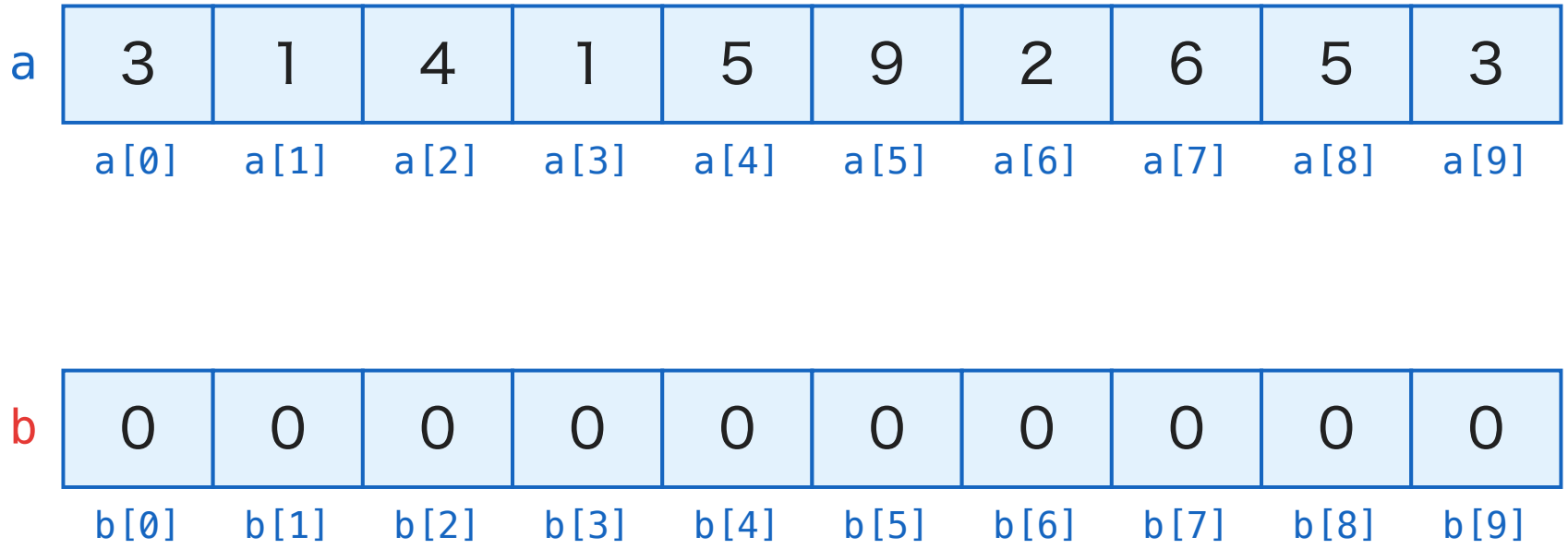
実行例

```
0 2 1 2 1 2 1 0 0 1
```

`b[0]` から順に、0〜9 のそれぞれが
何個あったかです（1 と 3 と 5 が 2 個ずつ）。

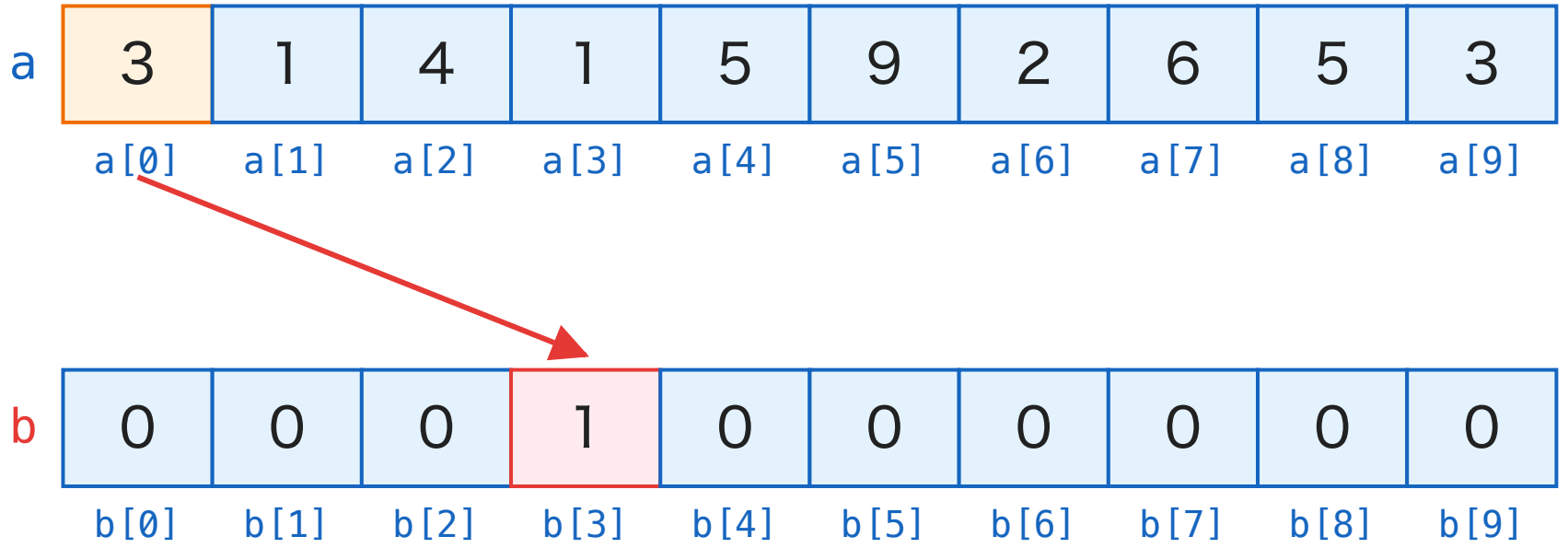
範囲外の値をそのまま添字にすると、
存在しない要素を使ってしまいます（第 5 回の
「範囲外の添字」）。

数え上げの動き（1/6）



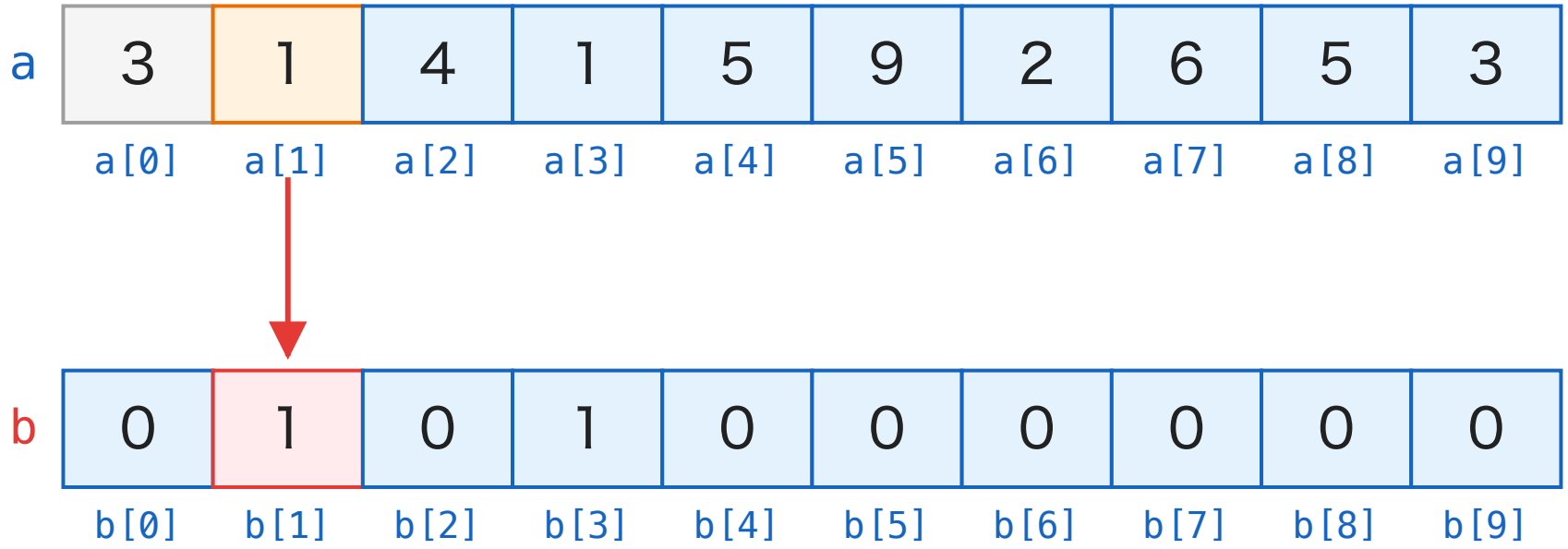
`b[]` をすべて 0 にしてから、`a[0]` から順に見ていきます。

数え上げの動き（2/6）



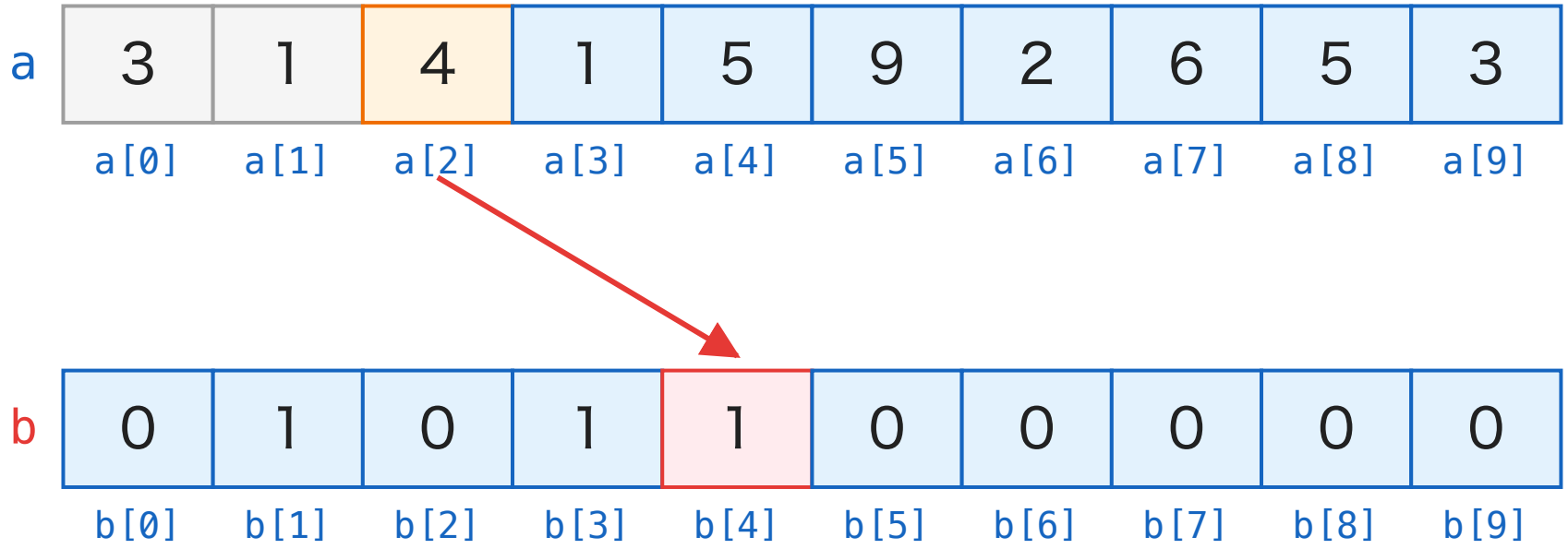
`i = 0`：`a[0]` は 3 なので、`b[3]` に 1 を加えます。

数え上げの動き（3/6）



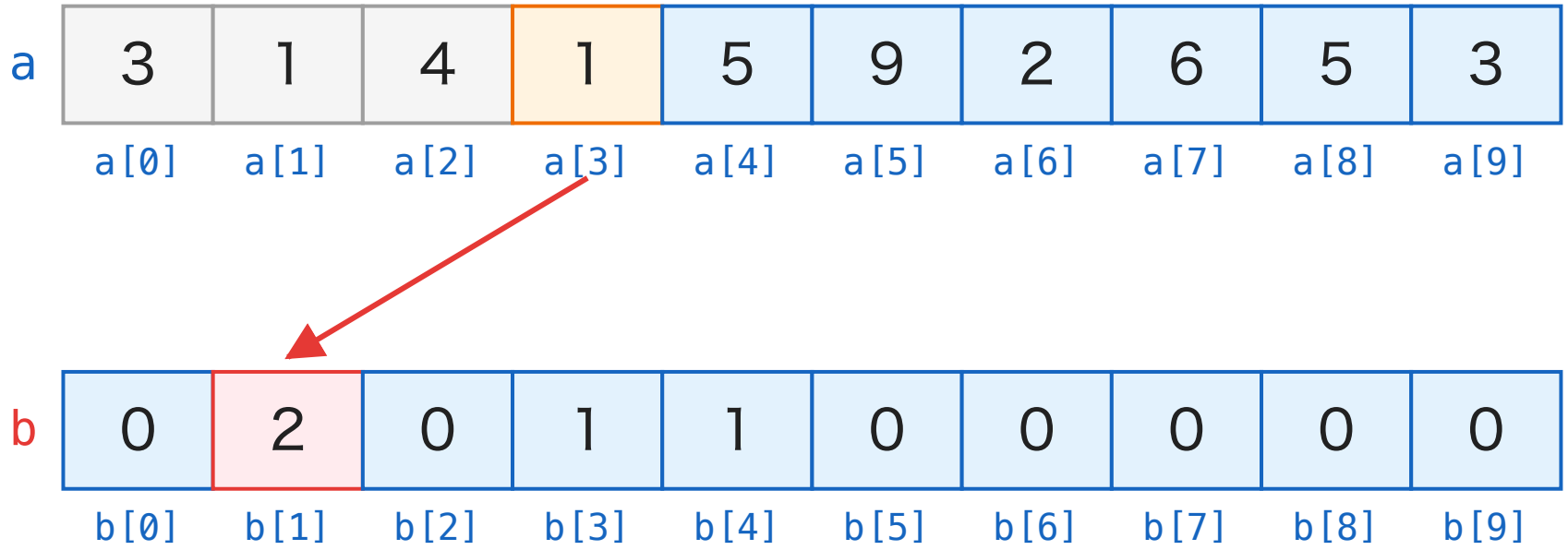
`i = 1`：`a[1]` は 1 なので、`b[1]` に 1 を加えます。

数え上げの動き（4/6）



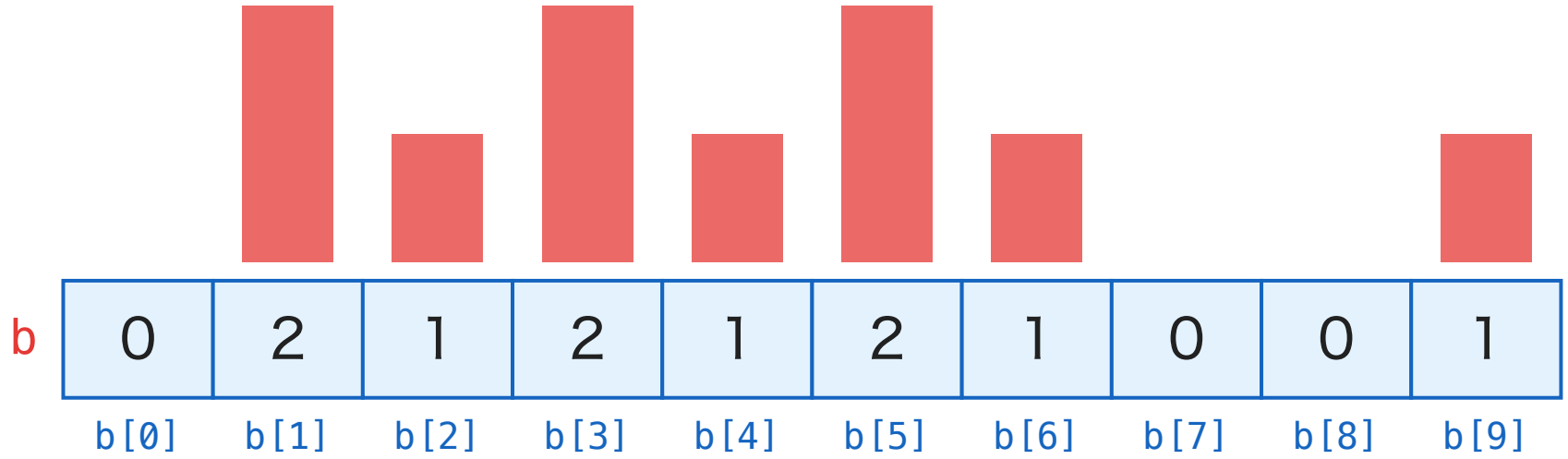
`i = 2`：`a[2]` は 4 なので、`b[4]` に 1 を加えます。

数え上げの動き（5/6）



`i = 3`：`a[3]` は再び 1 なので、`b[1]` が 2 になります。同じ値が出るたびに、同じ要素が増えていきます。

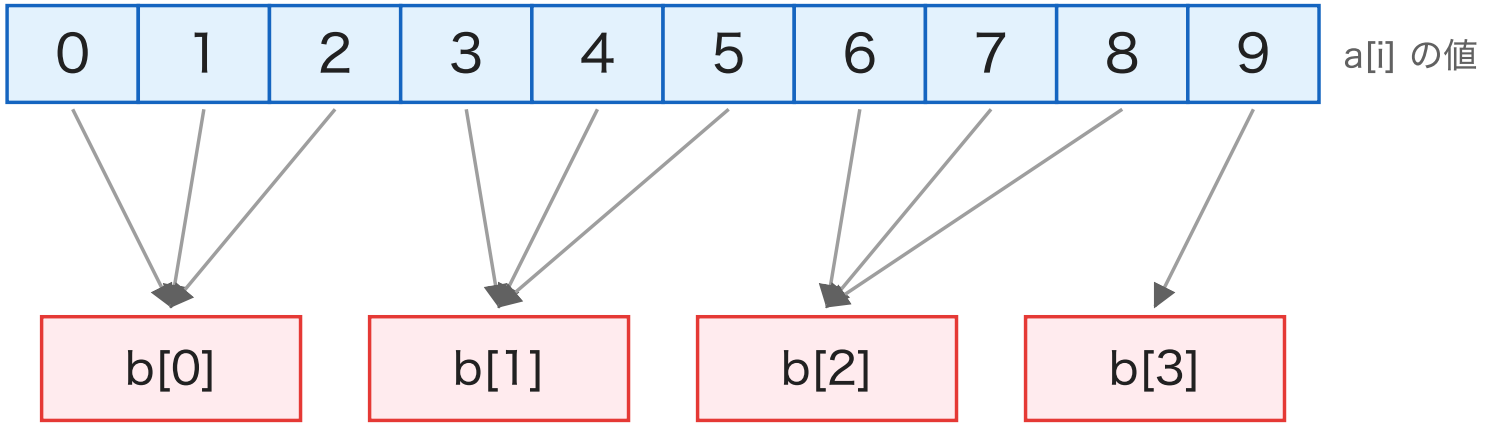
数え上げの動き（6/6）



`i = 9` まで続けると、`b[k]` は値 `k` の個数になります。棒の高さで表したものがヒストグラムです。

数え上げの工夫

範囲を区切って数えたい場合は、`b[a[i] / 3]++;` のように添字の計算を工夫します（例: `{0,1,2}, {3,4,5}, {6,7,8}, {9}` の 4 区分なら、配列は `b[4]`）。



おまけ: 数え上げの結果を使うと、簡単にソートできます（分布数え上げソート）。

3. プログラミング試験の準備

試験の実施方法（予告）

- **プログラミング試験 @第08回**（11/12・1-609）
- 試験時間は最大 2 時間、複数問題からの選択制（50 点満点）
- 問題の提示・解答・提出は aiJudge の Web エディタで行う（今回・次回に練習課題あり）
- 試験中は自動採点されない。終了 20 分前に採点が始まり、それまでの提出もまとめて採点される
- 解答は Web エディタ上で書く。打鍵・貼り付けなどの作業は記録される
- 自分のソースファイル・ネット検索・書籍資料の参照は可
- 他者や生成 AI とのやり取り（会話・チャット・SNS・ファイル送付・ChatGPT など）は**禁止**

試験の実施方法（予告・つづき）

- 自分の演習課題などの既存のコードを Web エディタに貼り付け、修正して解答に使うのはよい
- 使えるモニタは 1 台だけ。実習のように、実習室のモニタをノート PC の外部ディスプレイにはできない（仮想デスクトップでの切り替えはよい）
- 実習室 PC のブラウザからも受験できる。その場合は、自分の演習のソースコードなどを実習室 PC で使えるように準備しておく
- コンパイルできないソースファイルは採点されない
- 試験中は TA のサポートを一切受けられない
- コピーコードは機械的にチェックし、疑わしい場合は作業の記録を確認、必要なら口頭試問を行う。コピーと判定された場合は、学則に従い、不正行為として扱う

次回予告と今日やること

次回（11/05）：前半のまとめ（模擬試験的演習）

今日の 5 講時（実習）

1. 第 5 回課題の解答例を演習ページで確認する（FizzBuzz、電気料金の段階計算など）
2. 二重ループと配列の応用（統計量・選択ソート・ヒストグラム）を確認する
3. 演習課題 ex06 に取り組み、100 点以上を取る
4. ex06 の問題を Web エディタでも解き、提出まで操作を確かめる
    - 次回の ex07 と試験は、Web エディタからしか提出できない

質問は **Teams「★ 質問用チャネル」** か、教室の TA へ。