C++ プログラミング III

第11回

クイックソート

永並健吾

ソートアルゴリズム

ソート (Sort) とは, データ・数値を規定された大小関係に従って並べること.

- 昇順 (ascending order)　データを小さい順に並べること.
- 降順 (descending order)　データを大きい順に並べること, 昇順の逆.

ソートのアルゴリズムは数多く考案されてきたが, それぞれによって 計算効率, メモリ使用量, 並列化の容易さ, 適用制限 などが異なる.

	平均計算量	安定性 (※)	扱う講義
選択ソート	$O(n^2)$	安定でない	第10回 (今回)
バブルソート	$O(n^2)$	安定	第10回 (今回)
クイックソート	$O(n \log n)$	安定でない	第10回 (今回)
マージソート	$O(n \log n)$	安定	第11回
ヒープソート	$O(n \log n)$	安定でない	第12回

※ ソートの前後で, 値の等しい要素の順序が維持されているとき, 安定 であるという.

ソートアルゴリズムの効率

アルゴリズムの効率は, 以下の2つの指標で評価される.

- 時間計算量 (time complexity)
  解を出すまでのステップ数の総和. 入力サイズに対するオーダーで表現.
- 空間計算量 (space complexity)
  実行に必要なメモリの量. こちらも入力サイズに対するオーダーで表現.

※ オーダー表現については [ランダウの記号 – Wikipedia] など参照

時間計算量の理論的評価には, 以下の2種類の方法がある.

- 平均時間計算量 (average time complexity)
  サイズが等しい入力を与えた中での時間計算量の期待値.
- 最悪計算量 (worst case time complexity)
  サイズが等しい入力を与えた中での最大の時間計算量.

ソートアルゴリズムの比較

	選択ソート	クイックソート
平均時間計算量	$O(n^2)$	$O(n \log n)$

n は入力サイズ (n個の要素を並び替える)

選択ソートとクイックソートの平均時間計算量を比較すると,
クイックソートのほうが効率の良いと言える.

$n = 1{,}000{,}000$ (100万) に対して,
$n^2 = 10^{12}$ (1兆) であるが, $n \log n \fallingdotseq$ 13,815,510 (約1400万) である.

クイックソートだと1秒で計算可能な処理も, 選択ソートだと丸1日以上かかる
なんてことも起こりうる.

※ 実は, クイックソートの最悪計算時間量は $O(n^2)$ である (選択ソートの最悪時間計算量も $O(n^2)$).
そのため, 初期状態が「悪い」並び方だと, クイックソートでも効率よく計算できないのだが, そのような
場合は確率的にかなり少ない.

選択ソート

- 配列の各要素の中で最小値が格納されている要素を探す.
- その要素を先頭の要素と 入れ替える. (最小値が先頭の要素ならば, なにもしない)
  ここで先頭の要素は確定.
- 次に, 残りの要素の中で最小値が格納されている要素を探す.
- その要素を 2番目の要素と 入れ替える. (最小値が2番目の要素ならば, なにもしない)
  ここで2番目の要素は確定.

⋮ 最後の要素 (最後から2番目の要素) が確定するまで繰り返す.

「最小値を探す (線形探索)」と「要素を交換する」 を $(n-1)$回繰り返すアルゴリズム

選択ソートの例

未確定の要素

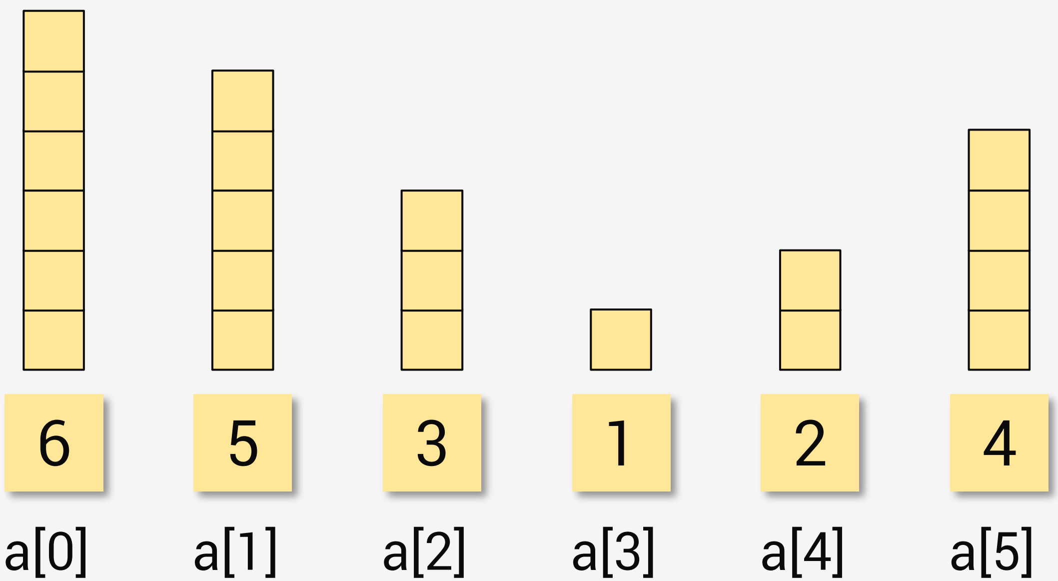
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

初期状態

ループ制御変数 i (初期値 0)を用意.



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

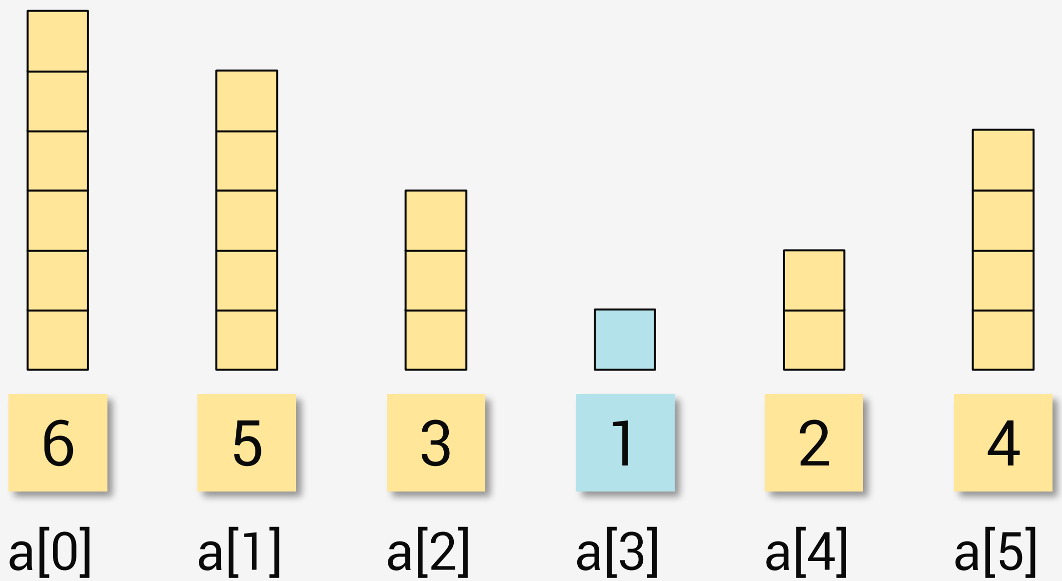
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 0-1: int i=0

a[i] 〜 a[n-1] の中で最小の値を探す



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

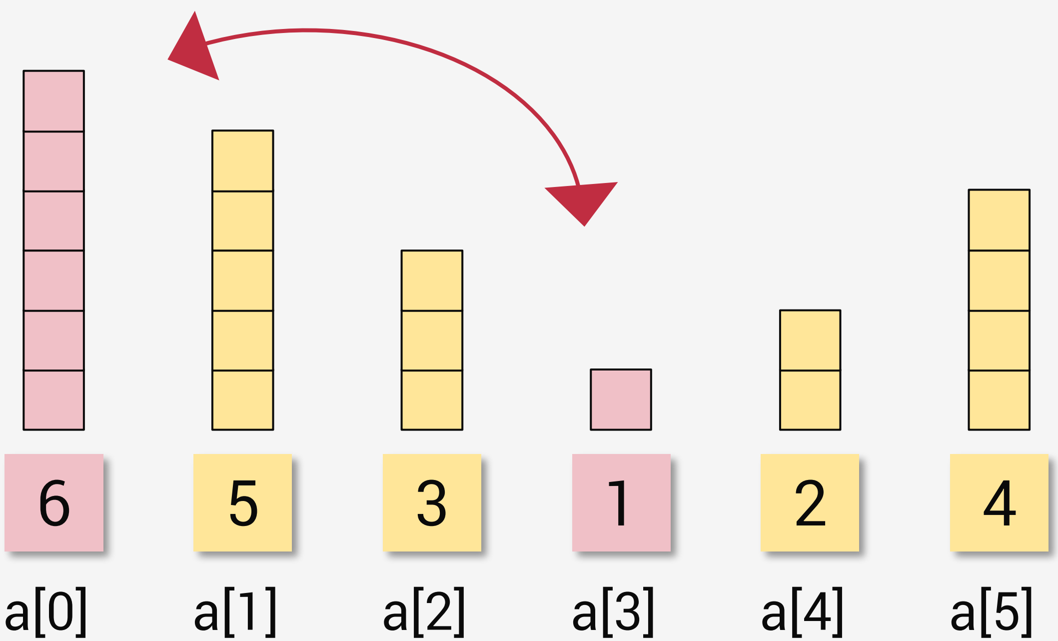
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 0-2: int i=0

最小の値を a[i] と入れ替える.



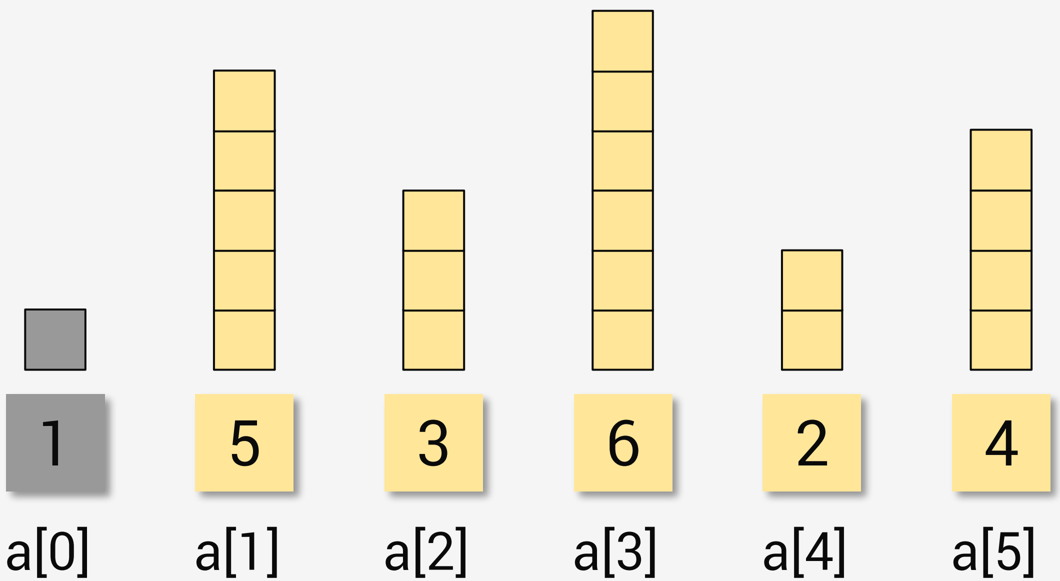
入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素
確定済の要素
交換する要素
(未確定要素のうち)
最小の要素

Step 0-3: int i=0→1

a[i]が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

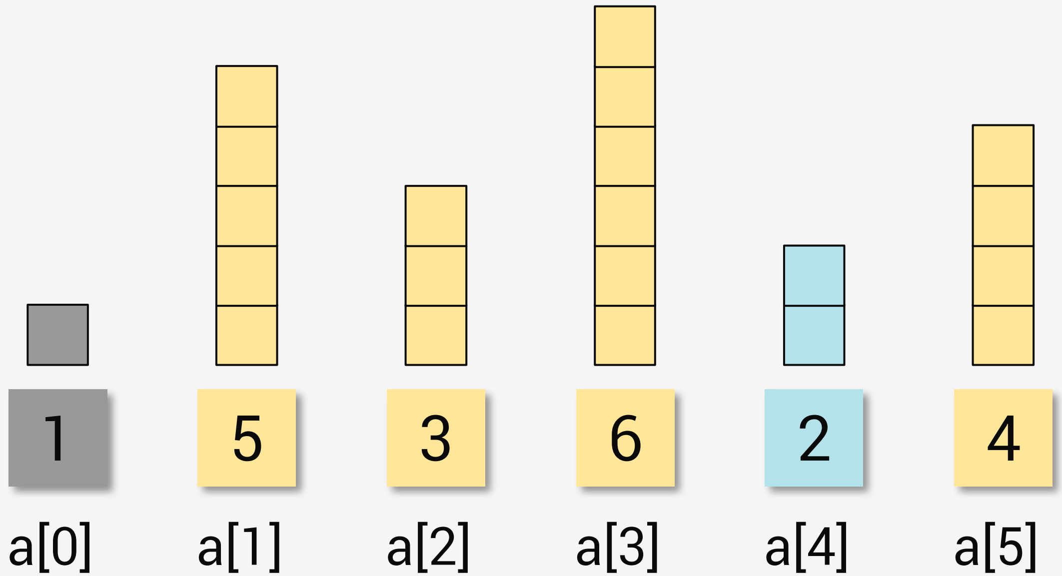
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 1-1: int i=1

a[i] ～ a[n-1] の中で最小の値を探す

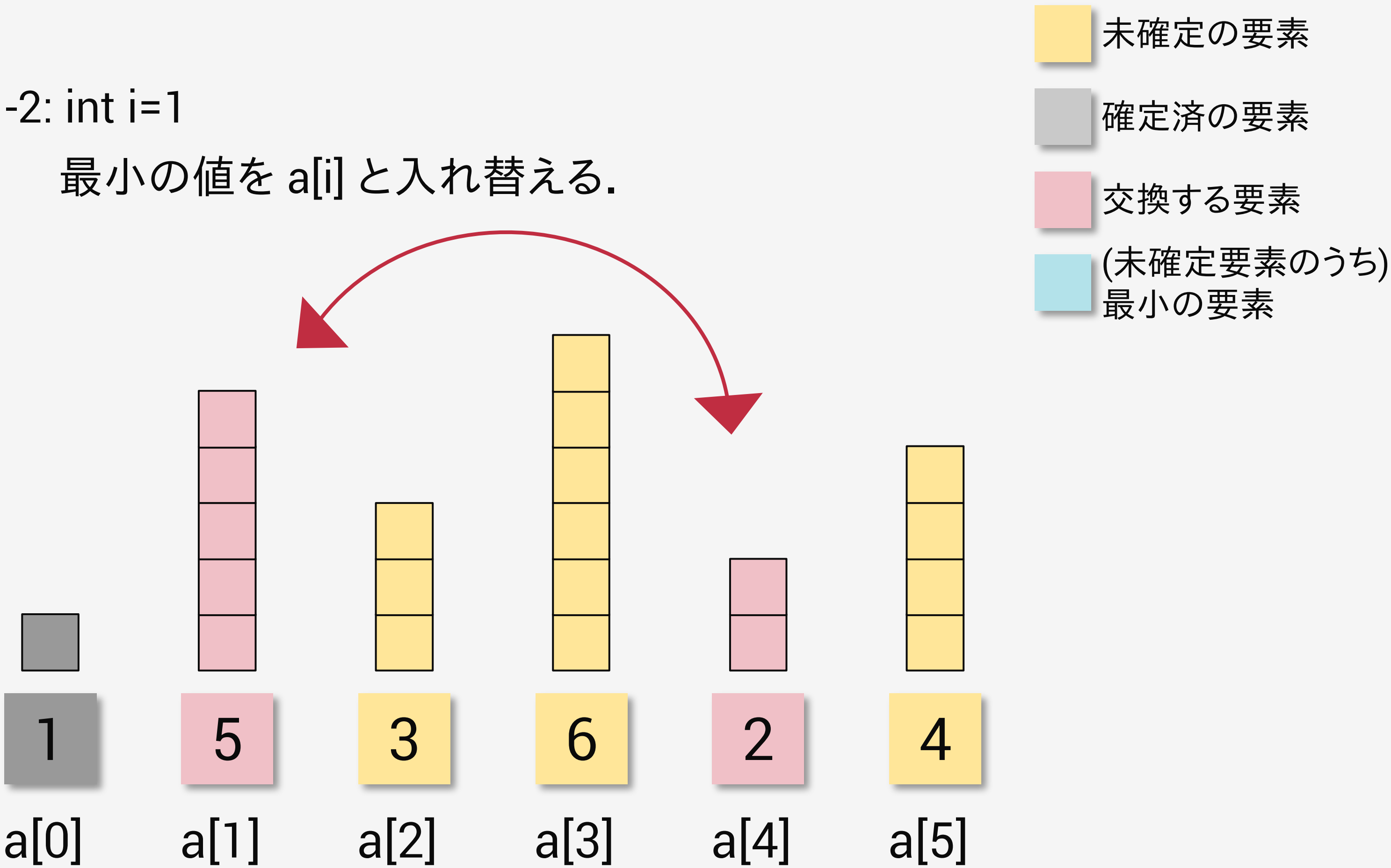


入力サイズ $n = 6$ の場合の例

選択ソートの例

Step 1-2: int i=1

最小の値を a[i] と入れ替える.

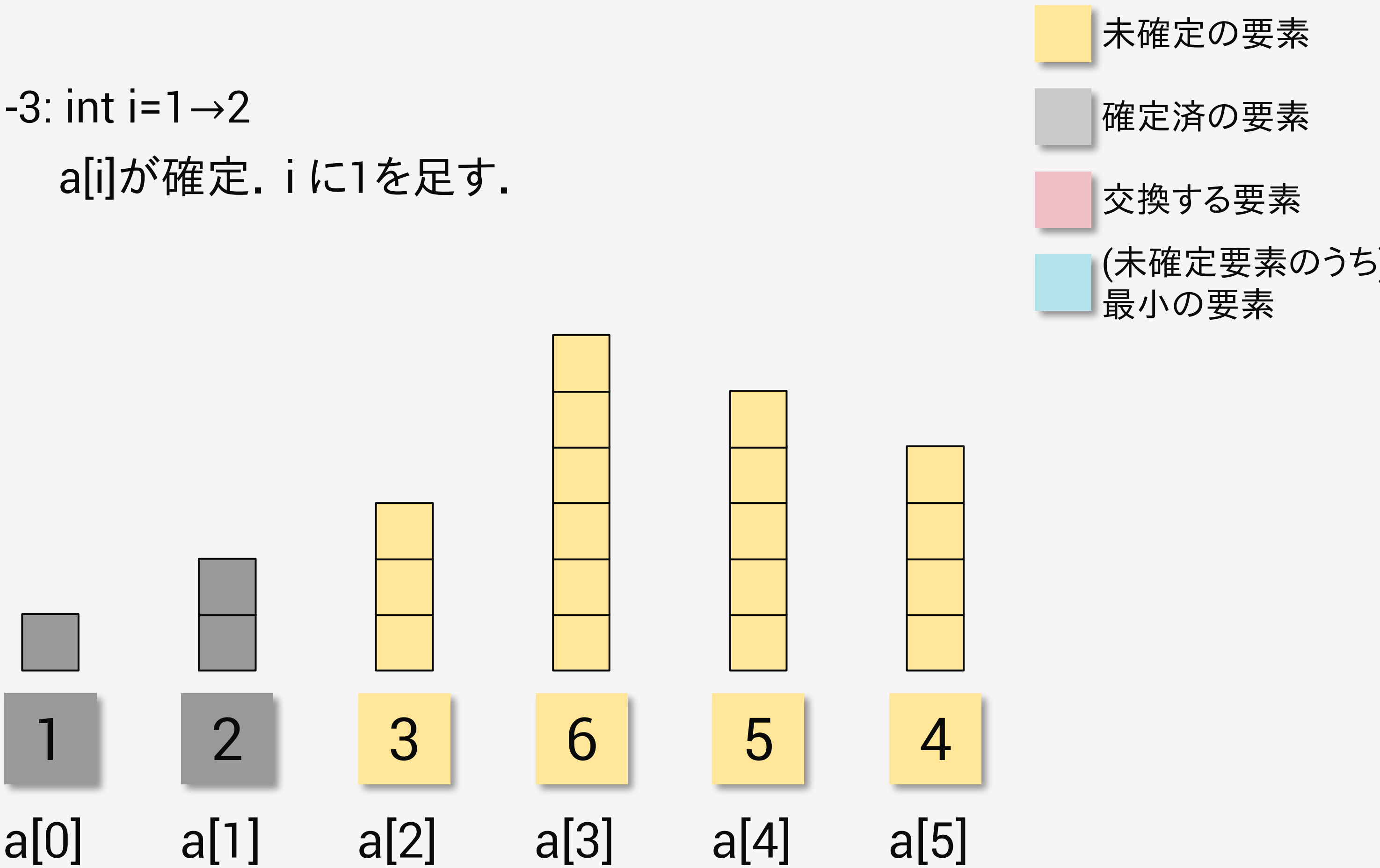


入力サイズ $n = 6$ の場合の例

選択ソートの例

Step 1-3: int i=1→2

a[i]が確定. i に1を足す.



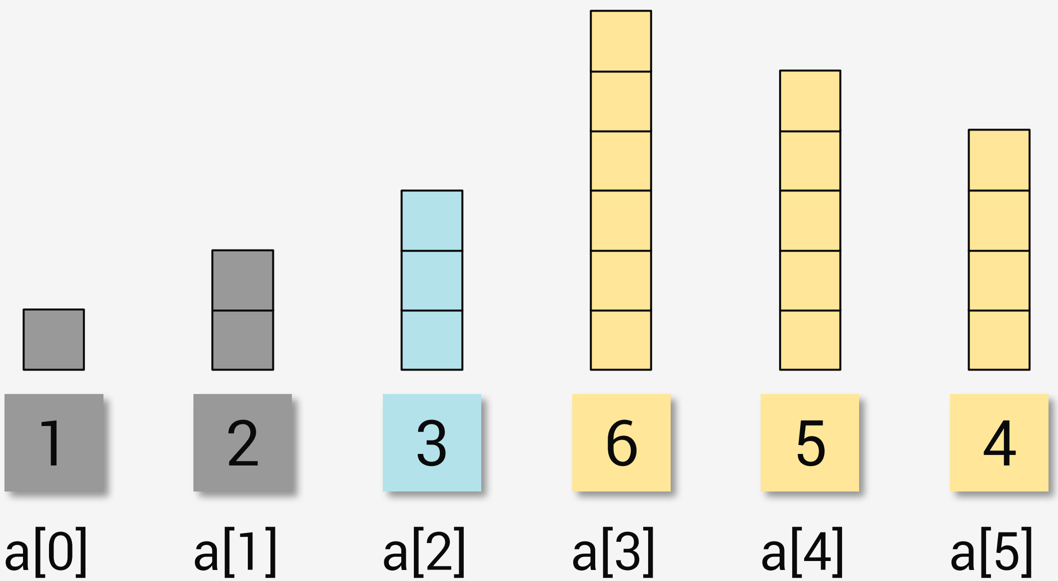
入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素
確定済の要素
交換する要素
(未確定要素のうち)
最小の要素

Step 2-1: int i=2

a[i] 〜 a[n-1] の中で最小の値を探す

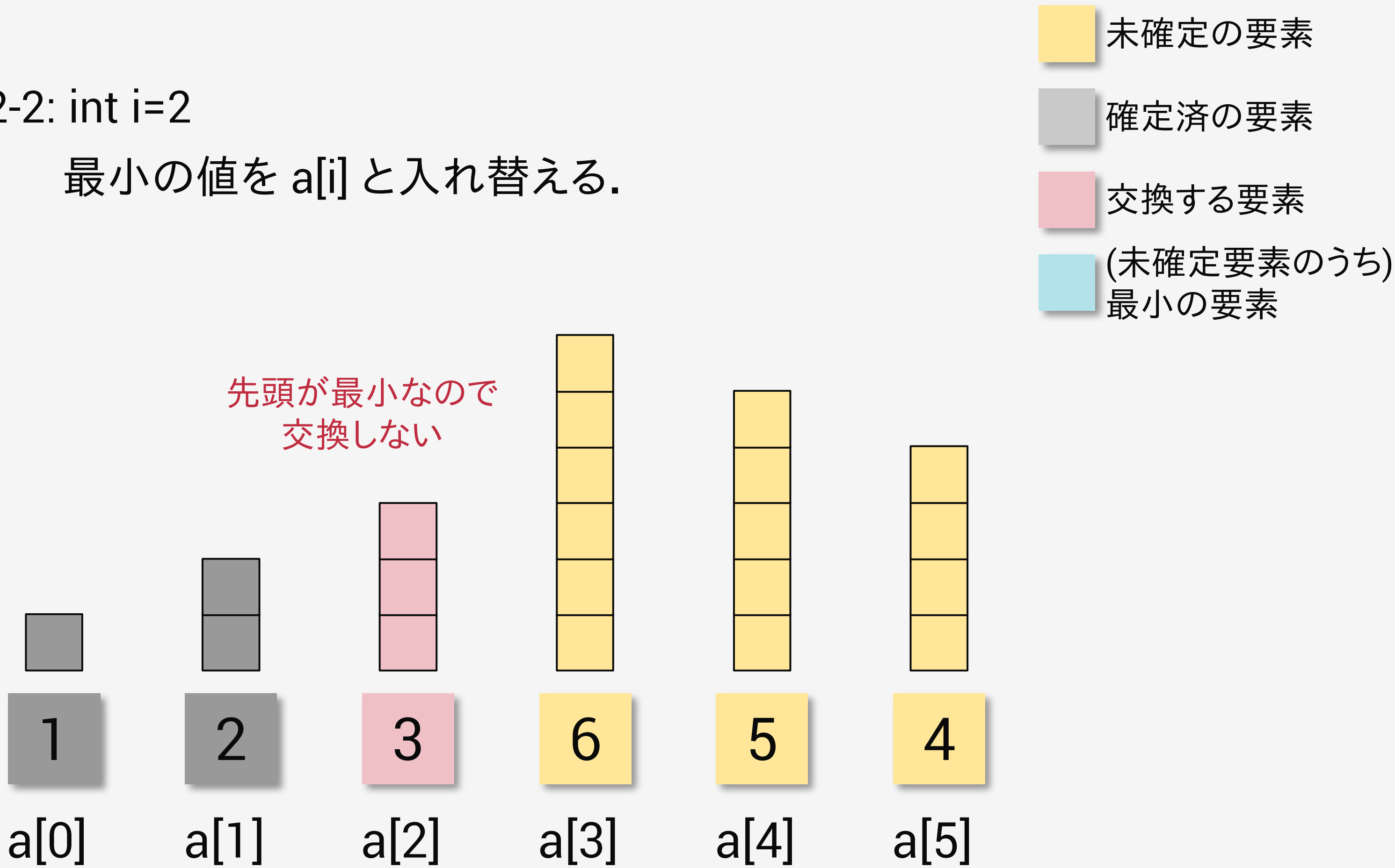


入力サイズ $n = 6$ の場合の例

選択ソートの例

Step 2-2: int i=2

最小の値を a[i] と入れ替える.



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

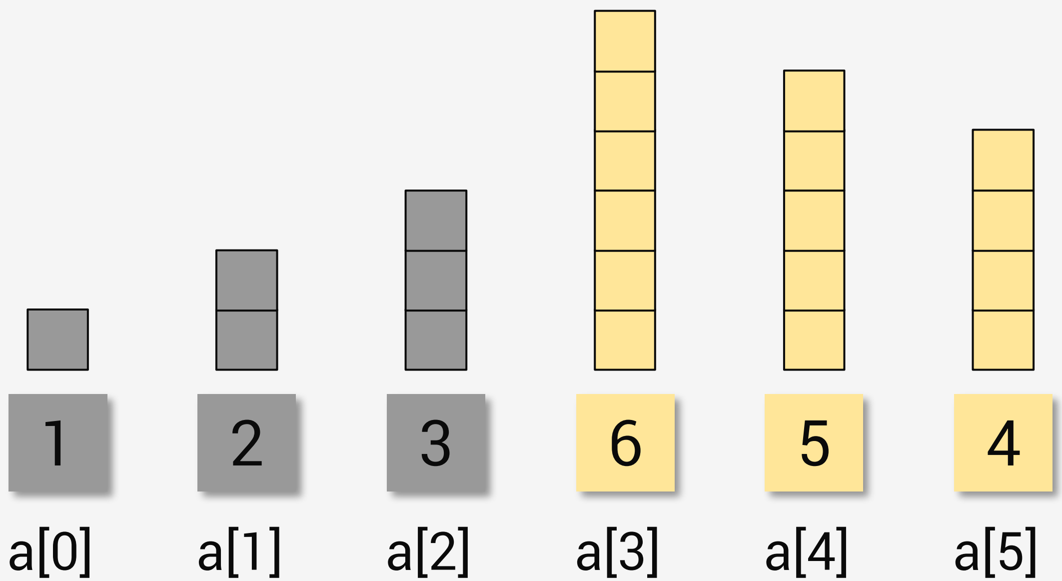
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 2-3: int i=2→3

a[i]が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

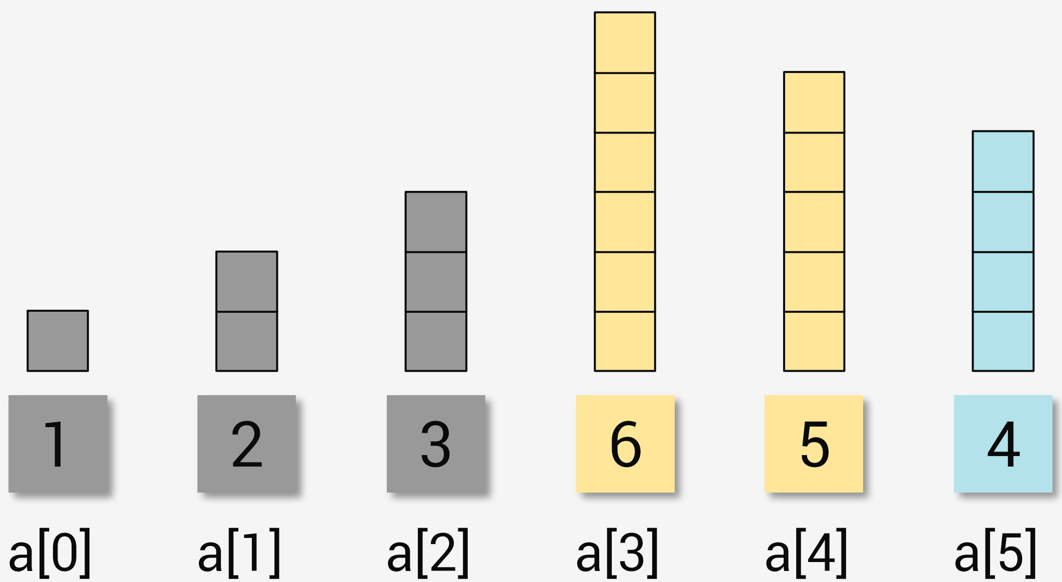
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 3-1: int i=3

a[i] 〜 a[n-1] の中で最小の値を探す

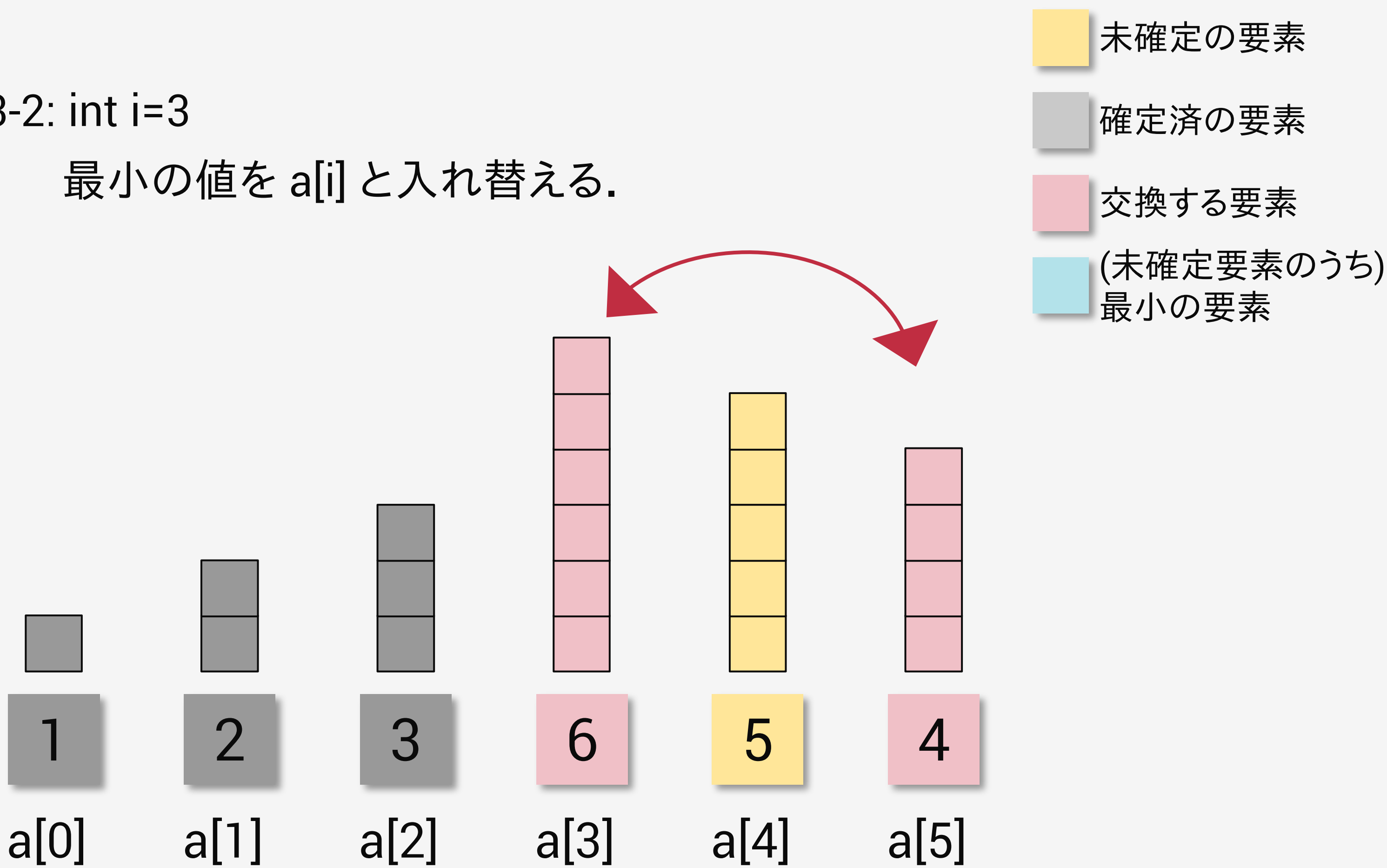


入力サイズ $n = 6$ の場合の例

選択ソートの例

Step 3-2: int i=3

最小の値を a[i] と入れ替える.



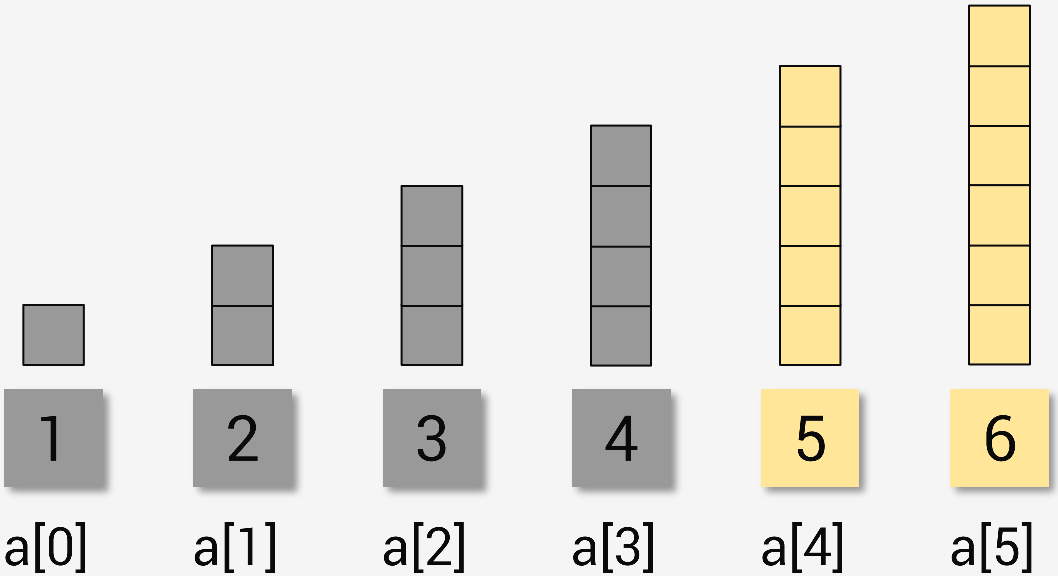
入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素
確定済の要素
交換する要素
(未確定要素のうち)
最小の要素

Step 3-3: int i=3→4

a[i]が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素

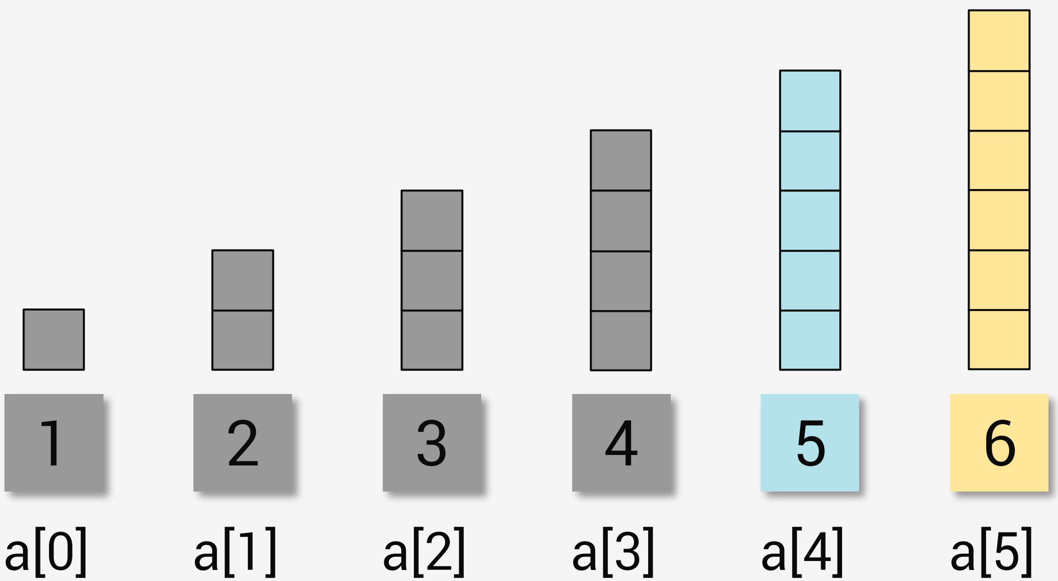
確定済の要素

交換する要素

(未確定要素のうち)
最小の要素

Step 4-1: int i=4

a[i] ～ a[n-1] の中で最小の値を探す



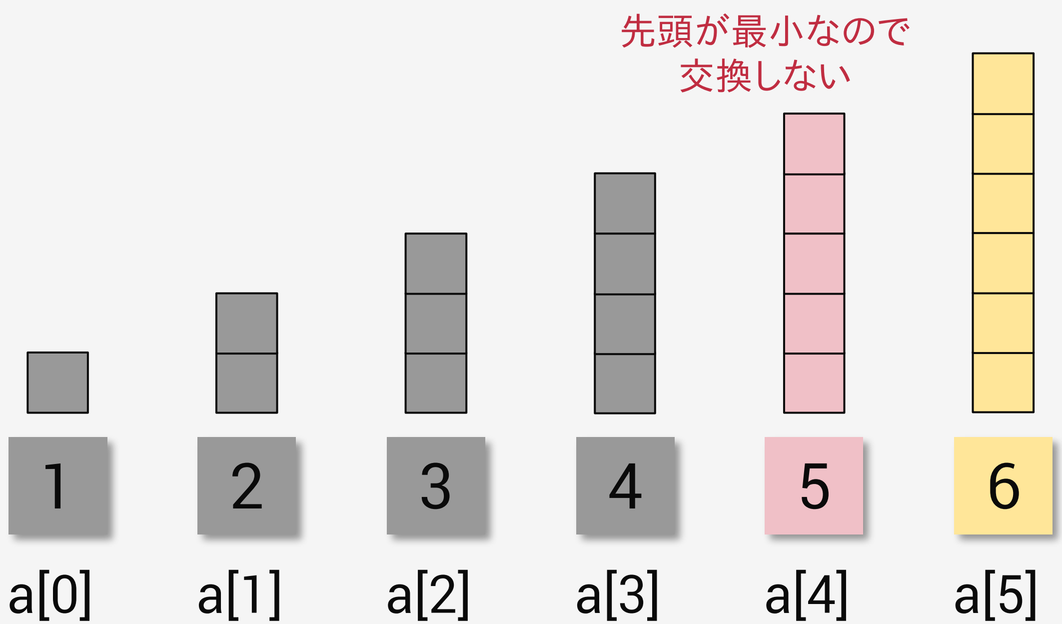
入力サイズ $n = 6$ の場合の例

選択ソートの例

未確定の要素
確定済の要素
交換する要素
(未確定要素のうち)
最小の要素

Step 4-2: int i=4

最小の値を a[i] と入れ替える.

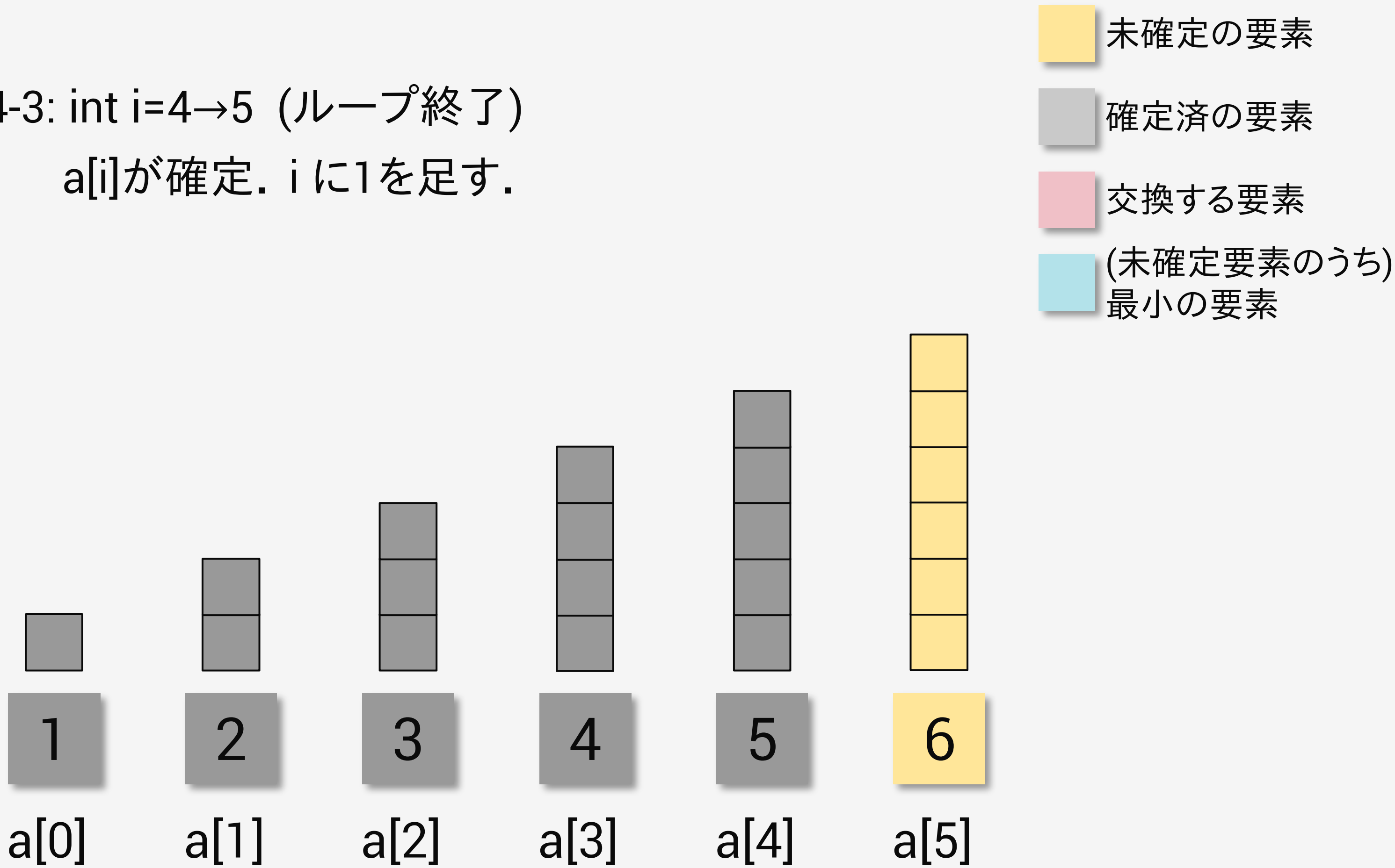


入力サイズ $n = 6$ の場合の例

選択ソートの例

Step 4-3: int i=4→5 (ループ終了)

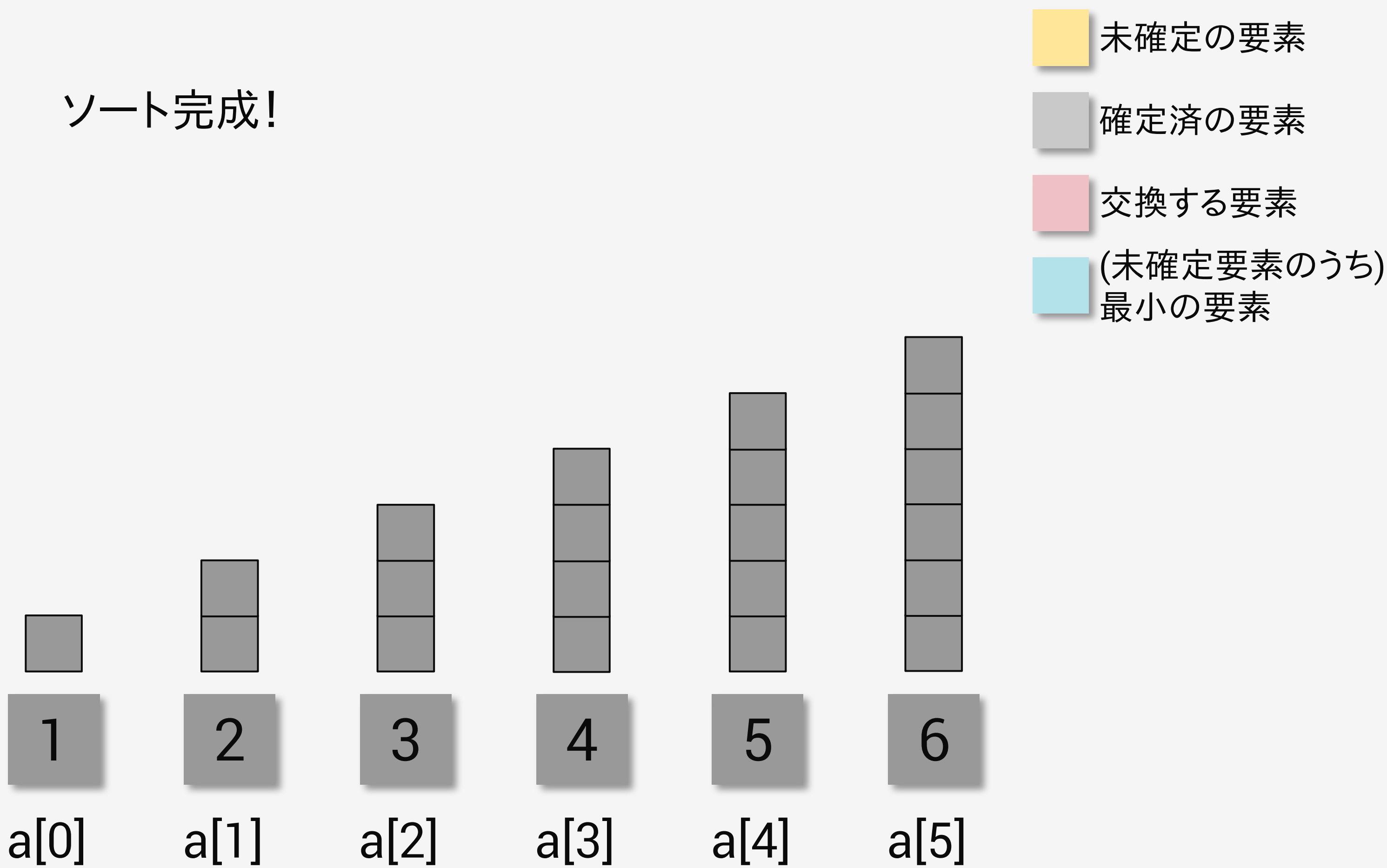
a[i]が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

選択ソートの例

ソート完成！



入力サイズ $n = 6$ の場合の例

選択ソートの実装

```
void select_sort(int a[], int n)
{
  for( int i=0 ; i<n-1 ; i++){

    // 最小値(の添え字)を見つける(線形探索)
    int mind{i};
    for( int j=i+1 ; j<n ; j++ ){
      if( a[j] < a[mind] )
        mind = j;
    }

    // 先頭と最小値を入れ替える std::swap(a[i],a[mind]); と同じ
    int tmp {a[i]};
    a[i] = a[mind];
    a[mind] = tmp;
  }
}
```

バブルソート

- 配列a の連続する2要素 a[j-1] と a[j] に対し, a[j-1]>a[j] ならば, その2つの要素を 入れ替える, という操作を j=n-1, n-2, ... , 1 に対して繰り返す.
  この操作を パス という. これで, 最小の要素が先頭に来たので, 先頭の要素は確定.
- 次に, 2番目の要素から, 末尾の要素までに対して, もう一度パス操作を行う.
  (j=n-1, n-2, ... ,2 に対して繰り返す.) これで, 2番目の要素が確定.

⋮ 最後の要素 (最後から2番目の要素) が確定するまで
$(n-1)$回のパス操作を繰り返す.

「要素を交換する」(かどうかの判定) をひたすら $n(n-1)/2$ 回繰り返すアルゴリズム

バブルソートの例

未確定の要素

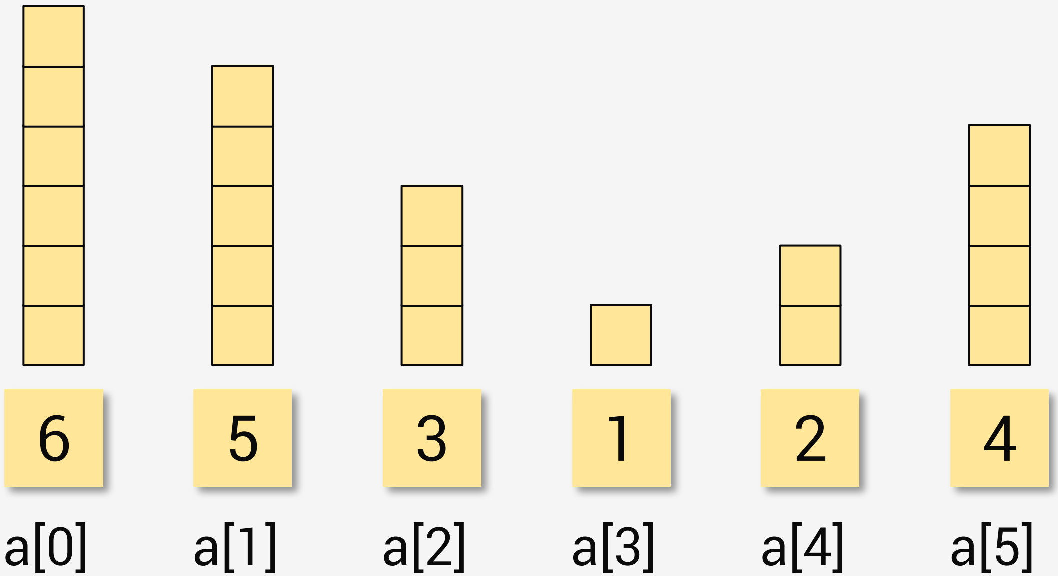
確定済の要素

比較・交換する要素

初期状態

ループ制御変数 i (初期値 1)を用意.

今度は1から $n-1$ まで



入力サイズ $n = 6$ の場合の例

バブルソートの例

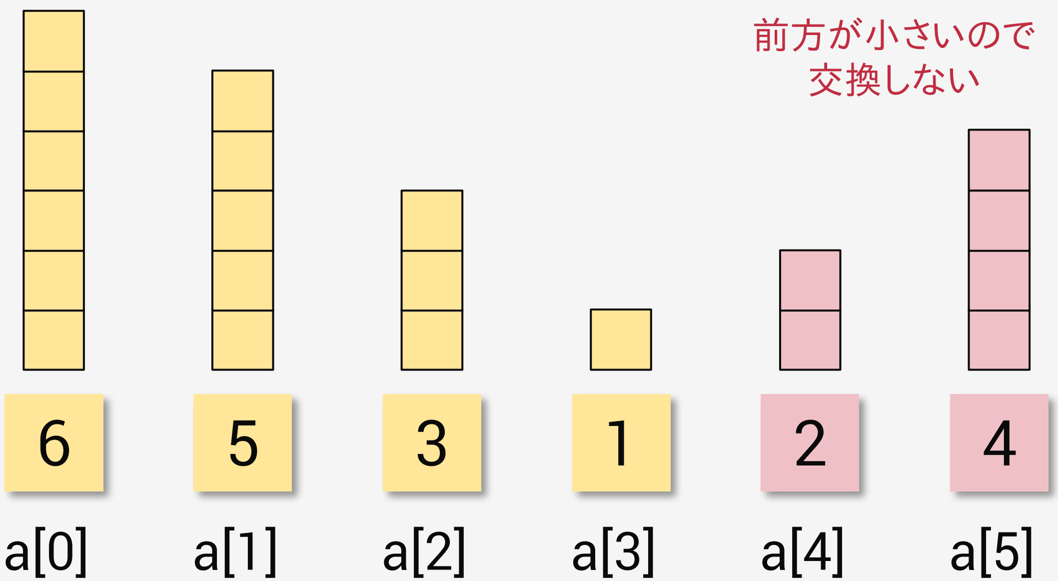
Step 1-1: 1回目のパス操作 int i=1

int j=5 (a[5] と a[4] の比較・交換)

未確定の要素

確定済の要素

比較・交換する要素

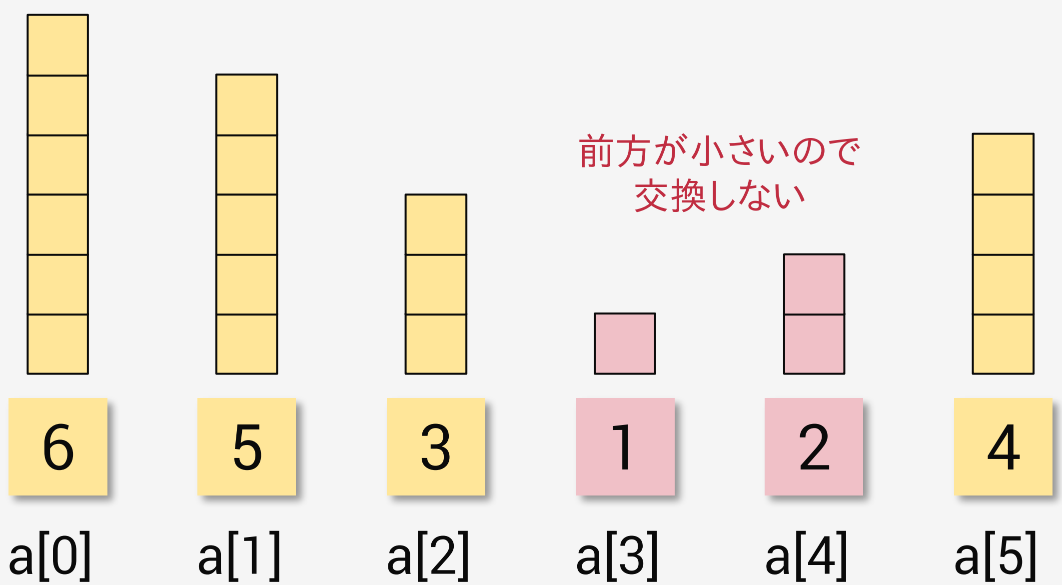


入力サイズ $n = 6$ の場合の例

バブルソートの例

未確定の要素
確定済の要素
比較・交換する要素

Step 1-2: 1回目のパス操作 int i=1
int j=4 (a[4] と a[3] の比較・交換)

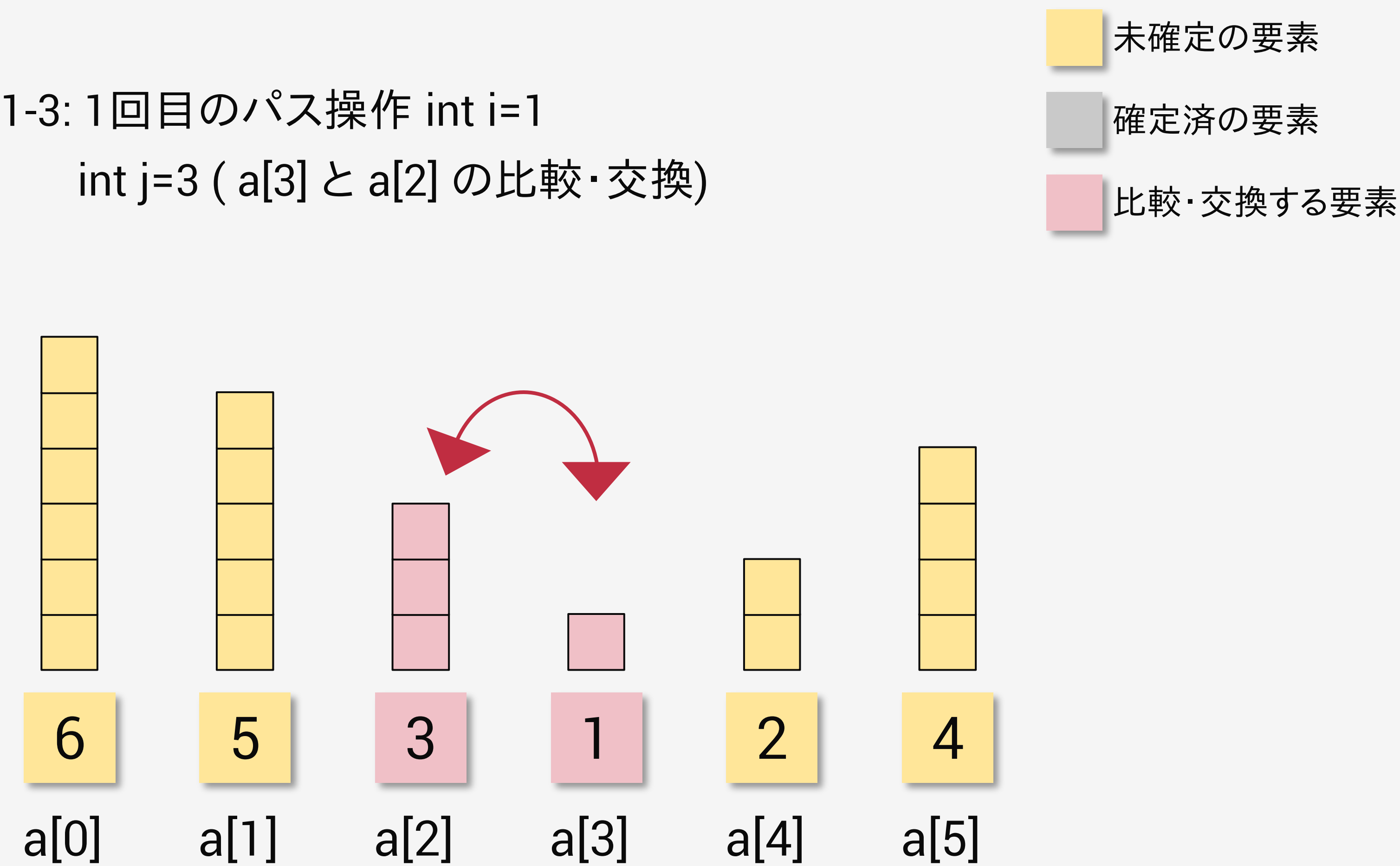


入力サイズ $n = 6$ の場合の例

バブルソートの例

Step 1-3: 1回目のパス操作 int i=1

int j=3 (a[3] と a[2] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

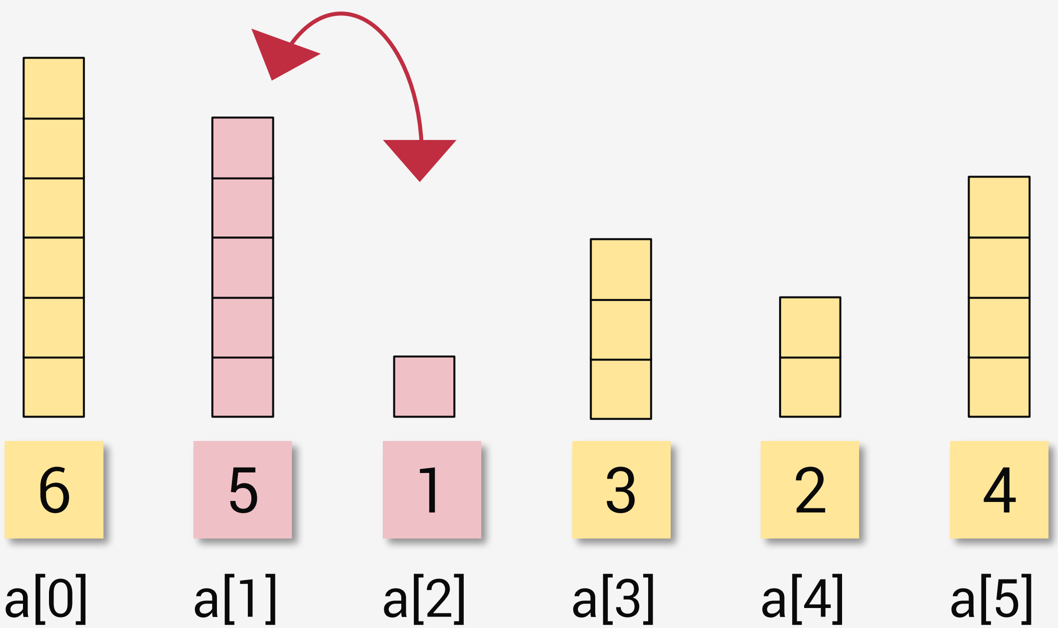
未確定の要素

確定済の要素

比較・交換する要素

Step 1-4: 1回目のパス操作 int i=1

int j=2 (a[2] と a[1] の比較・交換)

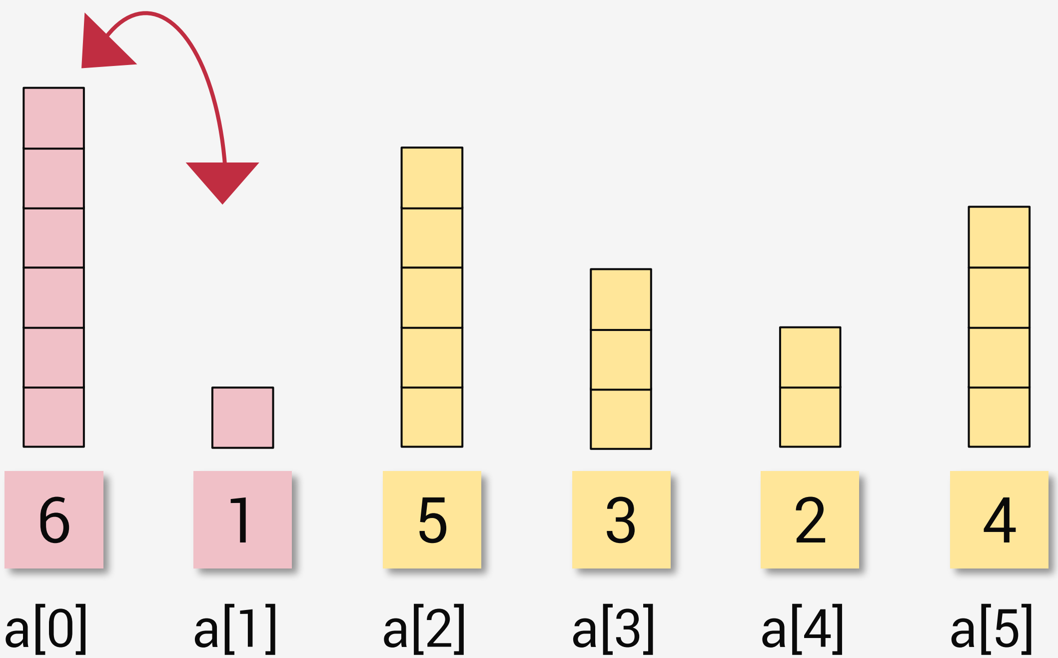


入力サイズ $n = 6$ の場合の例

バブルソートの例

未確定の要素
確定済の要素
比較・交換する要素

Step 1-5: 1回目のパス操作 int i=1
int j=1 (a[1] と a[0] の比較・交換)

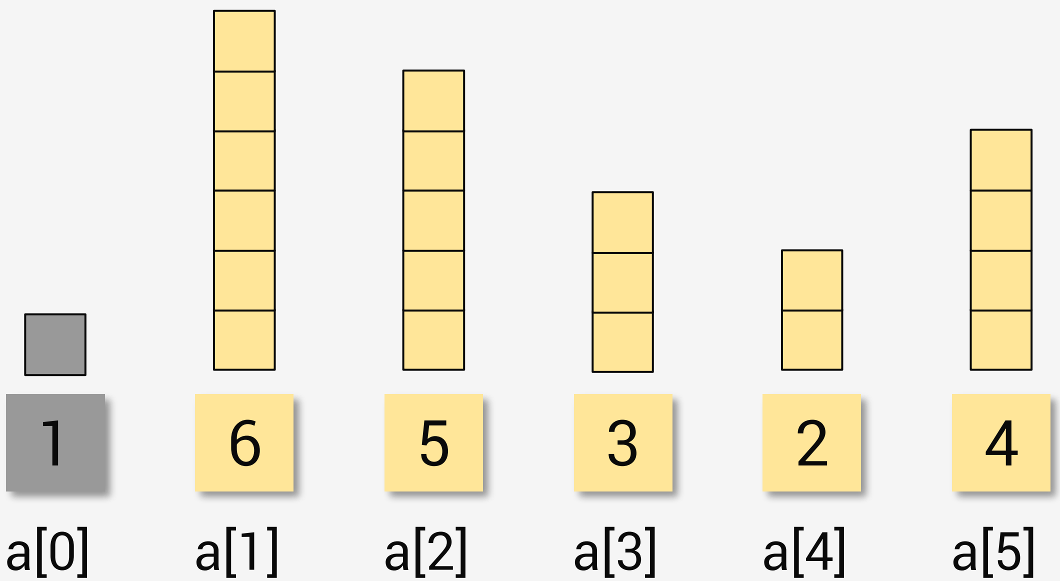


入力サイズ $n = 6$ の場合の例

バブルソートの例

未確定の要素
確定済の要素
比較・交換する要素

Step 1-6: 1回目のパス操作 int i=1→2
a[0] が確定. i に1を足す.



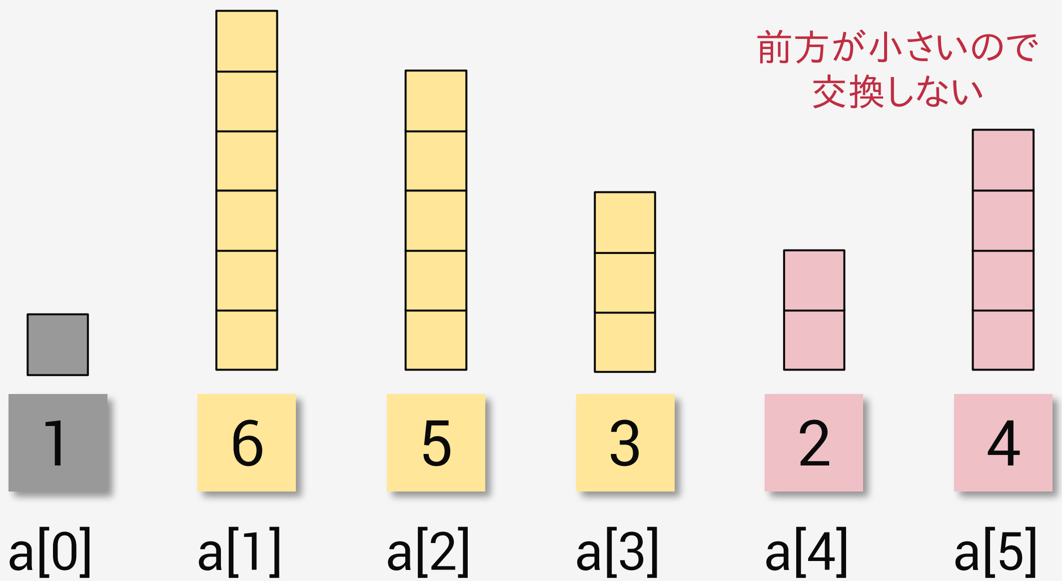
入力サイズ $n=6$ の場合の例

バブルソートの例

Step 2-1: 2回目のパス操作 int i=2

int j=5 (a[5] と a[4] の比較・交換)

- 未確定の要素
- 確定済の要素
- 比較・交換する要素



入力サイズ $n = 6$ の場合の例

バブルソートの例

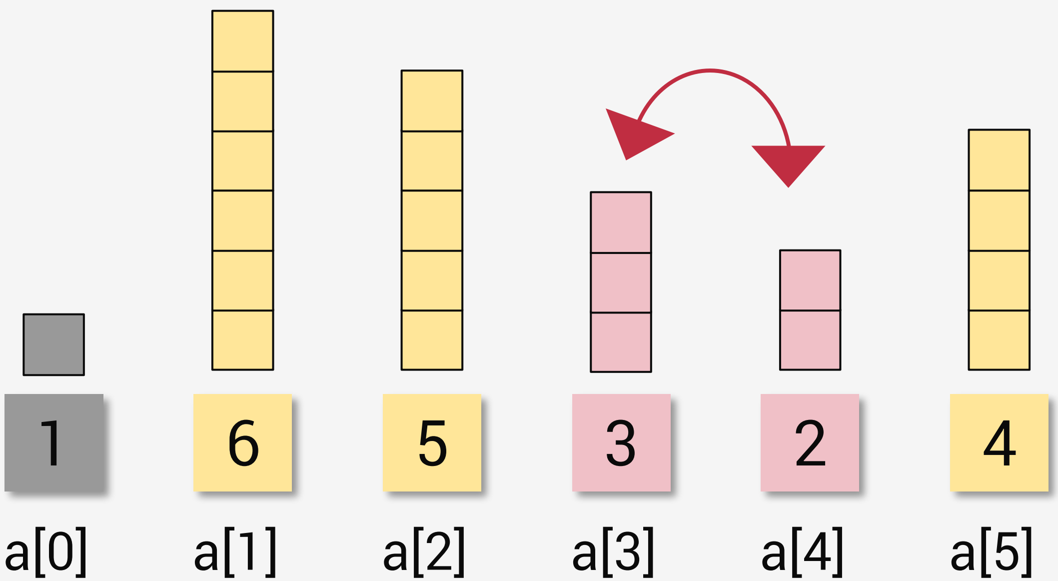
未確定の要素

確定済の要素

比較・交換する要素

Step 2-2: 2回目のパス操作 int i=2

int j=4 (a[4] と a[3] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

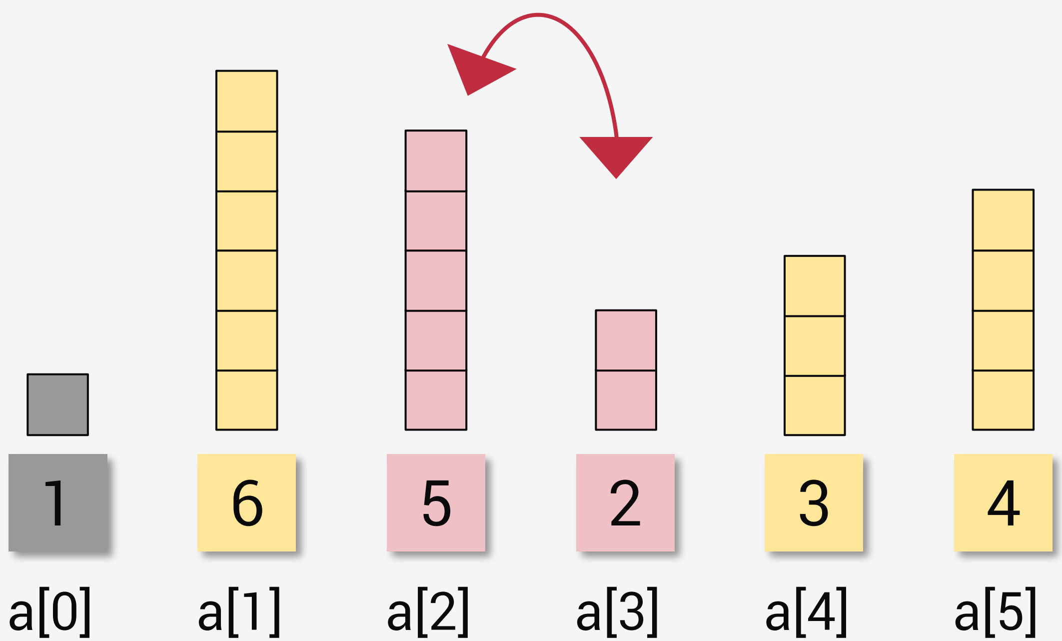
未確定の要素

確定済の要素

比較・交換する要素

Step 2-3: 2回目のパス操作 int i=2

int j=3 (a[3] と a[2] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

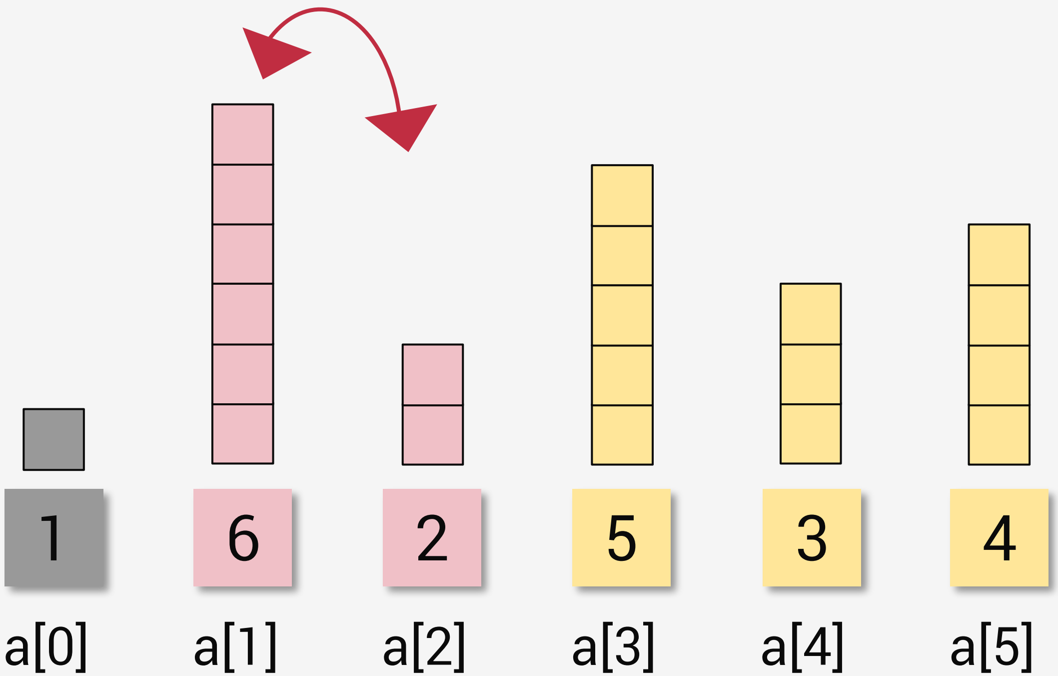
Step 2-4: 2回目のパス操作 int i=2

int j=2 (a[2] と a[1] の比較・交換)

未確定の要素

確定済の要素

比較・交換する要素



入力サイズ $n = 6$ の場合の例

バブルソートの例

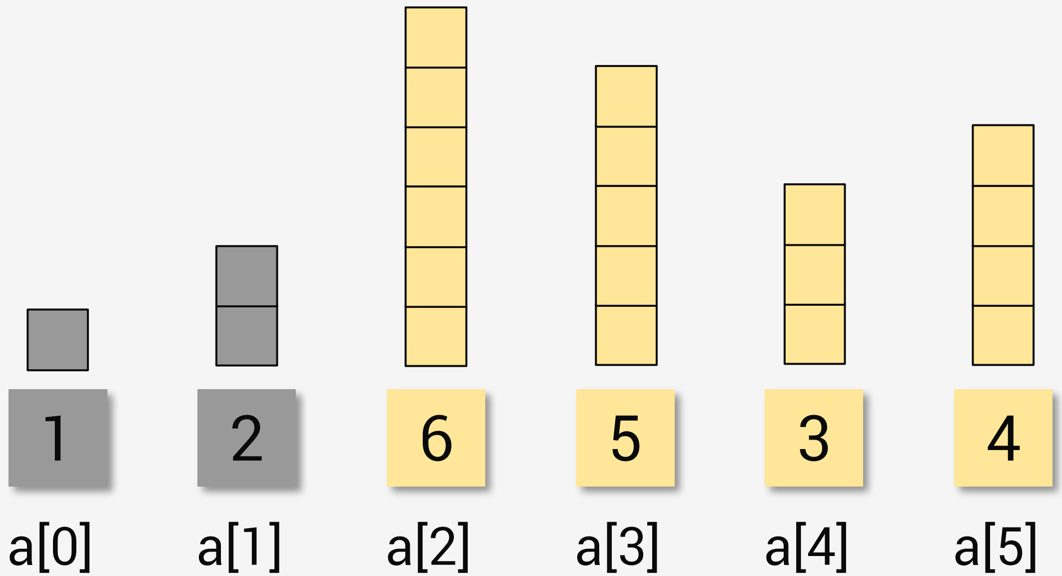
未確定の要素

確定済の要素

比較・交換する要素

Step 2-5: 2回目のパス操作 int i=2→3

a[1] が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

バブルソートの例

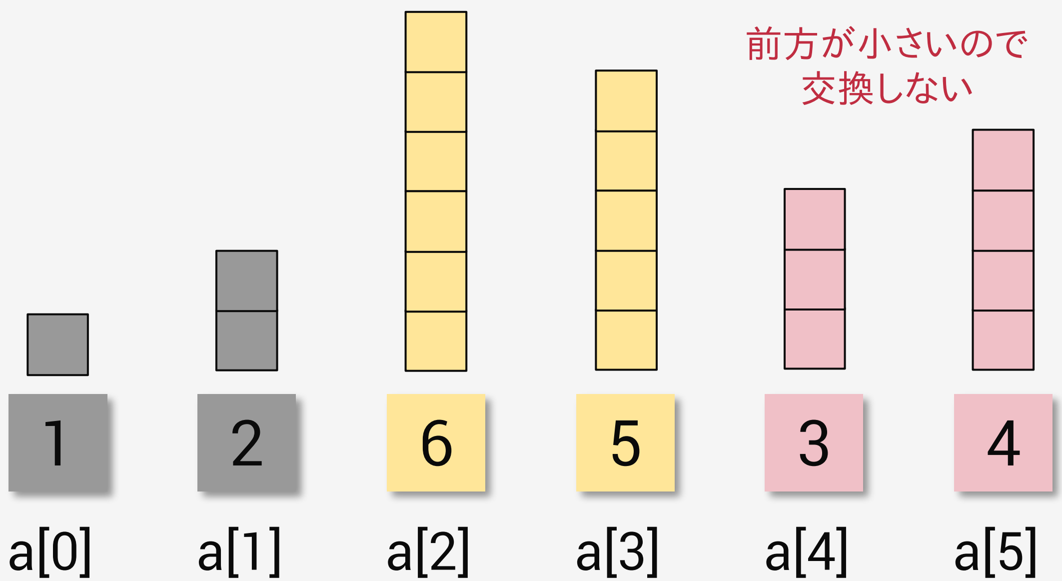
未確定の要素

確定済の要素

比較・交換する要素

Step 3-1: 3回目のパス操作 int i=3

int j=5 (a[5] と a[4] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

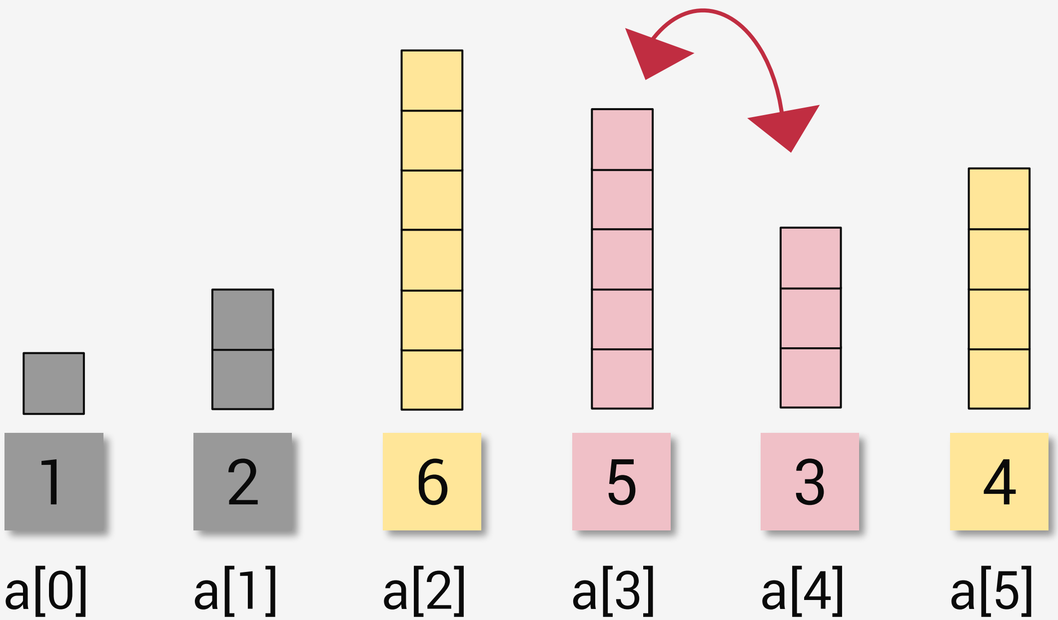
未確定の要素

確定済の要素

比較・交換する要素

Step 3-2: 3回目のパス操作 int i=3

int j=4 (a[4] と a[3] の比較・交換)

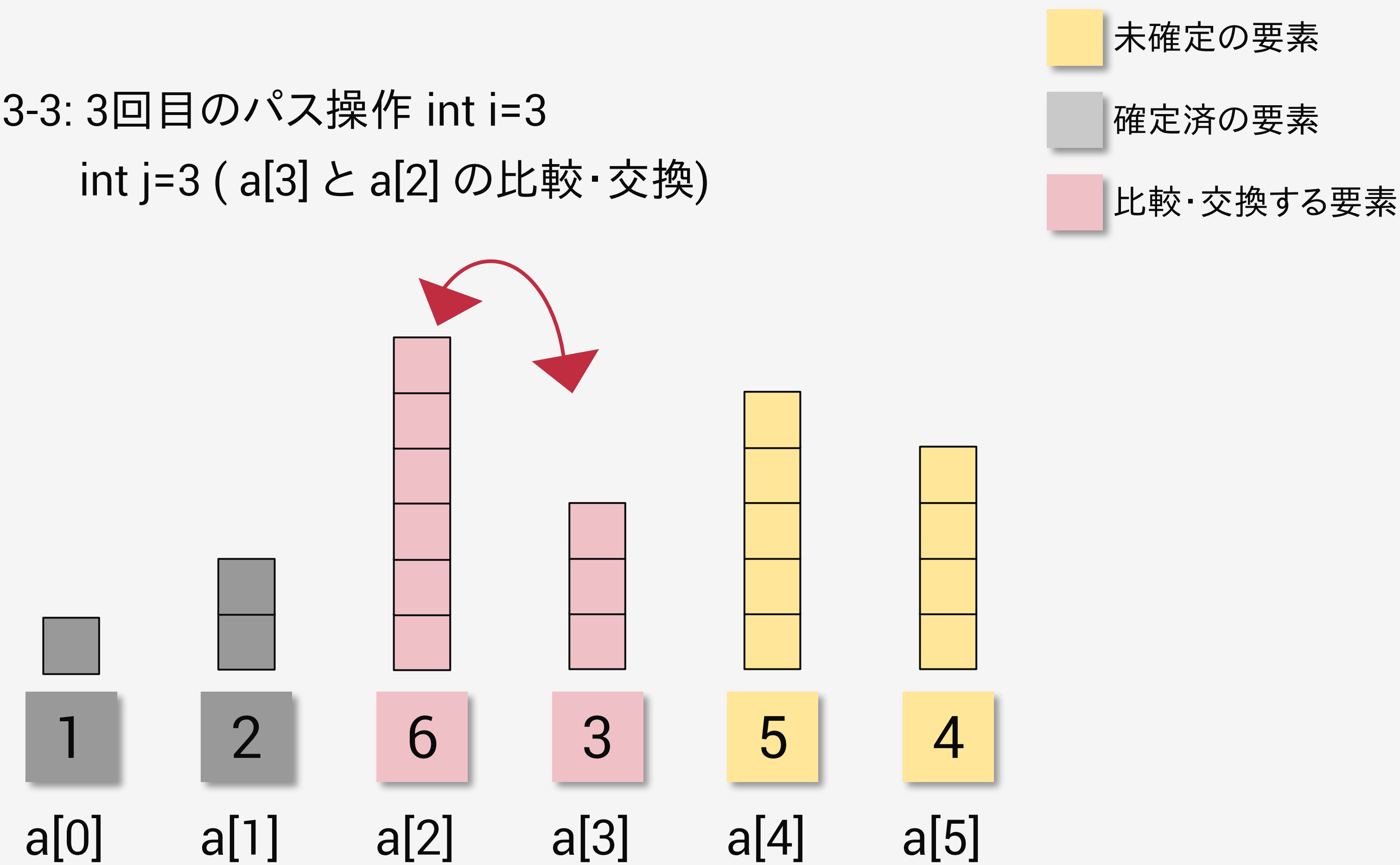


入力サイズ $n = 6$ の場合の例

バブルソートの例

Step 3-3: 3回目のパス操作 int i=3

int j=3 (a[3] と a[2] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

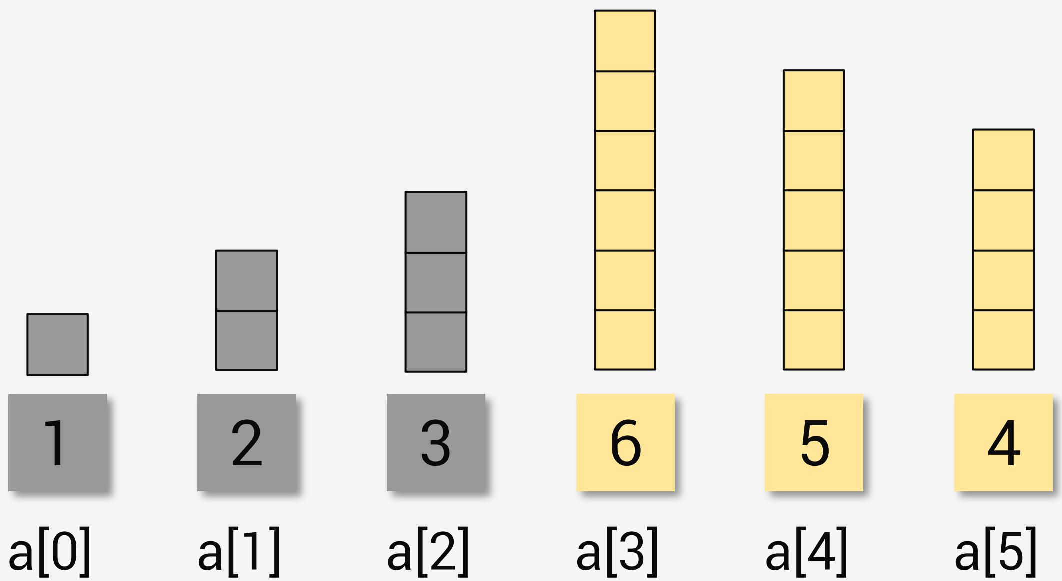
未確定の要素

確定済の要素

比較・交換する要素

Step 3-4: 3回目のパス操作 int i=3→4

a[2] が確定. i に1を足す.

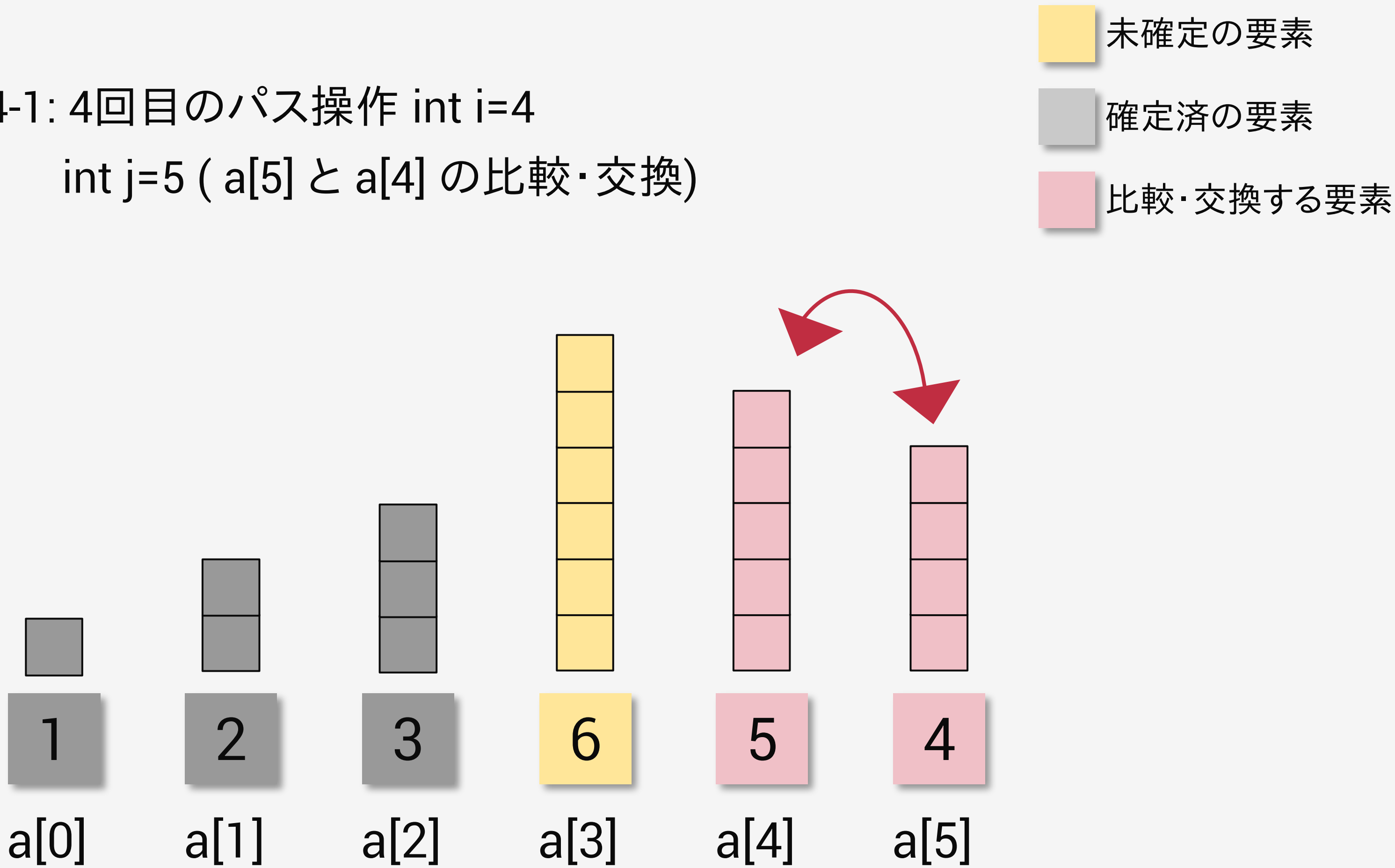


入力サイズ $n = 6$ の場合の例

バブルソートの例

Step 4-1: 4回目のパス操作 int i=4

int j=5 (a[5] と a[4] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

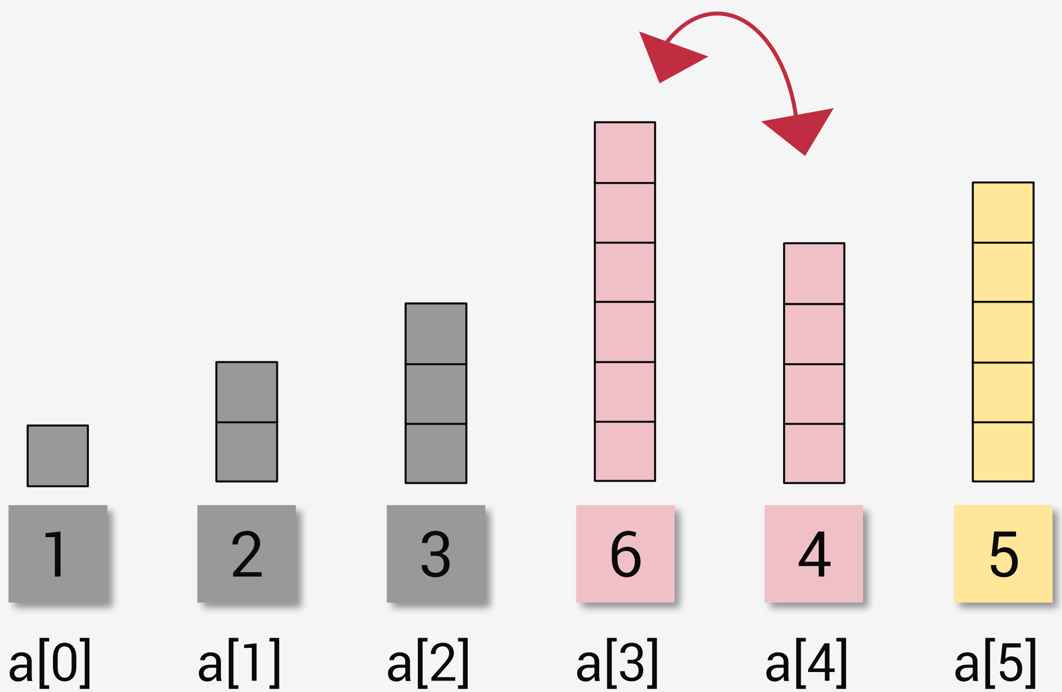
未確定の要素

確定済の要素

比較・交換する要素

Step 4-2: 4回目のパス操作 int i=4

int j=4 (a[4] と a[3] の比較・交換)

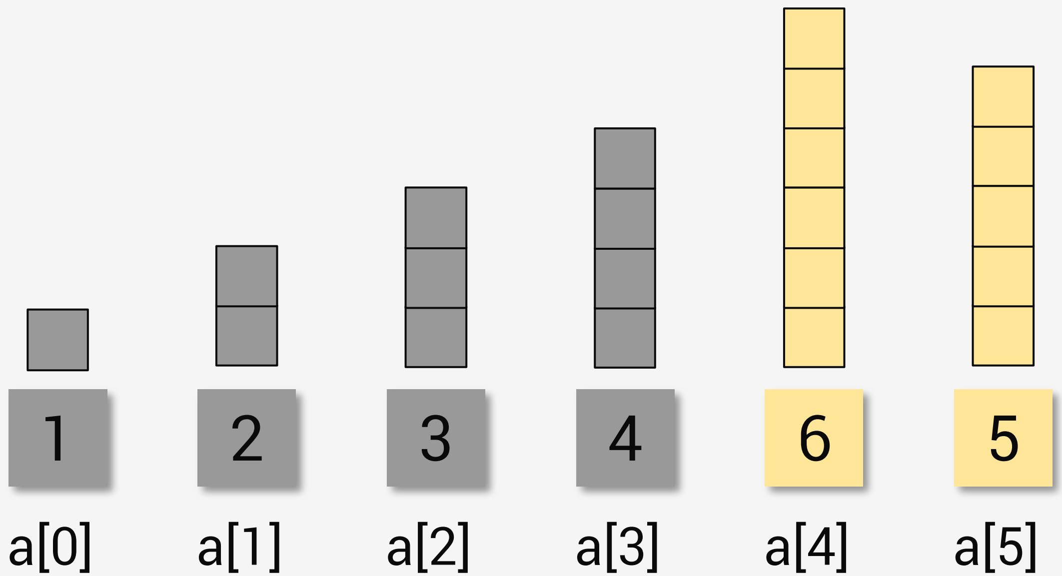


入力サイズ $n = 6$ の場合の例

バブルソートの例

未確定の要素
確定済の要素
比較・交換する要素

Step 4-3: 4回目のパス操作 int i=4→5
a[3] が確定. i に1を足す.



入力サイズ $n = 6$ の場合の例

バブルソートの例

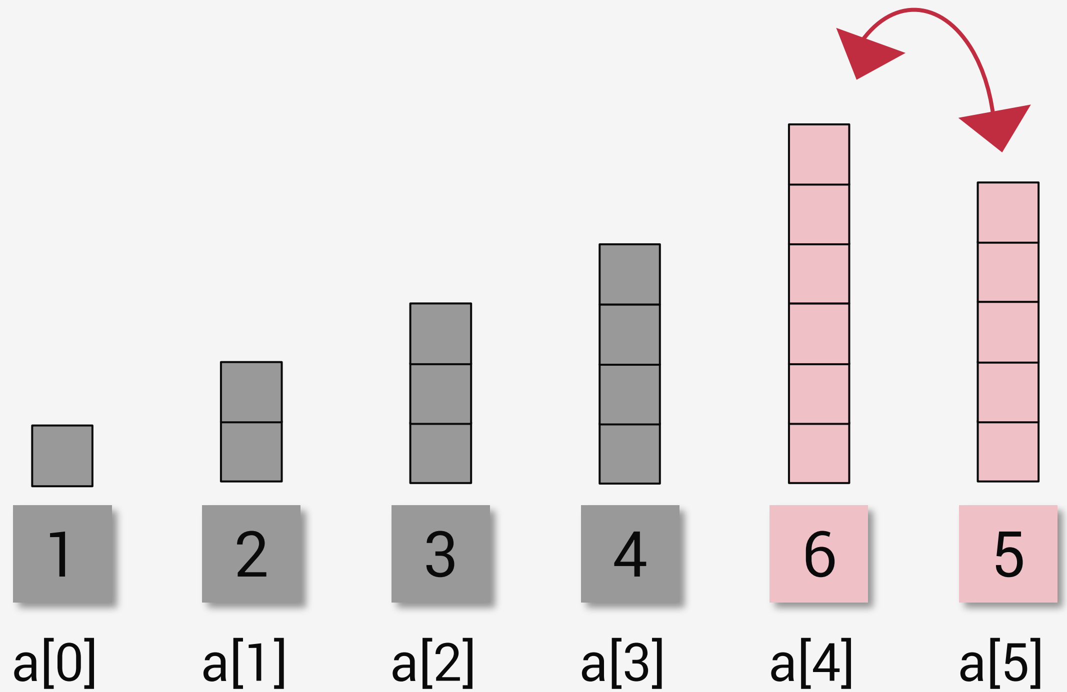
未確定の要素

確定済の要素

比較・交換する要素

Step 5-1: 5回目のパス操作 int i=5

int j=5 (a[5] と a[4] の比較・交換)



入力サイズ $n = 6$ の場合の例

バブルソートの例

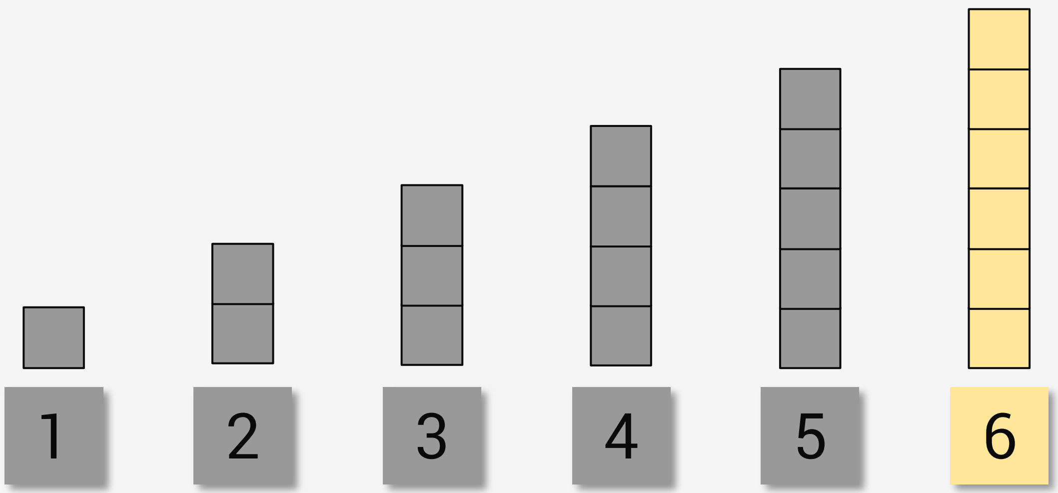
未確定の要素

確定済の要素

比較・交換する要素

Step 5-2: 5回目のパス操作 int i=5→6

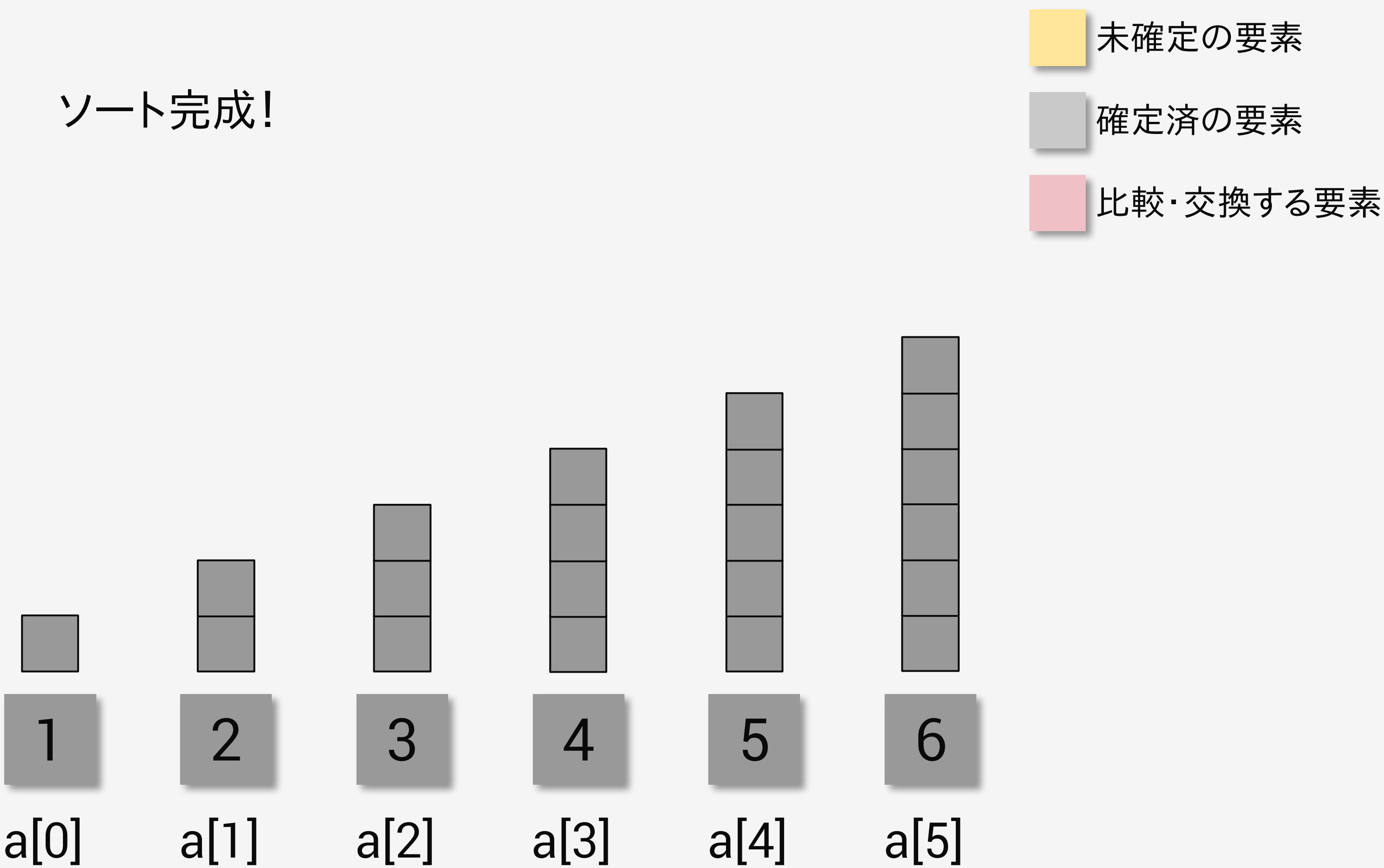
a[4] が確定. i に1を足す. (ループ終了)



a[0] a[1] a[2] a[3] a[4] a[5]

入力サイズ $n = 6$ の場合の例

バブルソートの例



入力サイズ $n = 6$ の場合の例

バブルソートの実装

```
void buble_sort(int a[], int n)
{
  for( int i=1 ; i<n ; i++){

    //パス操作
    for( int j=n-1 ; j>=i ; j--){

      //比較・交換
      if( a[j] < a[j-1] ){
        int tmp{a[j]};
        a[j] = a[j-1];
        a[j-1] = tmp;
      }
    }
  }
}
```

クイックソートの概要

- もし, 要素が1つ以下なら, 何もせずソート終了.
- 要素が2つ以上の場合, 基準となる要素 (ピボット) を一つ決める. どれでも良いが, ここでは真ん中の要素 a[n/2] を選ぶことにする.
- 配列を「ピボットの値以上のグループ」と「ピボットの値以下のグループ」に分ける. (ピボット自身は高々一方のグループに含まれる.)
- 2つのグループそれぞれでクイックソートを再帰的に行う.

クイックソートは再帰呼び出しを使ったアルゴリズム

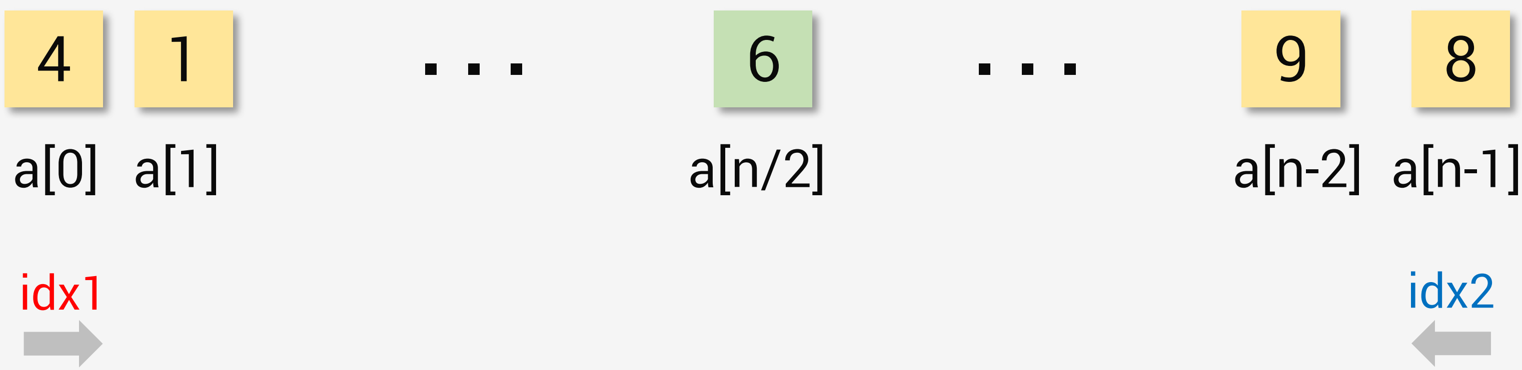
グループ分け

1. 2つのインデックス(添え字用の整数) idx1, idx2 を用意して，
   それぞれの初期値は配列の先頭，末尾の添え字とする (最初はそれぞれ，0, n-1).
2. ピボットの値(a[(idx1+idx2)/2]) を pvt とする.
3. a[idx1] ≧ pvt となるまで，idx1を1ずつ増やしていく.
4. 同様に，a[idx2] ≦ pvt となるまで idx2 を1ずつ減らしていく.
5. a[idx1] と a[idx2] を入れ替える (idx1=idx2 の場合もある).
6. idx1に1増やし, idx2 に1減らす. idx1 ≧ idx2 となるまで，再び 3. ～ 5. を繰り返す.

クイックソートの例

初期状態

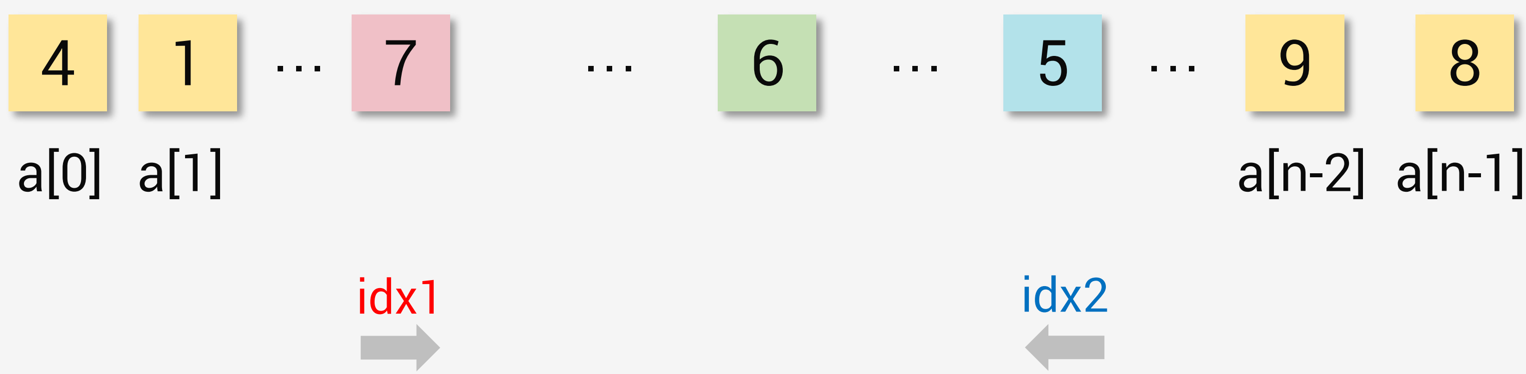
ピボットは a[n/2]，idx1は左端から右に，idx2は右端から左に動かしていく.



クイックソートの例

グループ分け

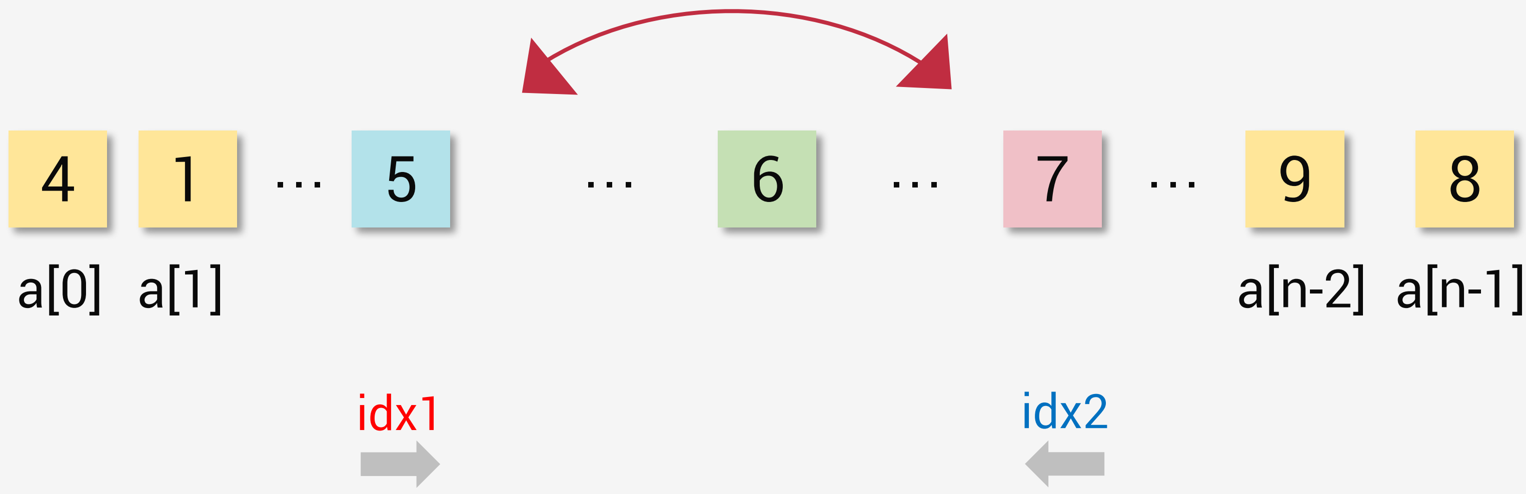
Step1： a[idx1] ≧ pvt, a[idx2] ≦ pvt となるまでそれぞれ idx1 idx2を動かす.



クイックソートの例

グループ分け

Step2：a[idx1] と a[idx2] を入れ替える.（idx1 idx2 の位置は変わらない)



クイックソートの例

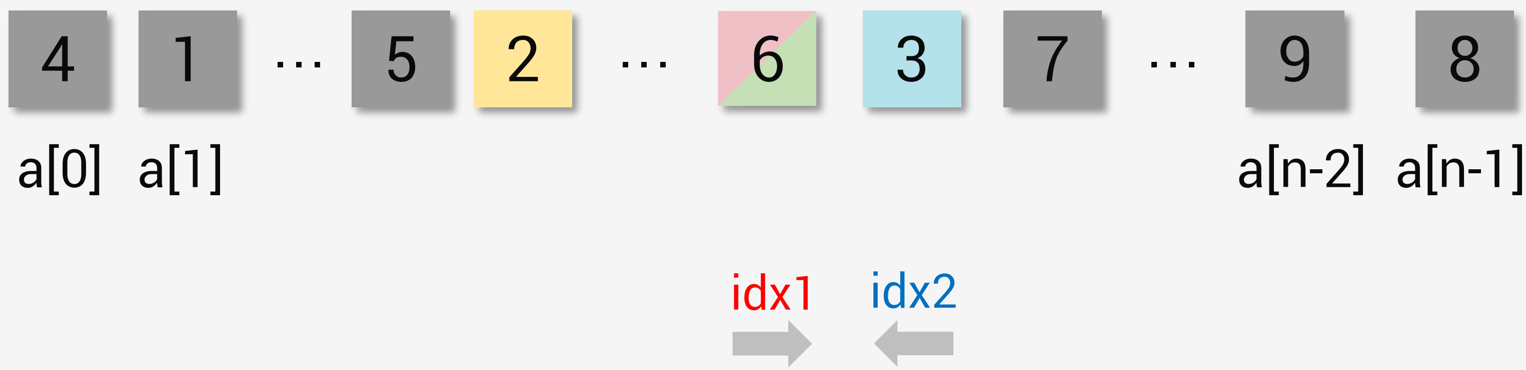
グループ分け

Step3：idx1 idx2 をそれぞれ一つ進めて，Step1, 2 を繰り返す．

クイックソートの例

グループ分け

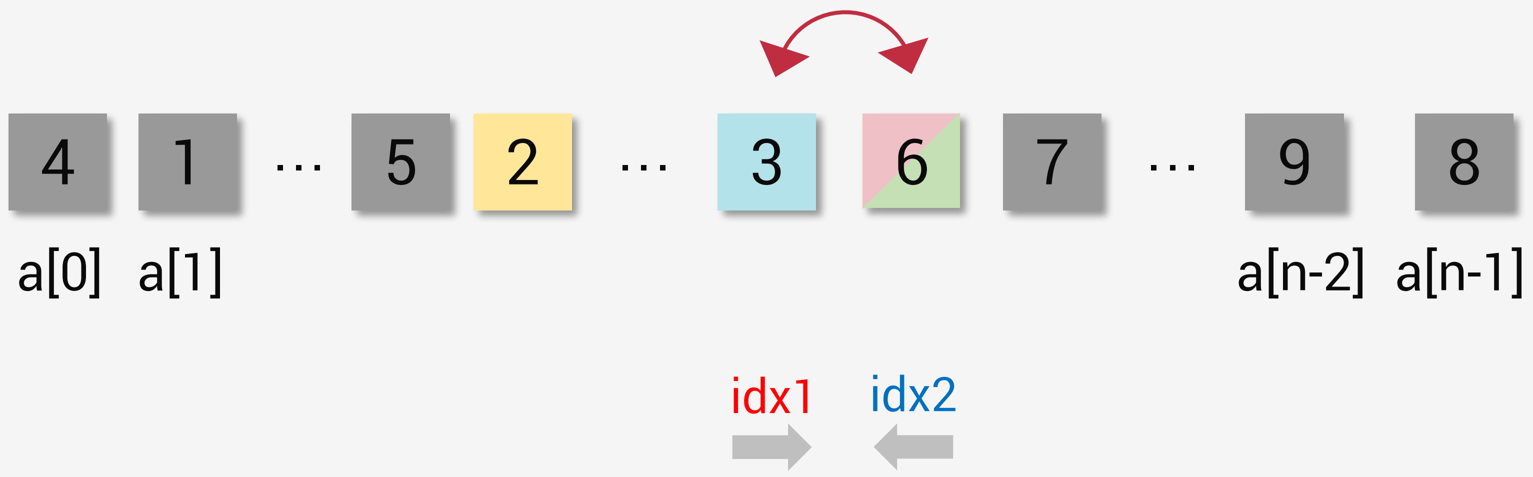
Step3：idx1 idx2 をそれぞれ一つ進めて，Step1, 2 を繰り返す．



クイックソートの例

グループ分け

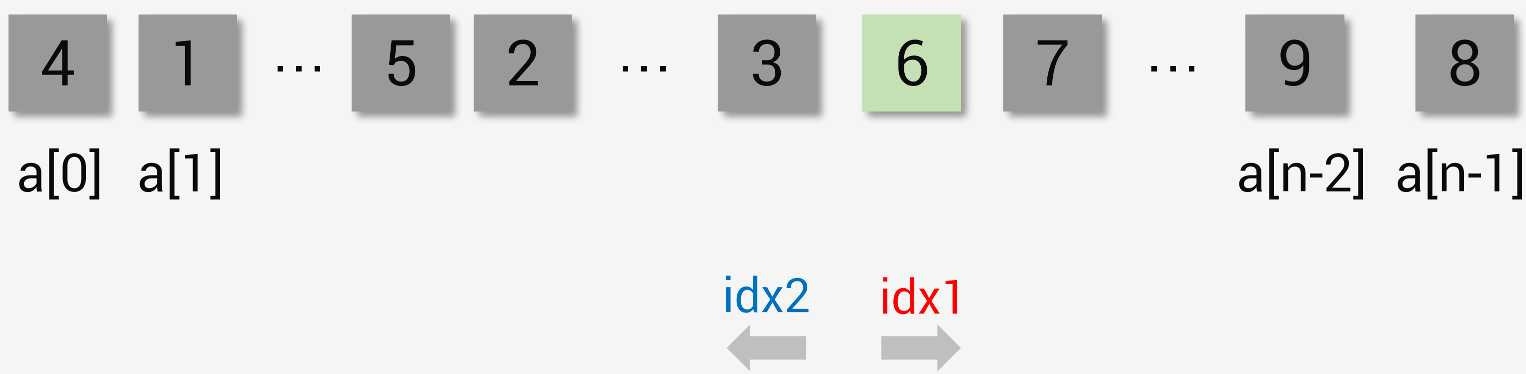
Step3：idx1 idx2 をそれぞれ一つ進めて，Step1, 2 を繰り返す．



クイックソートの例

グループ分け

Step3：idx1 idx2 をそれぞれ一つ進めて，Step1, 2 を繰り返す．

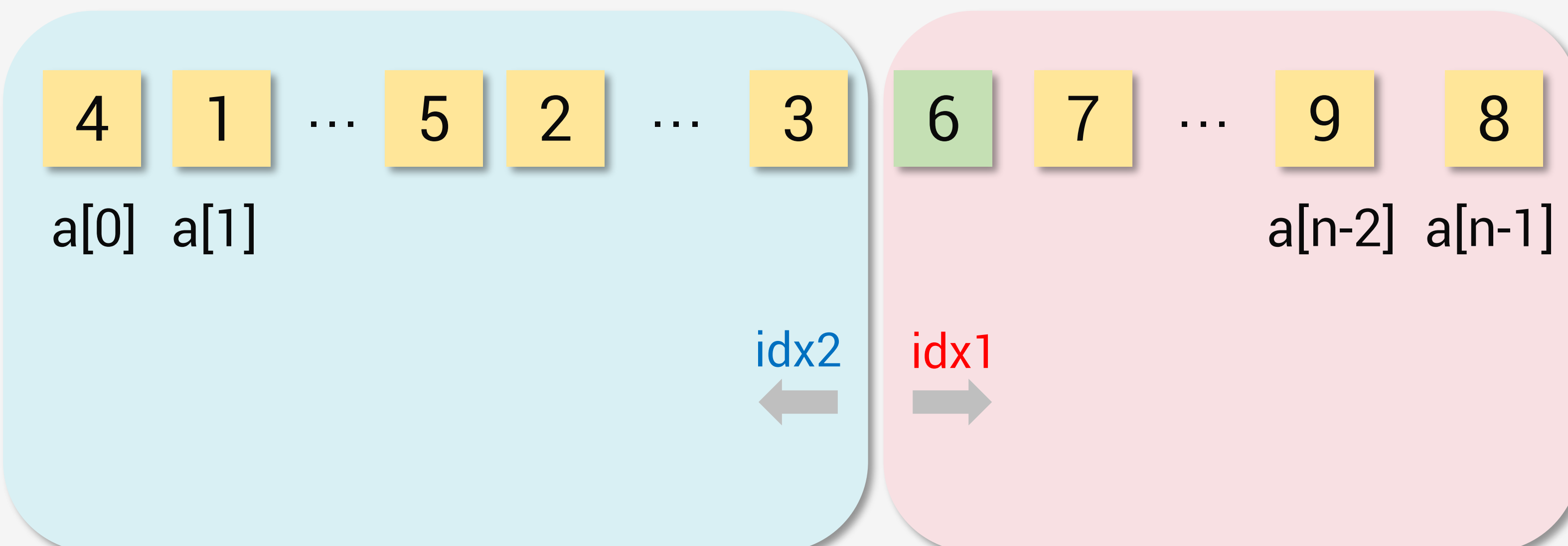


idx1 ≧ idx2 となったので，
グループ分け終了

クイックソートの例

再帰呼び出し

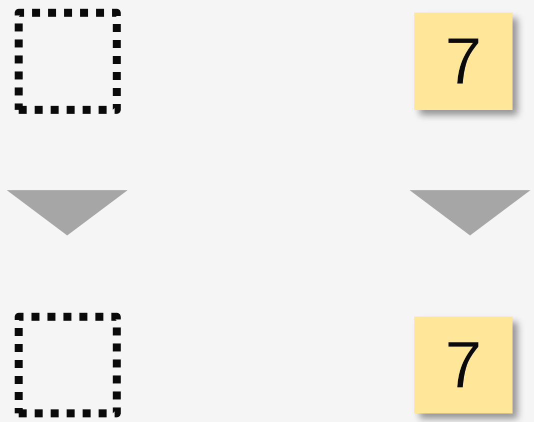
- ピボット以下グループ a[0],a[1], … , a[idx2] に対してクイックソートを実行
- ピボット以上グループ a[idx1], … , a[n-2], a[n-1] に対してクイックソートを実行



※ この例ではピボットが「ピボット以上グループ」に含まれているが,「ピボット以下グループ」に含まれる場合や, どちらにも含まれない場合もある.

クイックソートの終了条件

- もし, 要素が1つ以下なら, 何もせずソート終了.



本日はここまで

お疲れ様でした.

それでは, 演習課題 (クイックソートの実装) に取り組みましょう.


授業後でも質問があれば, 永並(s02967@cc.Seikei.ac.jp)まで