Sorting Algorithm

排序演算法

Algorithm的意思？

Algorithm noun [C]

（尤指電腦使用的）演算法，計算程式

a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem

Source: Cambridge Dictionary

Algorithm 演算法

- 用來解決特定問題的方法
- 有限指令或步驟

Algorithm 演算法

必須包含以下5種基本要素：

- 輸入：0個或以上的輸入資料
- 輸出：1個或以上輸出資料
- 明確性：演算法的描述必須無歧義
- 有限性：有限步驟內完成任務
- 有效性：有效可行

例如：煮飯的演算法

量米 ▶ 量米 ▶ 把米放進電鍋裡煮熟 ▶

輸入

等待米飯被煮熟 ▶ 盛飯

輸出

例如：煮飯的演算法

選擇排序法

Selection Sort

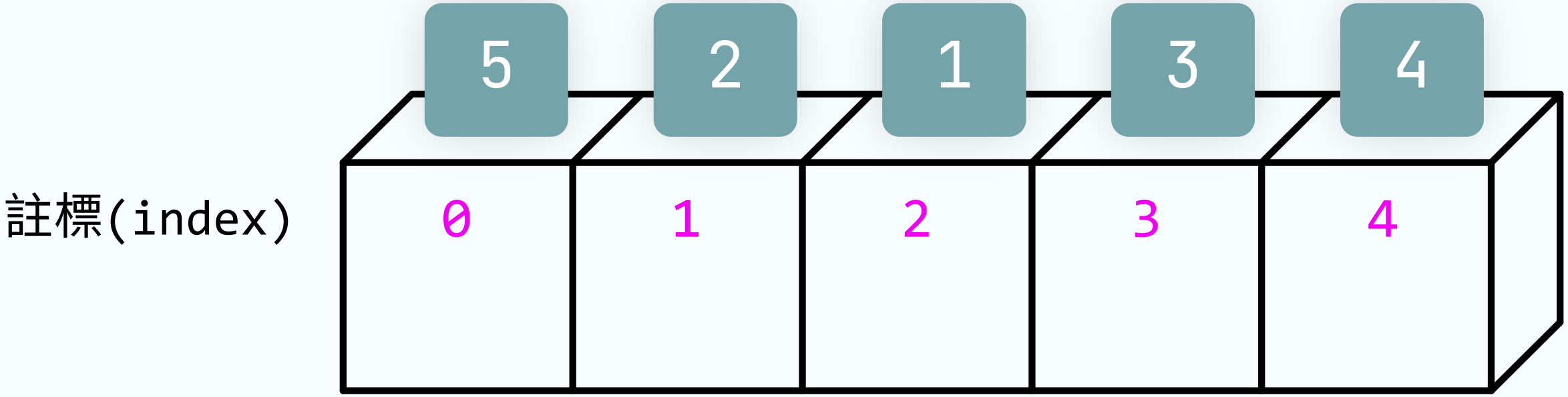
選擇排序法

精神：

- 找最小值(或最大值，如果要遞減排序)
- 丟到「未排序好的數字」後面

選擇排序法

假設陣列中有5個數，
要從小到大排列

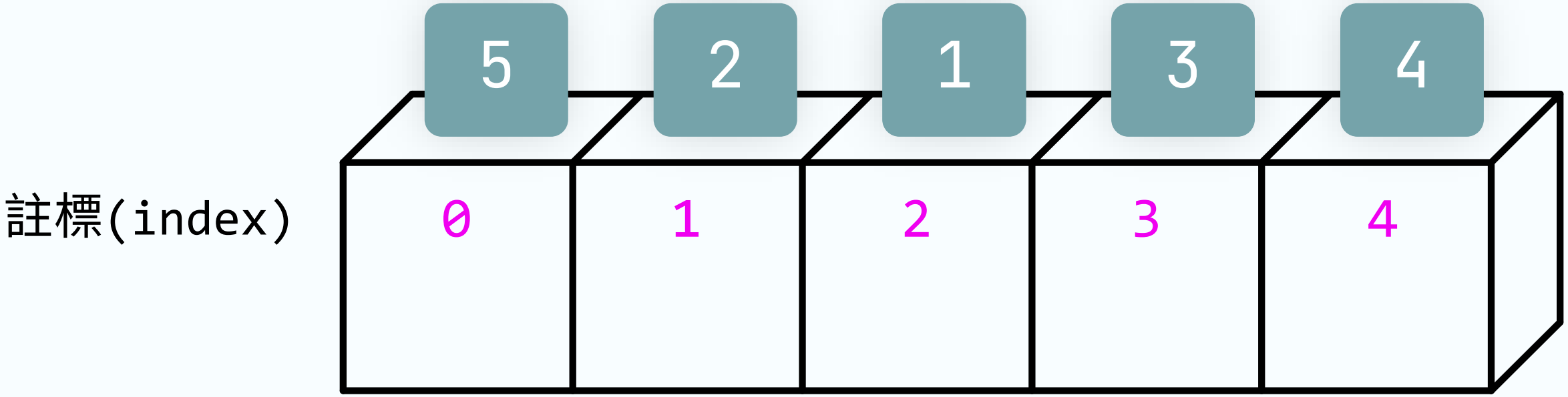


選擇排序法

先假設陣列裡第一個數為最小值

Min: 5, index: 0

未排序的第一個數: 0

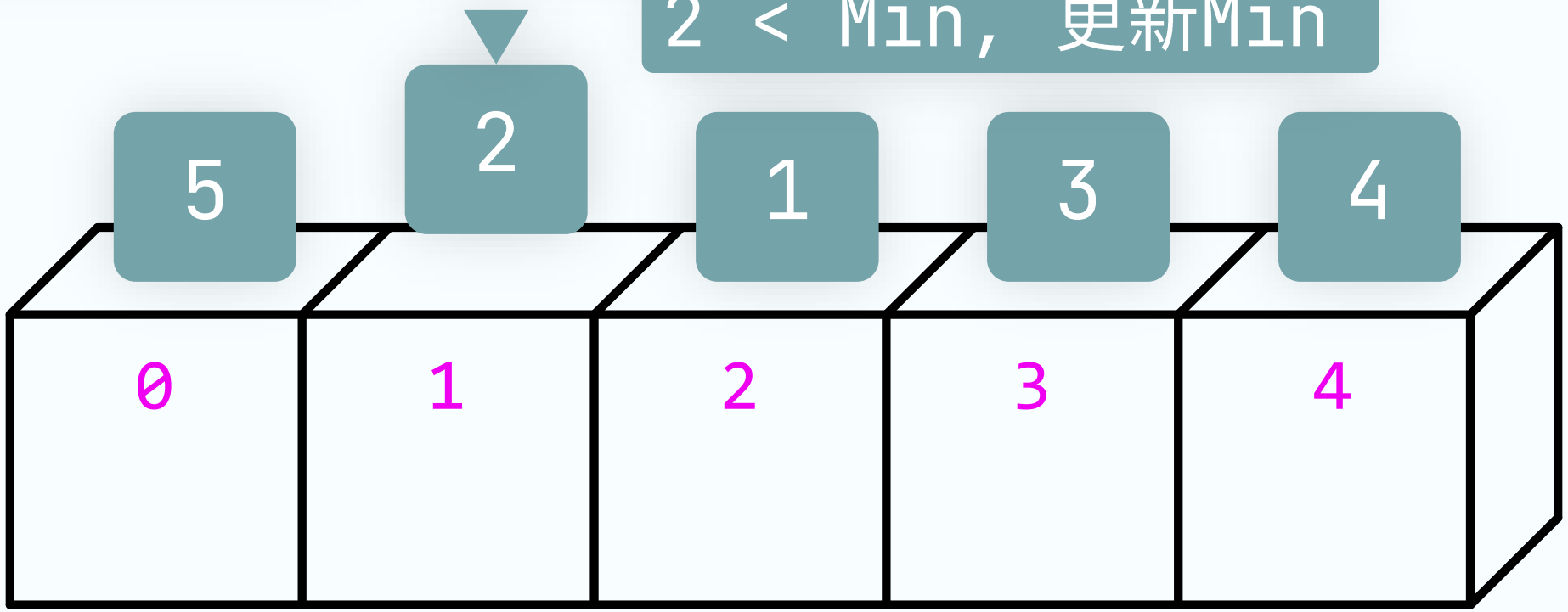


選擇排序法

Min: 5, index: 0

未排序的第一個數: 0

2 < Min, 更新Min

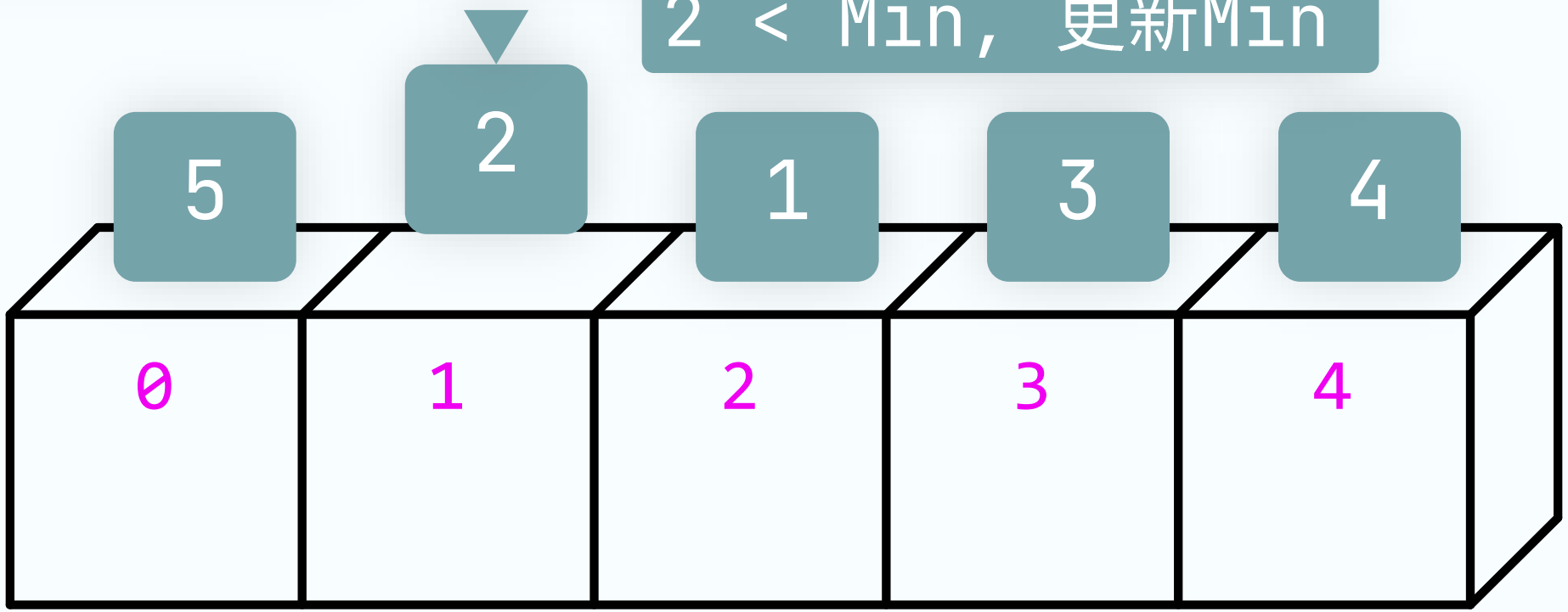


選擇排序法

Min: 2, index: 1

未排序的第一個數: 0

2 < Min, 更新Min

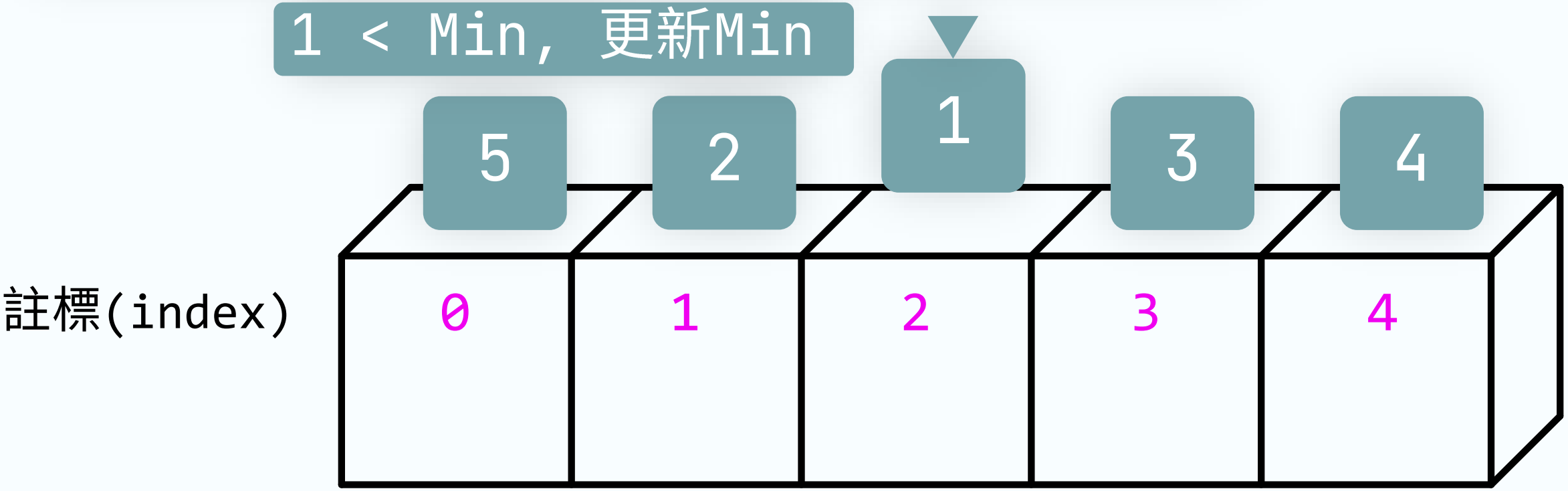


選擇排序法

Min: 2, index: 1

未排序的第一個數: 0

1 < Min, 更新Min

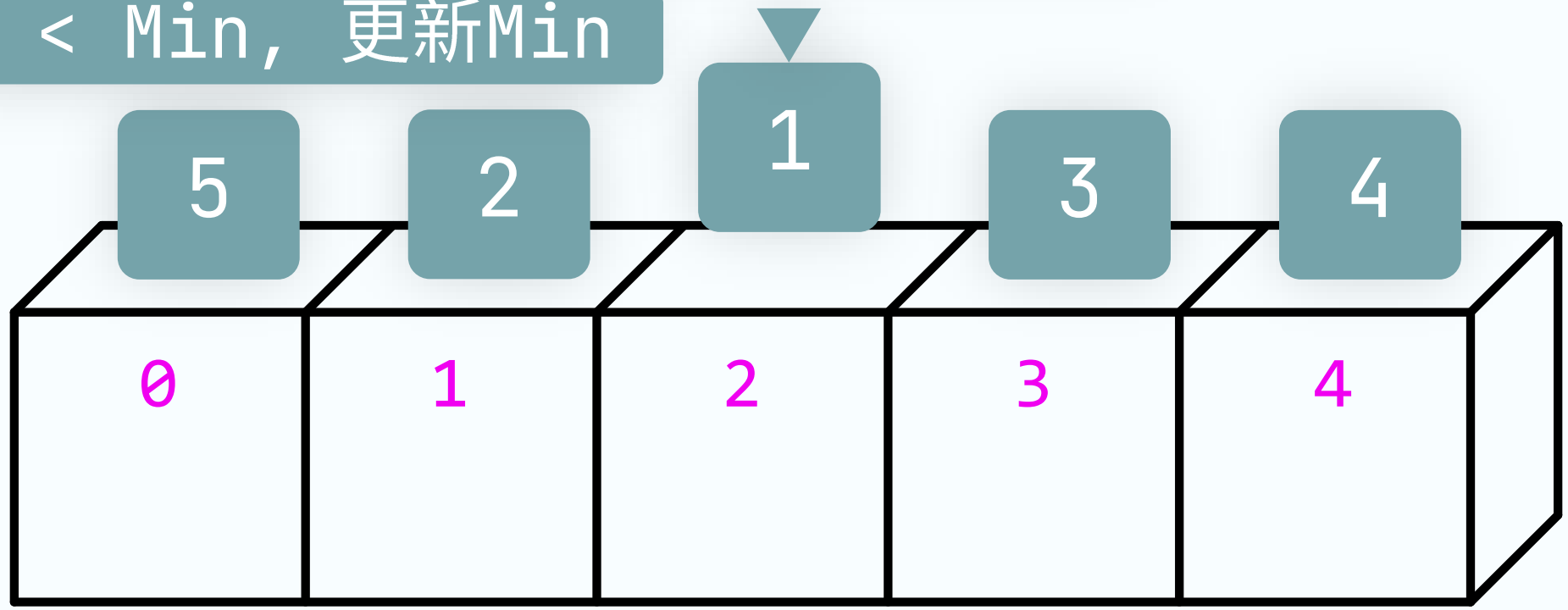


選擇排序法

Min: 1, index: 2

未排序的第一個數: 0

1 < Min, 更新Min

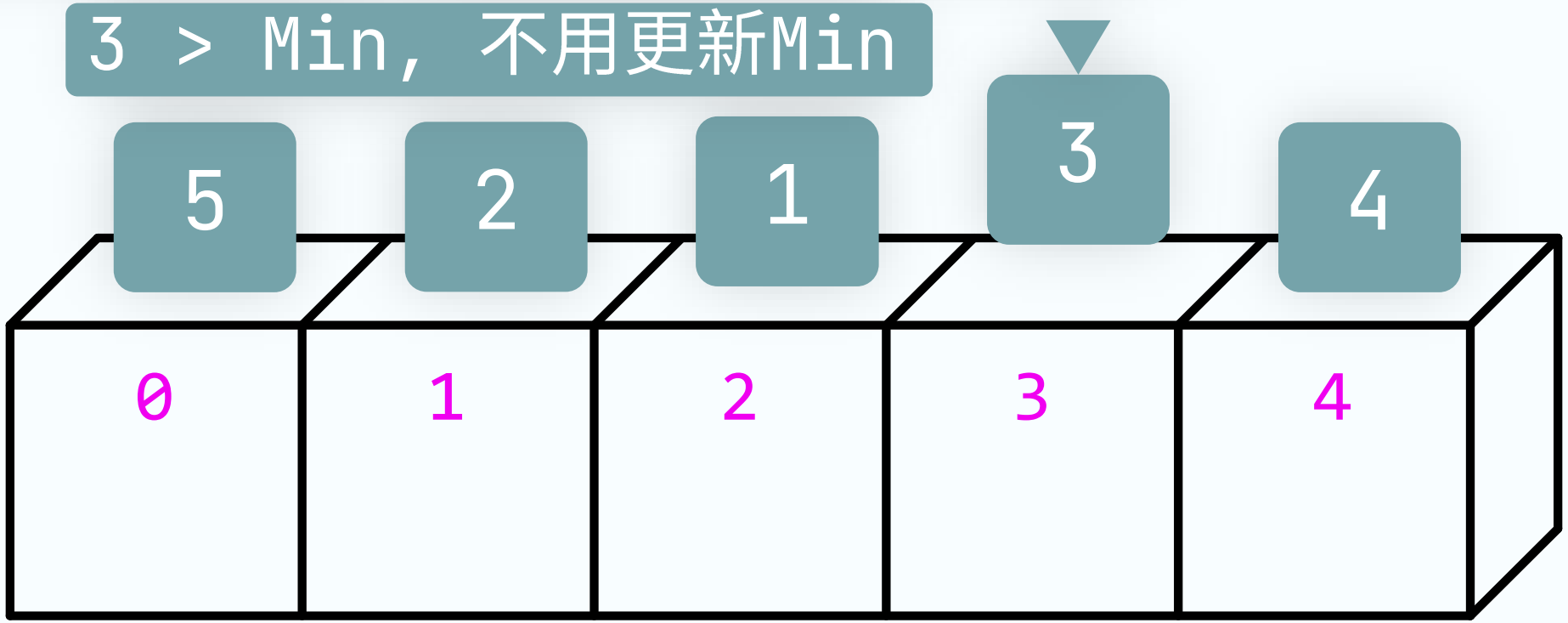


選擇排序法

Min: 1, index: 2

未排序的第一個數: 0

3 > Min, 不用更新Min

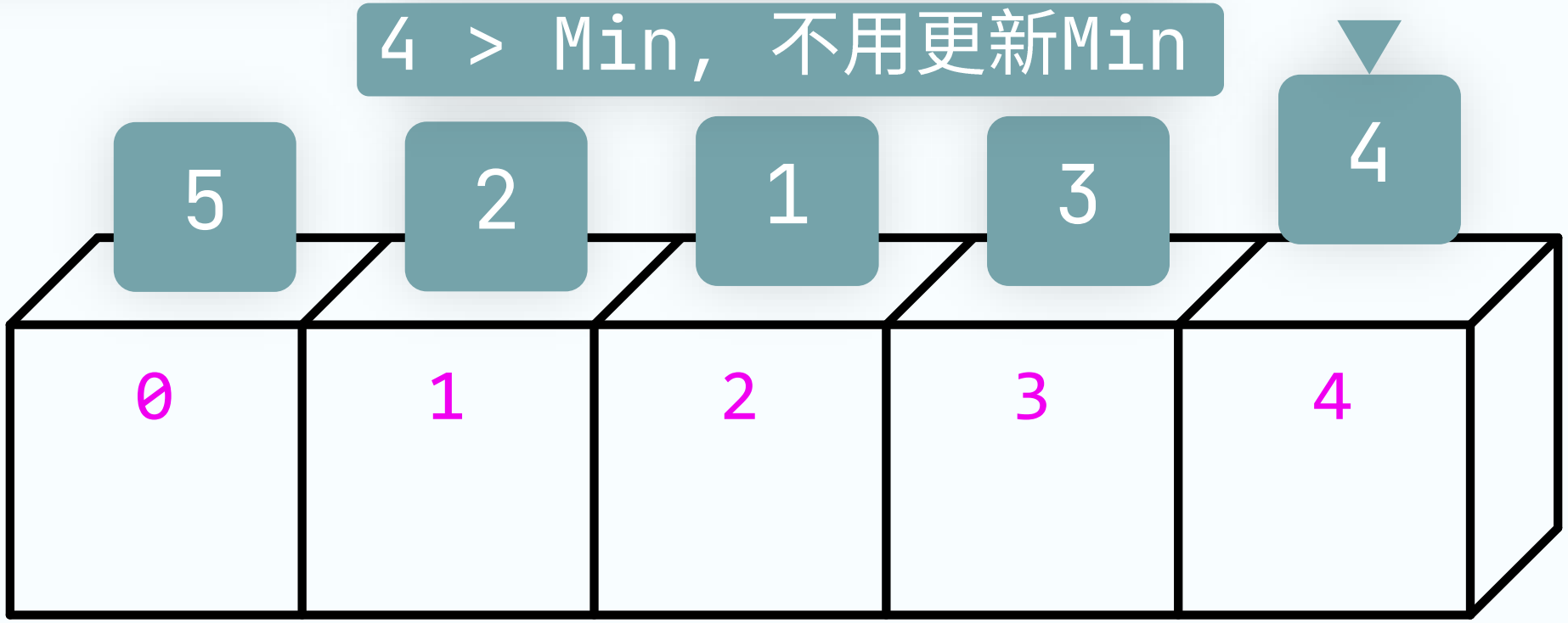


選擇排序法

Min: 1, index: 2

未排序的第一個數: 0

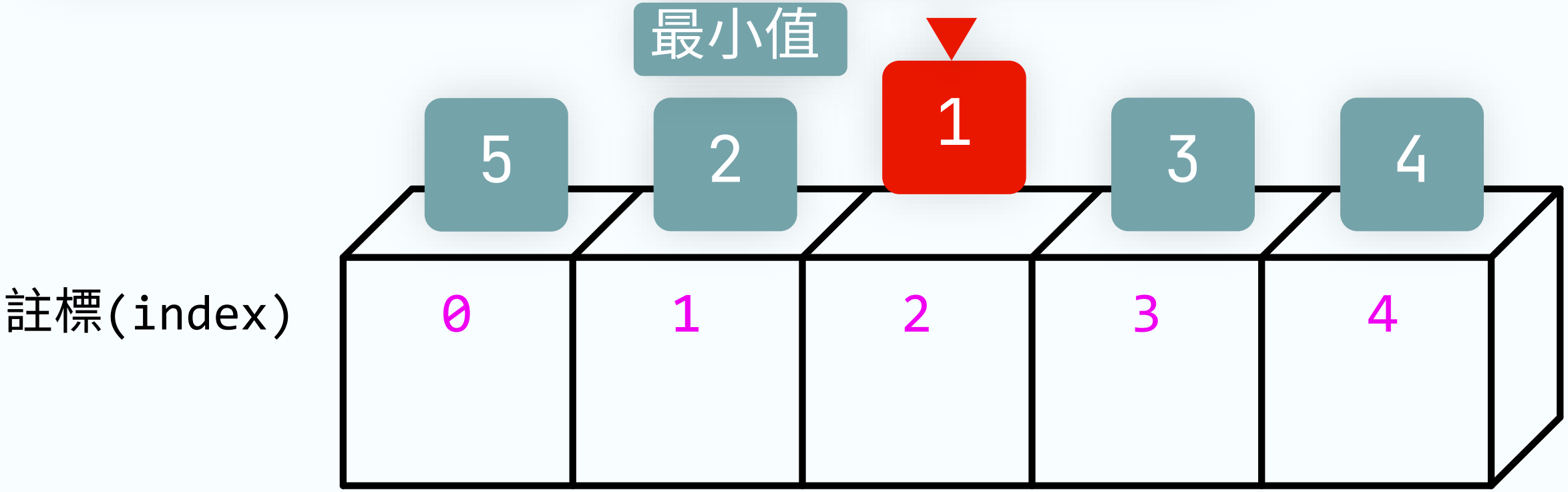
4 > Min, 不用更新Min



選擇排序法

Min: 1, index: 2

未排序的第一個數: 0



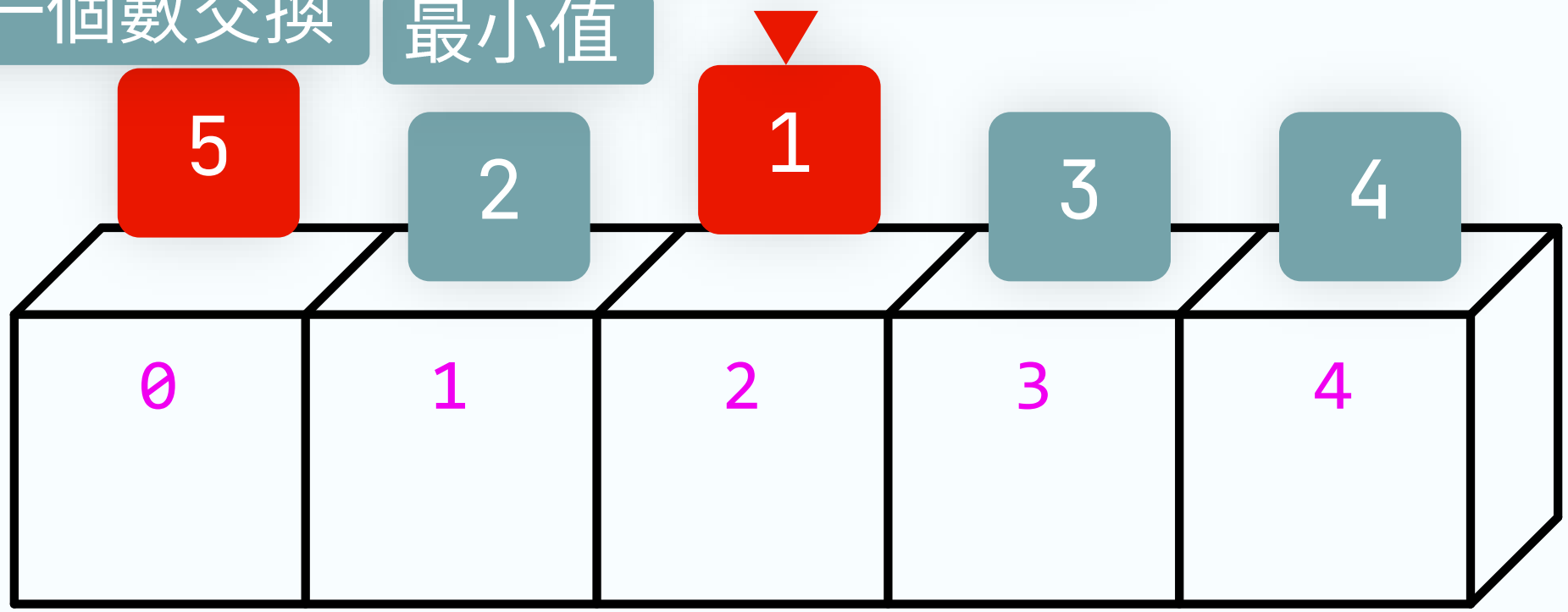
選擇排序法

Min: 1, index: 2

未排序的第一個數: 0

和未排序的第一個數交換

最小值



註標(index)

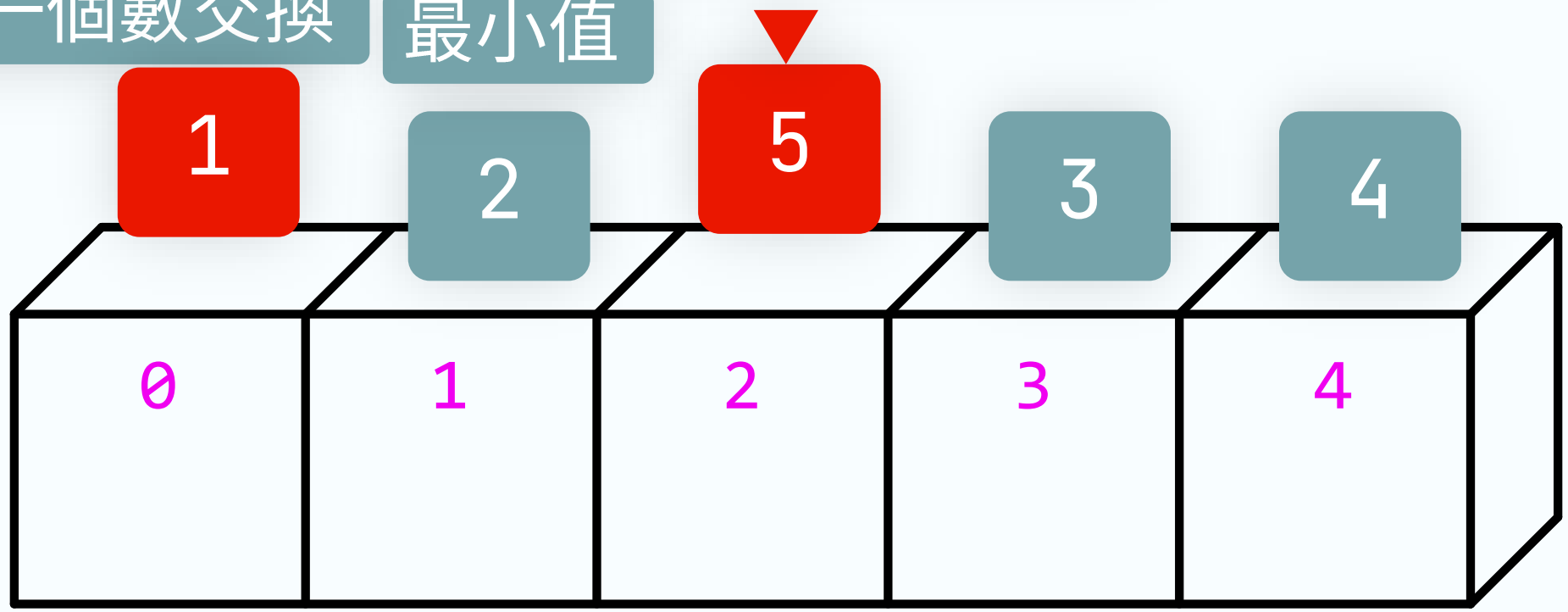
選擇排序法

Min: 1, index: 2

未排序的第一個數: 0

和未排序的第一個數交換

最小值

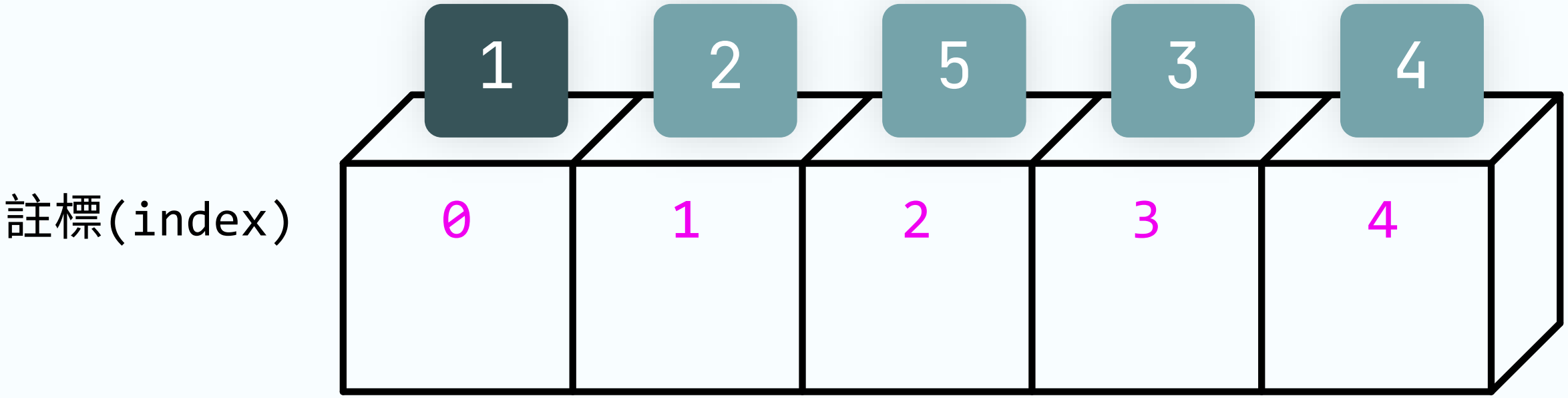


選擇排序法

先假設未排序的第一個數為最小值

Min: 2, index: 1

未排序的第一個數: 1

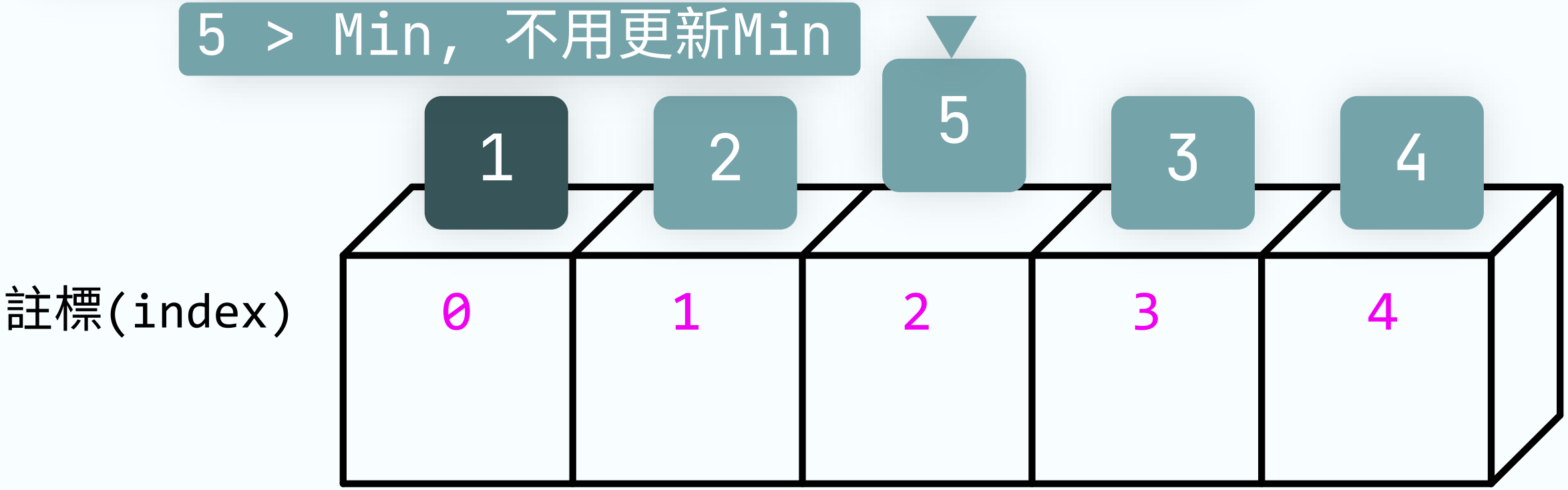


選擇排序法

Min: 2, index: 1

未排序的第一個數: 1

5 > Min, 不用更新Min

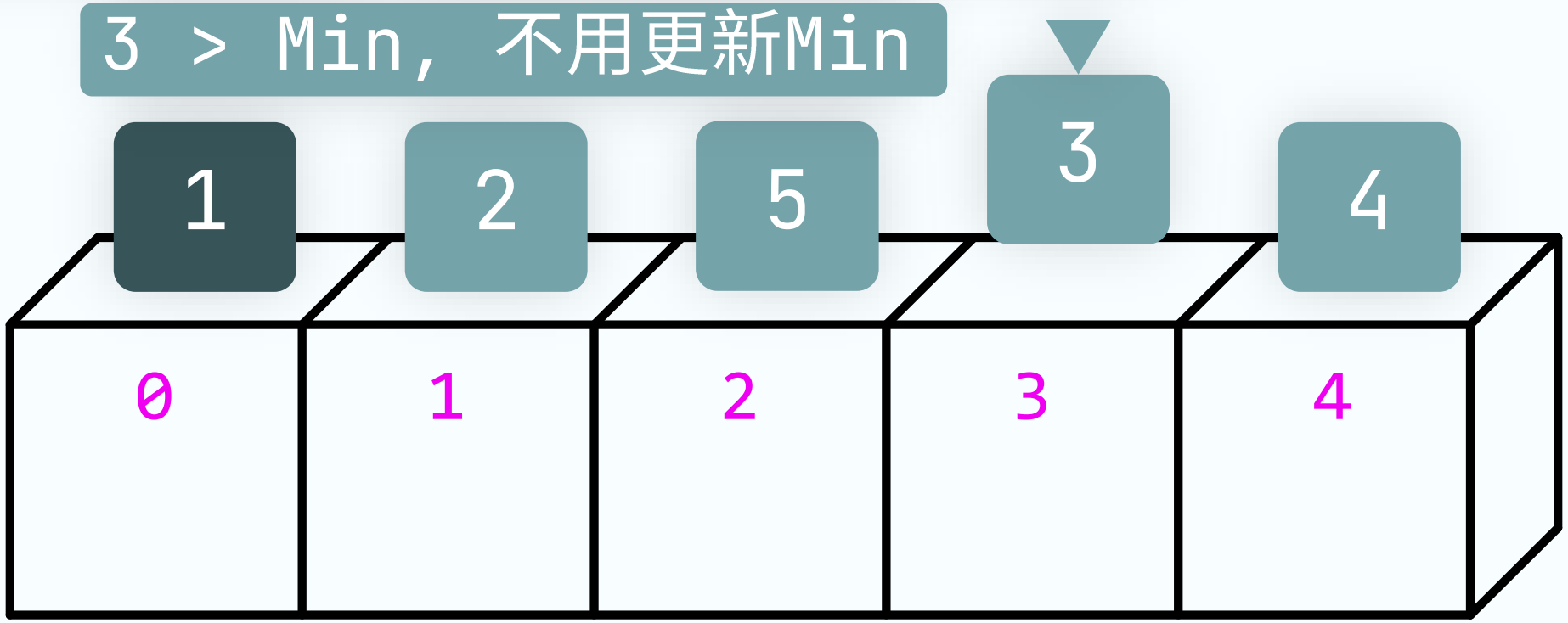


選擇排序法

Min: 2, index: 1

未排序的第一個數: 1

3 > Min, 不用更新Min



選擇排序法

Min: 2, index: 1

未排序的第一個數: 1

4 > Min, 不用更新Min

	1	2	5	3	4
註標(index)	0	1	2	3	4

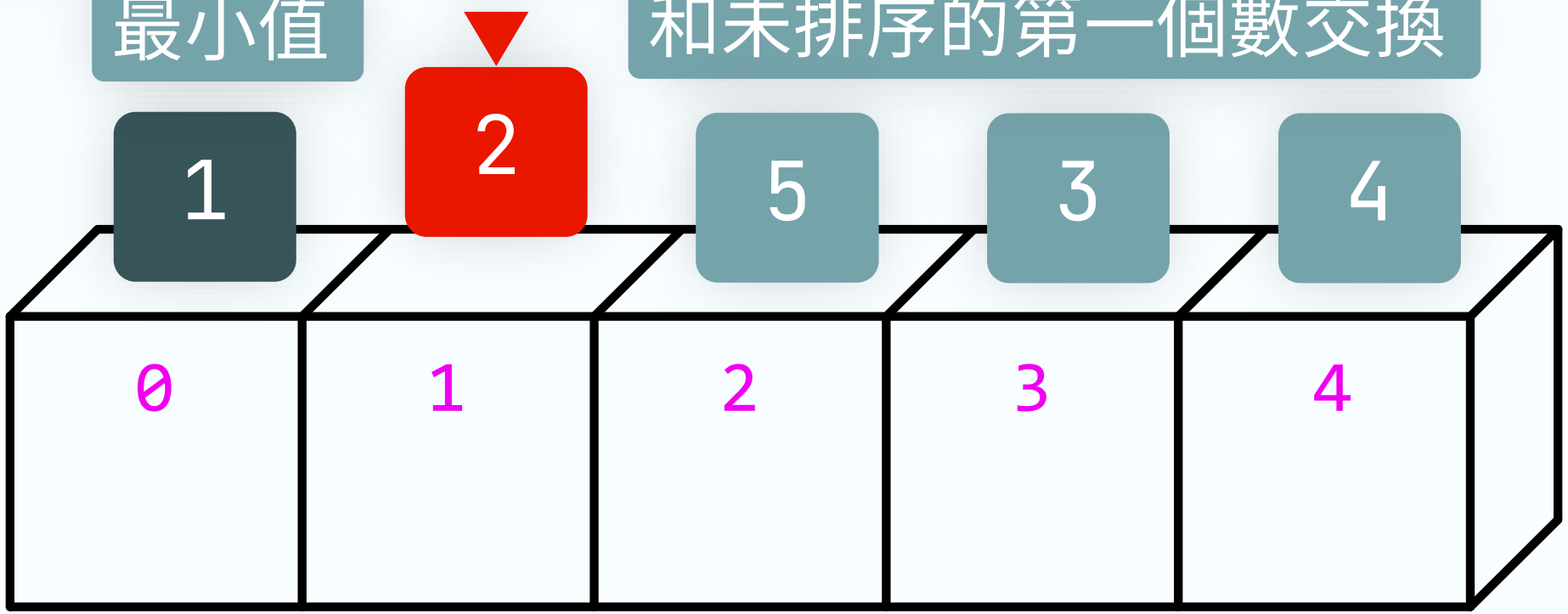
選擇排序法

Min: 2, index: 1

未排序的第一個數: 1

最小值

和未排序的第一個數交換

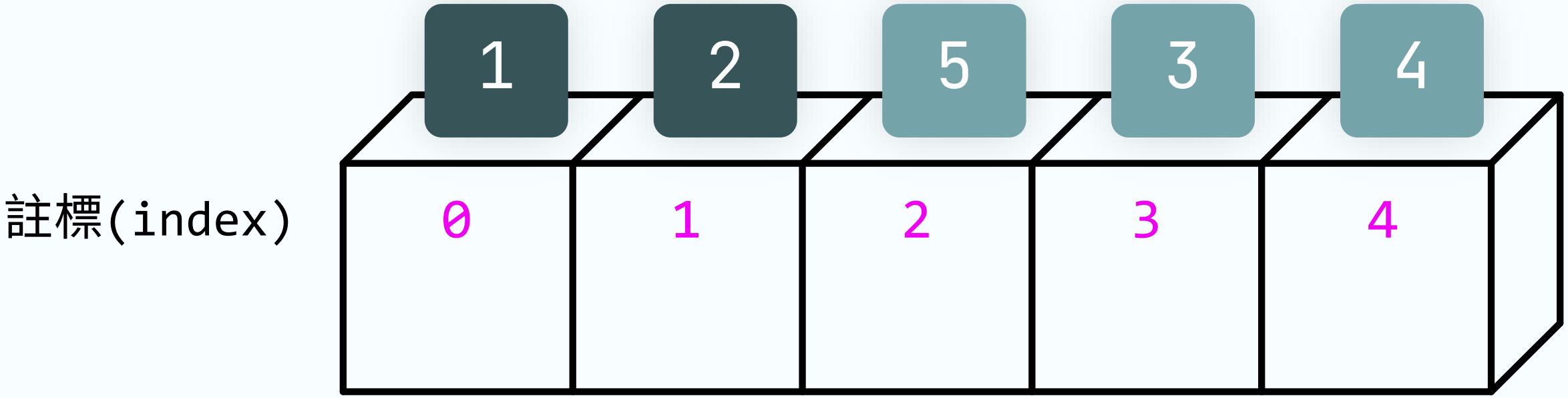


選擇排序法

先假設未排序的第一個數為最小值

Min: 5, index: 2

未排序的第一個數: 2

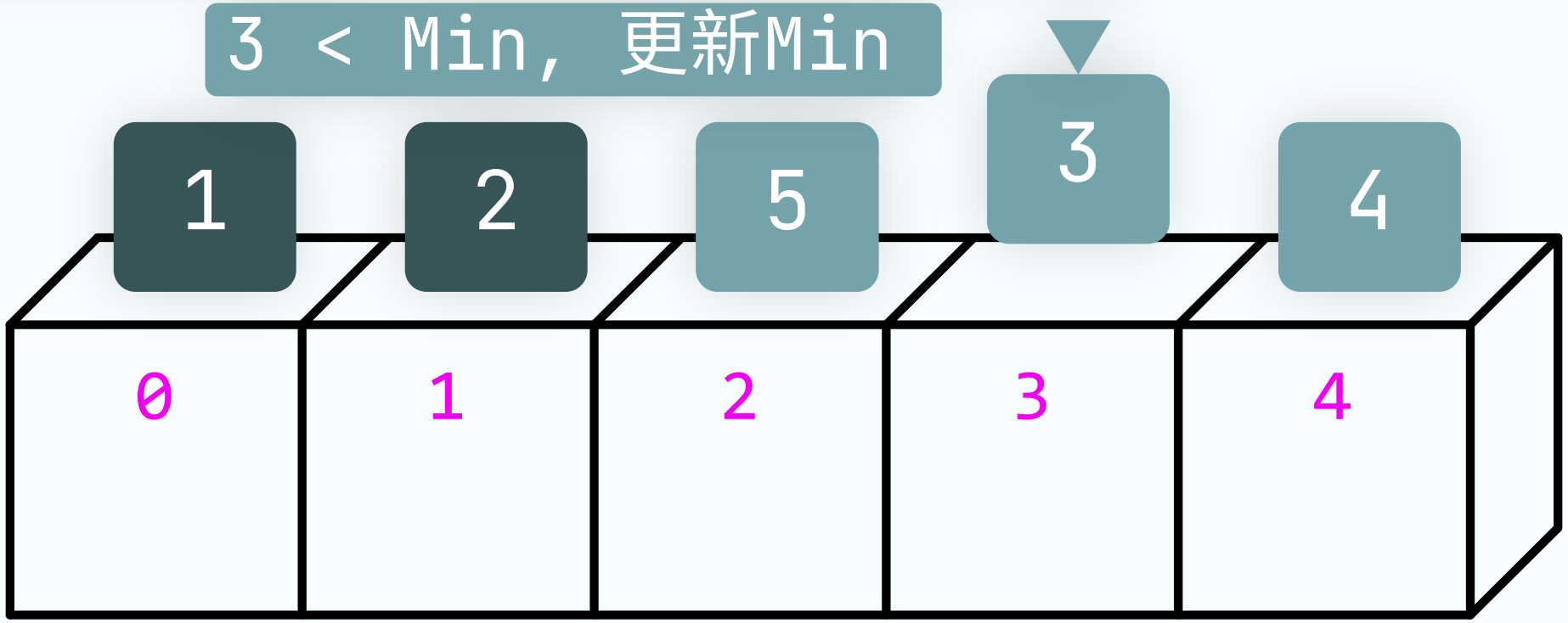


選擇排序法

Min: 5, index: 2

未排序的第一個數: 2

3 < Min, 更新Min

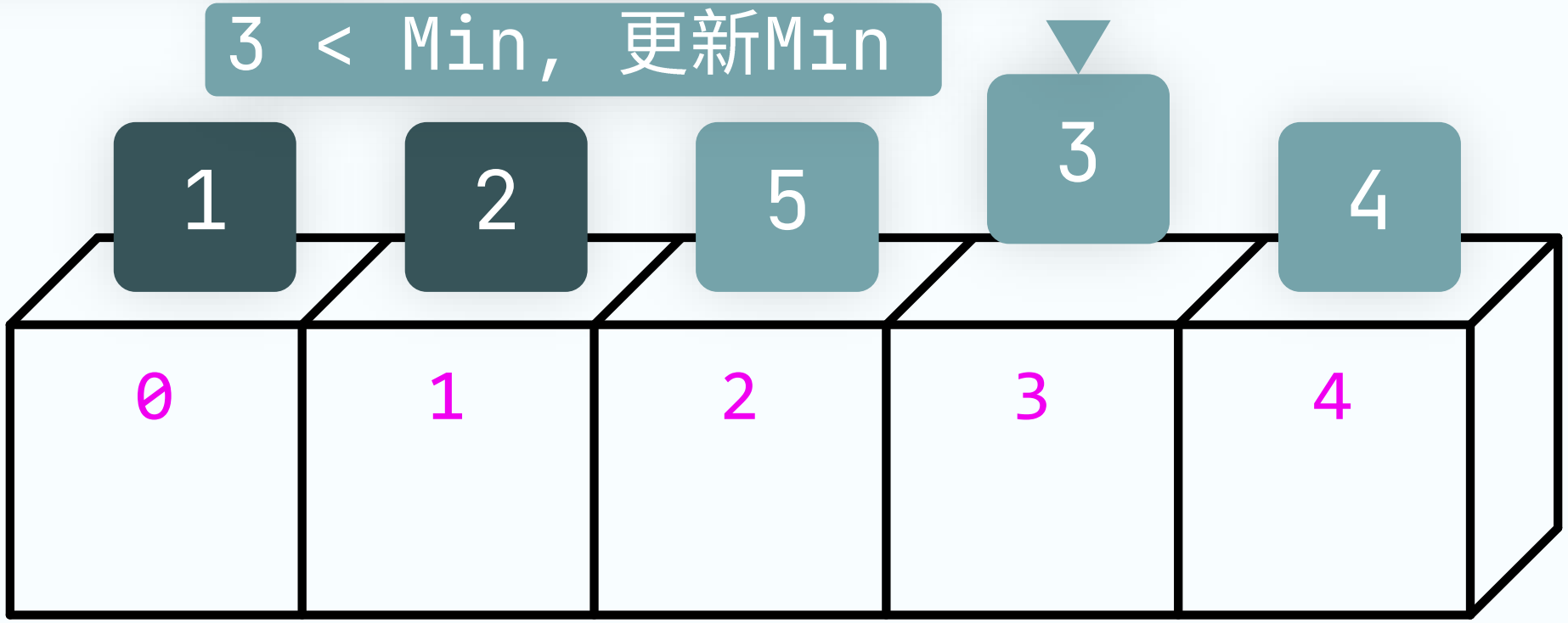


選擇排序法

Min: 3, index: 3

未排序的第一個數: 2

3 < Min, 更新Min

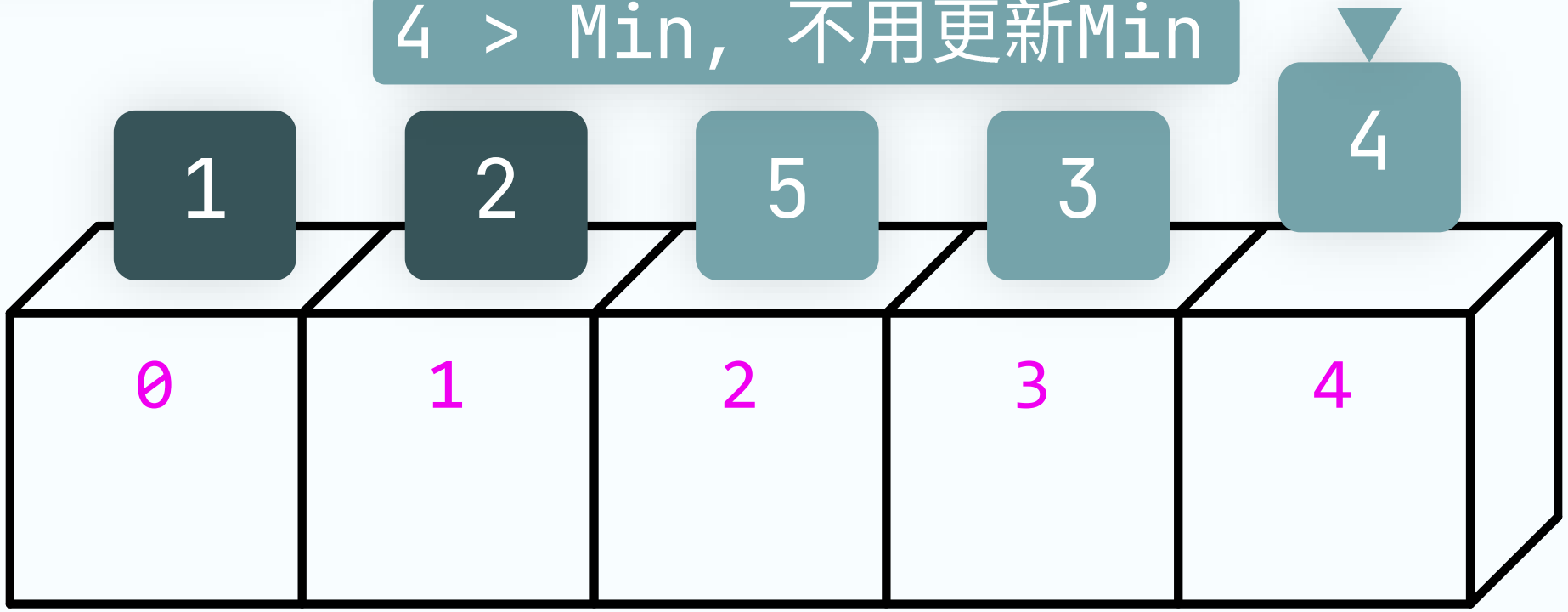


選擇排序法

Min: 3, index: 3

未排序的第一個數: 2

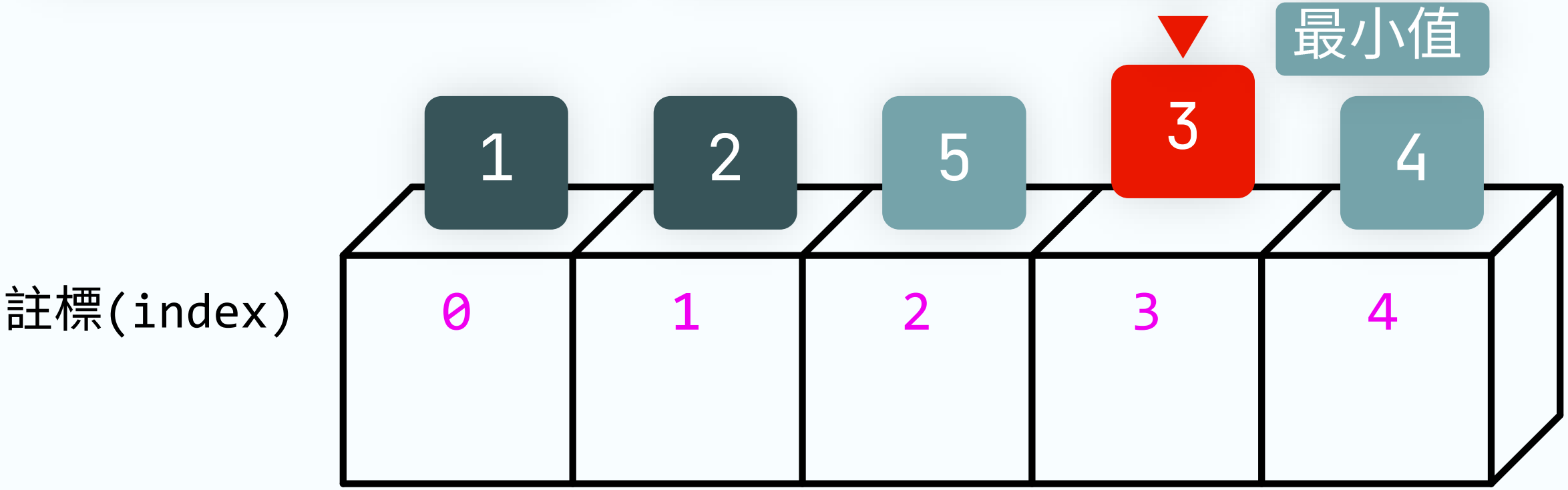
4 > Min, 不用更新Min



選擇排序法

Min: 3, index: 3

未排序的第一個數: 2



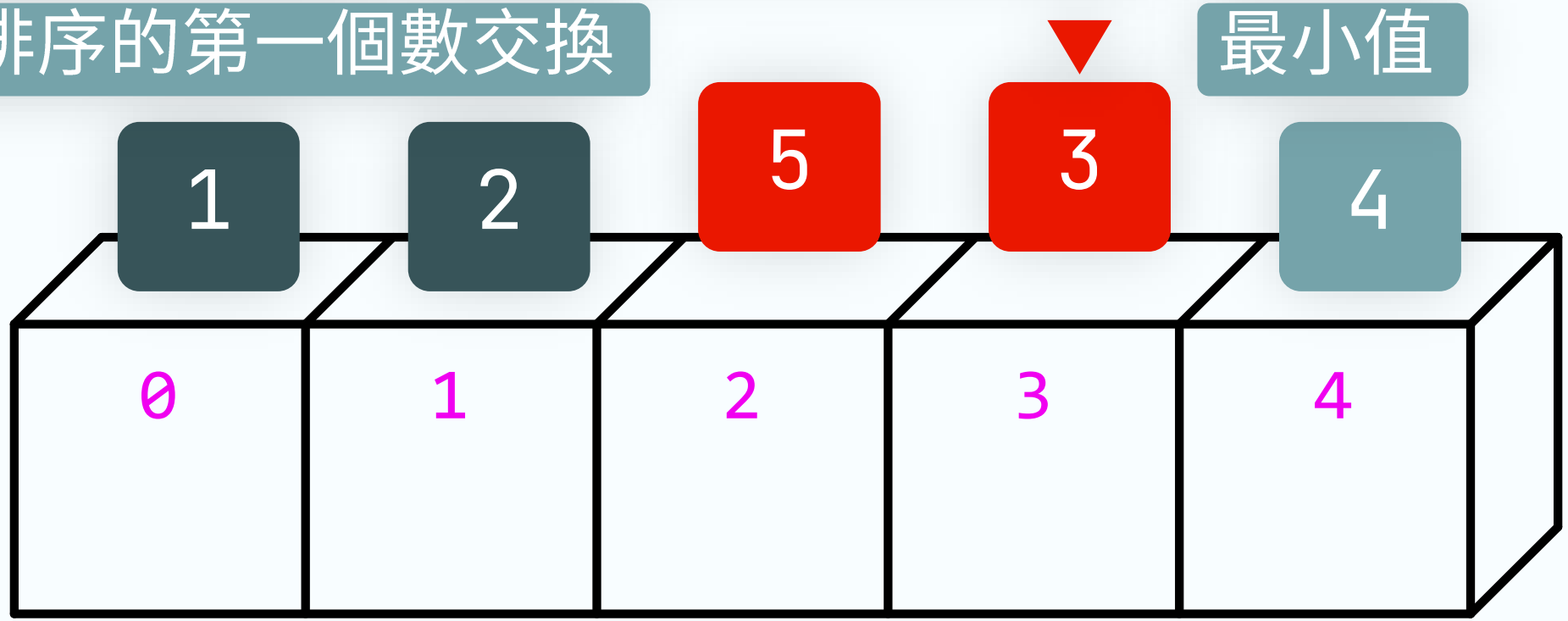
選擇排序法

Min: 3, index: 3

未排序的第一個數: 2

和未排序的第一個數交換

最小值



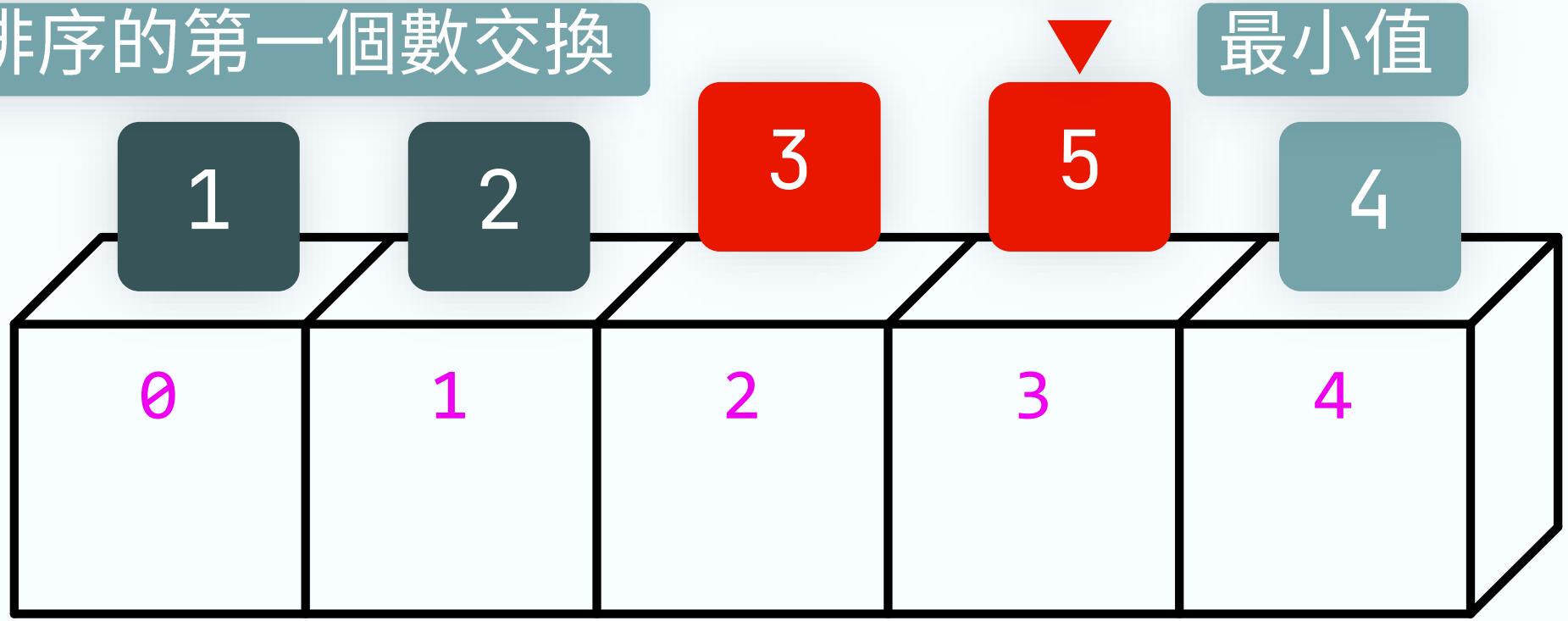
選擇排序法

Min: 3, index: 3

未排序的第一個數: 2

和未排序的第一個數交換

最小值

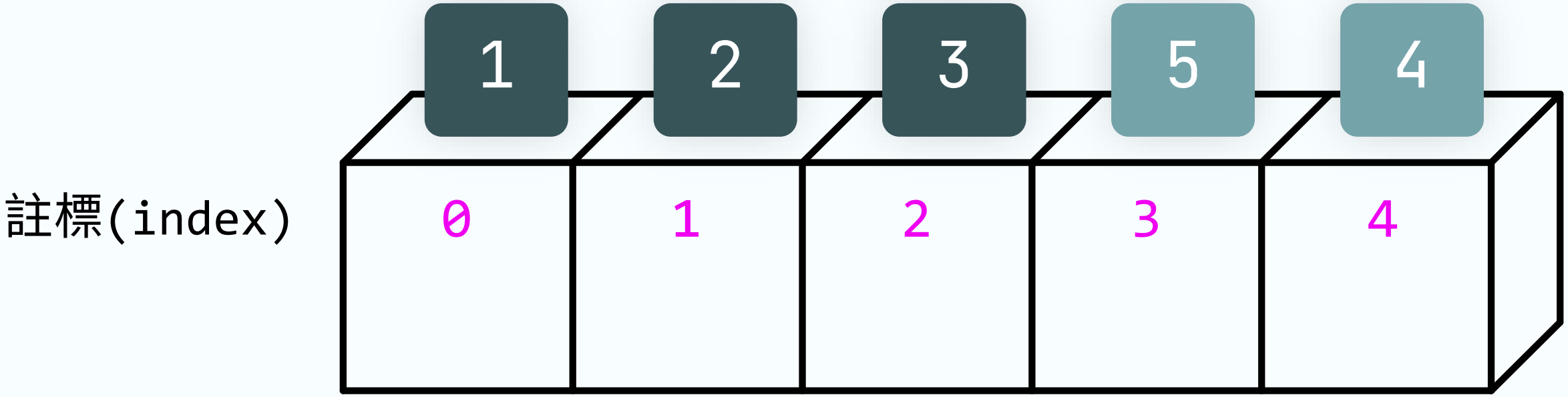


選擇排序法

先假設未排序的第一個數為最小值

Min: 5, index: 3

未排序的第一個數: 3

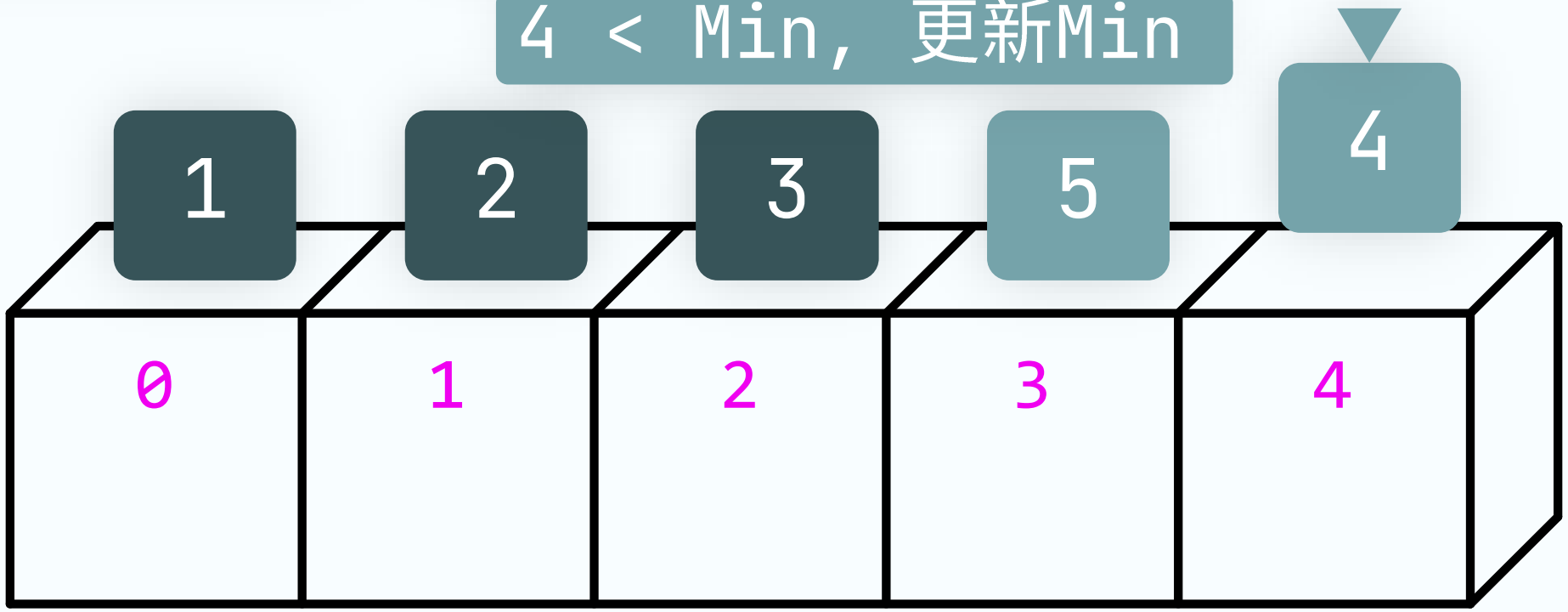


選擇排序法

Min: 5, index: 3

未排序的第一個數: 3

4 < Min, 更新Min



選擇排序法

Min: 4, index: 4

未排序的第一個數: 3

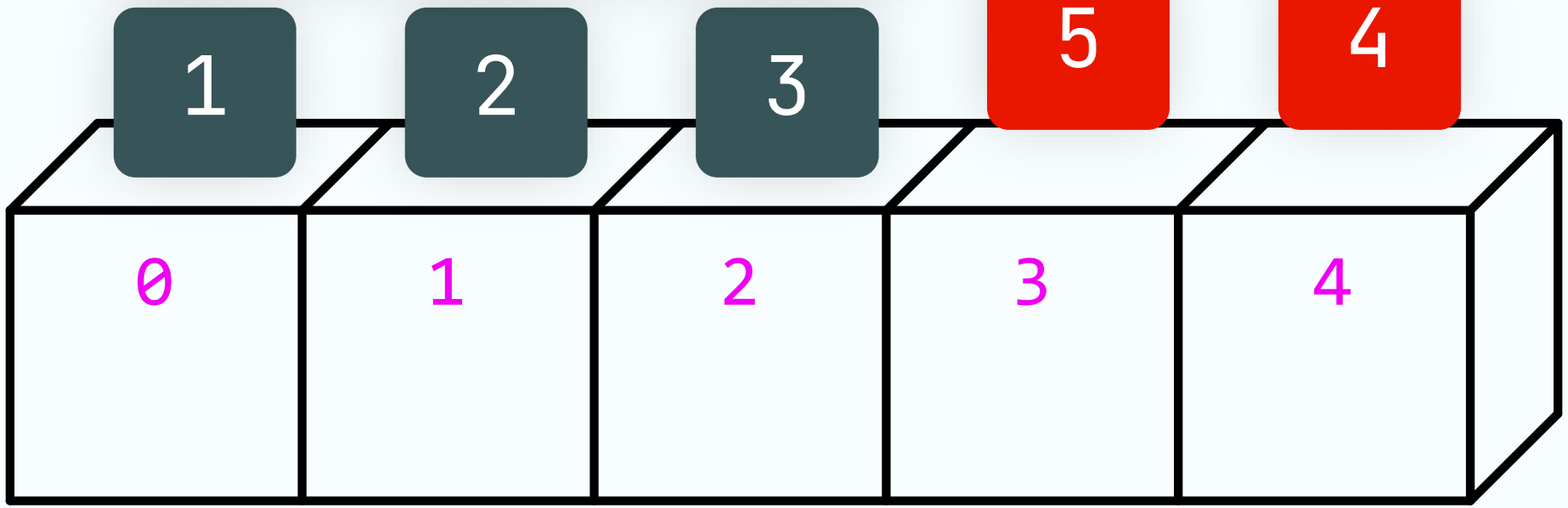
選擇排序法

Min: 4, index: 4

未排序的第一個數: 3

最小值

和未排序的第一個數交換



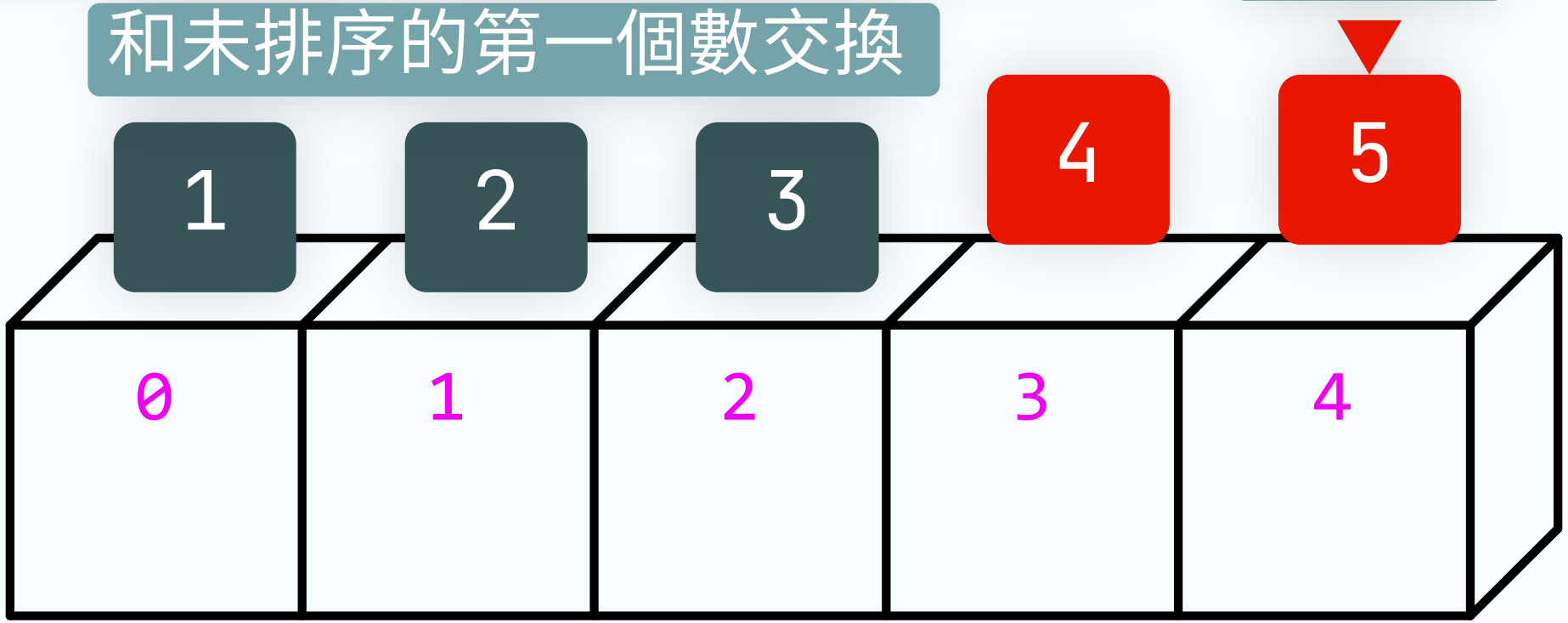
選擇排序法

Min: 4, index: 4

未排序的第一個數: 3

和未排序的第一個數交換

最小值

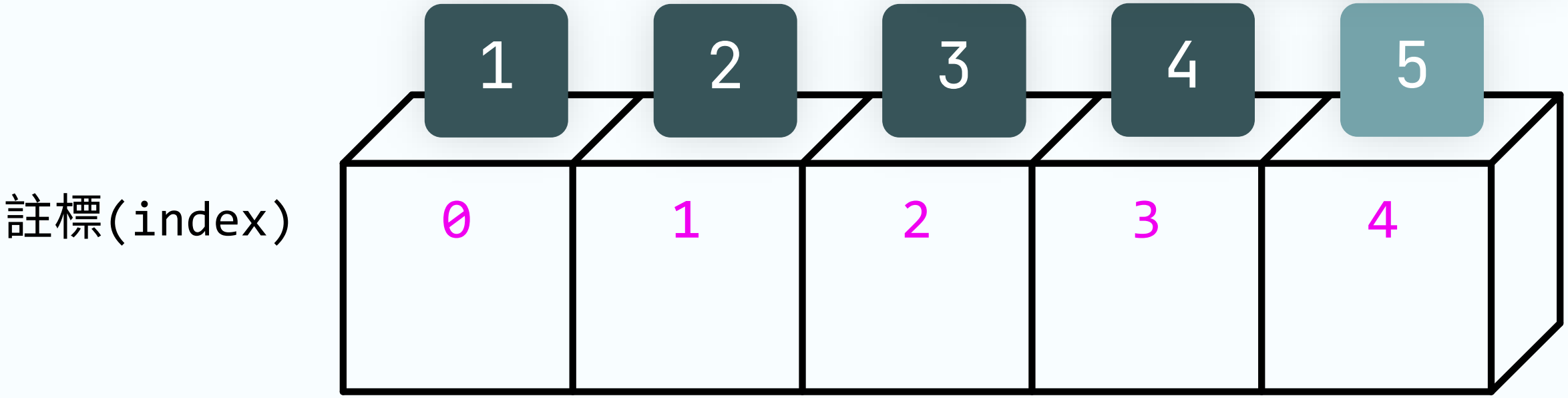


選擇排序法

Min: 4, index: 4

未排序的第一個數: 4

最後一個數也一起排好了

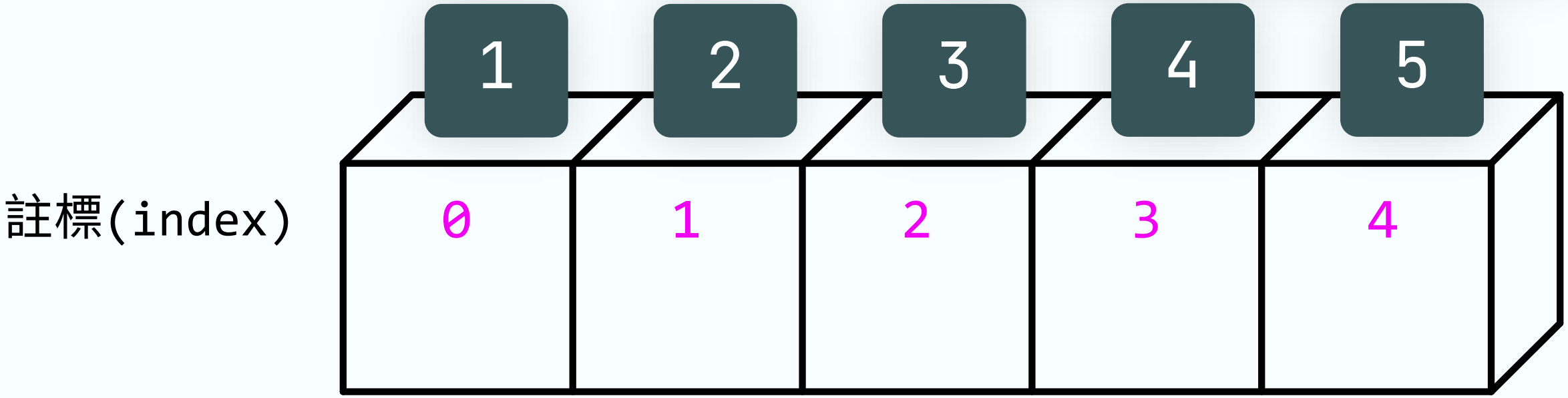


選擇排序法

Min: 4, index: 4

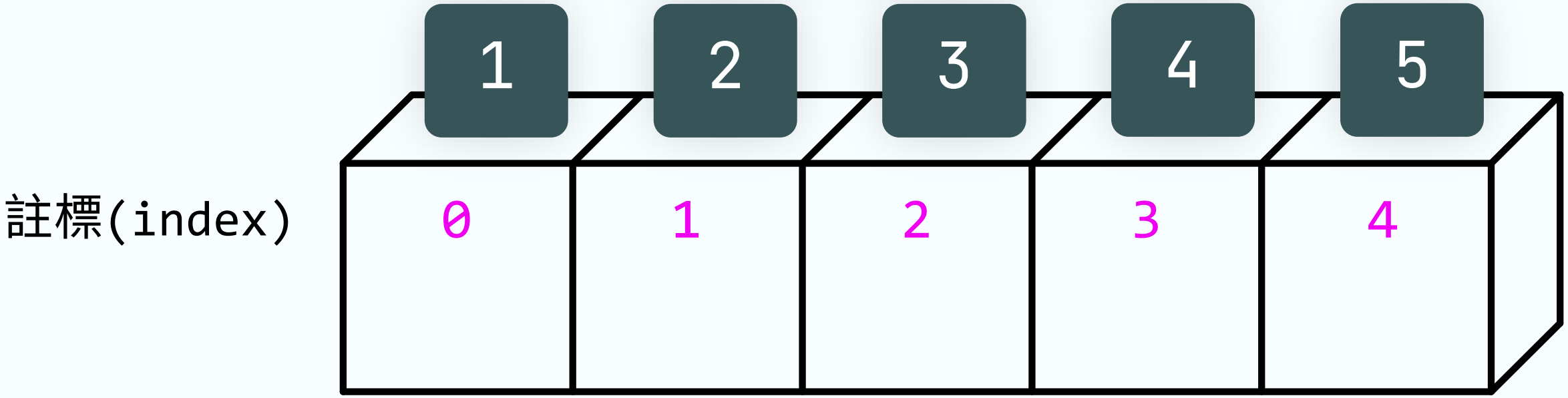
未排序的第一個數: 4

最後一個數也一起排好了



選擇排序法

如此一來，就排好資料們了



選擇排序法

```cpp
#include <iostream>
using namespace std;

int main()
{
    int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;

    for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
        min_index = to_sort;
        for (int check = to_sort + 1; check < arr_length; check++){
            if (arr[check] < arr[min_index]){
                min_index = check;
            }
        }

        tmp = arr[to_sort];
        arr[to_sort] = arr[min_index];
        arr[min_index] = tmp;

    }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }

    cout << endl;

    return 0;
}
```

選擇排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8      for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9          min_index = to_sort;
10         for (int check = to_sort + 1; check < arr_length; check++){
11             if (arr[check] < arr[min_index]){
12                 min_index = check;
13             }
14         }
15
16         tmp = arr[to_sort];
17         arr[to_sort] = arr[min_index];
18         arr[min_index] = tmp;
19
20     }
21
22     for (int i = 0; i < arr_length; i++){
23         cout << arr[i] << " ";
24     }
25     cout << endl;
26
27     return 0;
28 }
```

找最小值

選擇排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8      for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9          min_index = to_sort;
10         for (int check = to_sort + 1; check < arr_length; check++){
11             if (arr[check] < arr[min_index]){
12                 min_index = check;
13             }
14         }
15
16         tmp = arr[to_sort];
17         arr[to_sort] = arr[min_index];
18         arr[min_index] = tmp;
19
20     }
21
22     for (int i = 0; i < arr_length; i++){
23         cout << arr[i] << " ";
24     }
25     cout << endl;
26
27     return 0;
28 }
```

找最小值

交換

選擇排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8      for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9          min_index = to_sort;
10         for (int check = to_sort + 1; check < arr_length; check++){
11             if (arr[check] < arr[min_index]){
12                 min_index = check;
13             }
14         }
15
16         tmp = arr[to_sort];
17         arr[to_sort] = arr[min_index];
18         arr[min_index] = tmp;
19
20     }
21
22     for (int i = 0; i < arr_length; i++){
23         cout << arr[i] << " ";
24     }
25     cout << endl;
26
27     return 0;
28 }
```

找最小值

交換

輸出

選擇排序法

```
int main()
{
    int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;

    for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
        min_index = to_sort;
        for (int check = to_sort + 1; check < arr_length; check++){
            if (arr[check] < arr[min_index]){
                min_index = check;
            }
        }

        tmp = arr[to_sort];
```

找最小值

選擇排序法

宣告長度5的陣列arr

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

選擇排序法

宣告長度5的陣列arr

arr_length
記錄陣列arr長度

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

選擇排序法

宣告長度5的陣列arr

arr_length
記錄陣列arr長度

min_index記錄最小值的index

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

選擇排序法

宣告長度5的陣列arr

arr_length
記錄陣列arr長度

tmp交換
暫存用

min_index記錄最小值的index

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

選擇排序法

利用for迴圈從陣列開頭開始排序，
排到陣列倒數第一個元素(最後一個元素不用檢查)

```
int main()
{
                                                    tmp;

    for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
        min_index = to_sort;
        for (int check = to_sort + 1; check < arr_length; check++){
            if (arr[check] < arr[min_index]){
                min_index = check;
            }
        }

        tmp = arr[to_sort];
```

選擇排序法

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

先假設未排序的第一個元素為最小值

選擇排序法

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (                                                    ){
9           mi
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

利用for迴圈檢查
未排序的第一個元素 後面是否有更小的值

選擇排序法

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

如果檢查到比最小值還小的值

選擇排序法

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

如果檢查到比最小值還小的值

就更新最小值的位置(index)

選擇排序法

```
4   int main()
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
```

如此一來，就可以找到未排序的元素中的最小值

選擇排序法

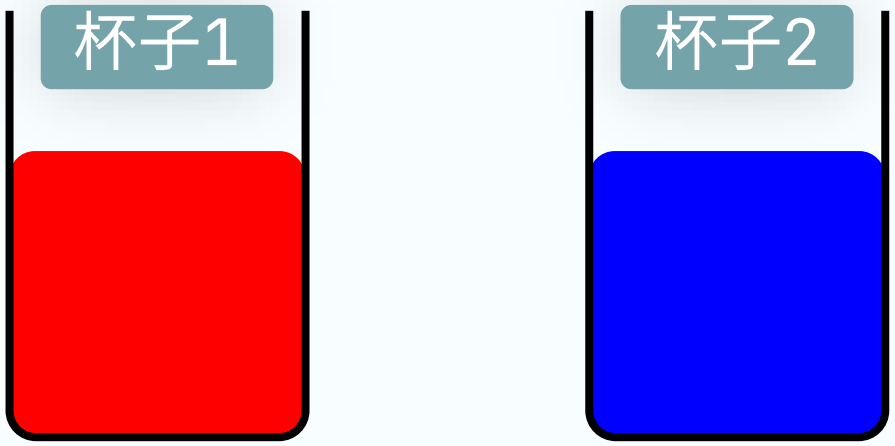
```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

交換

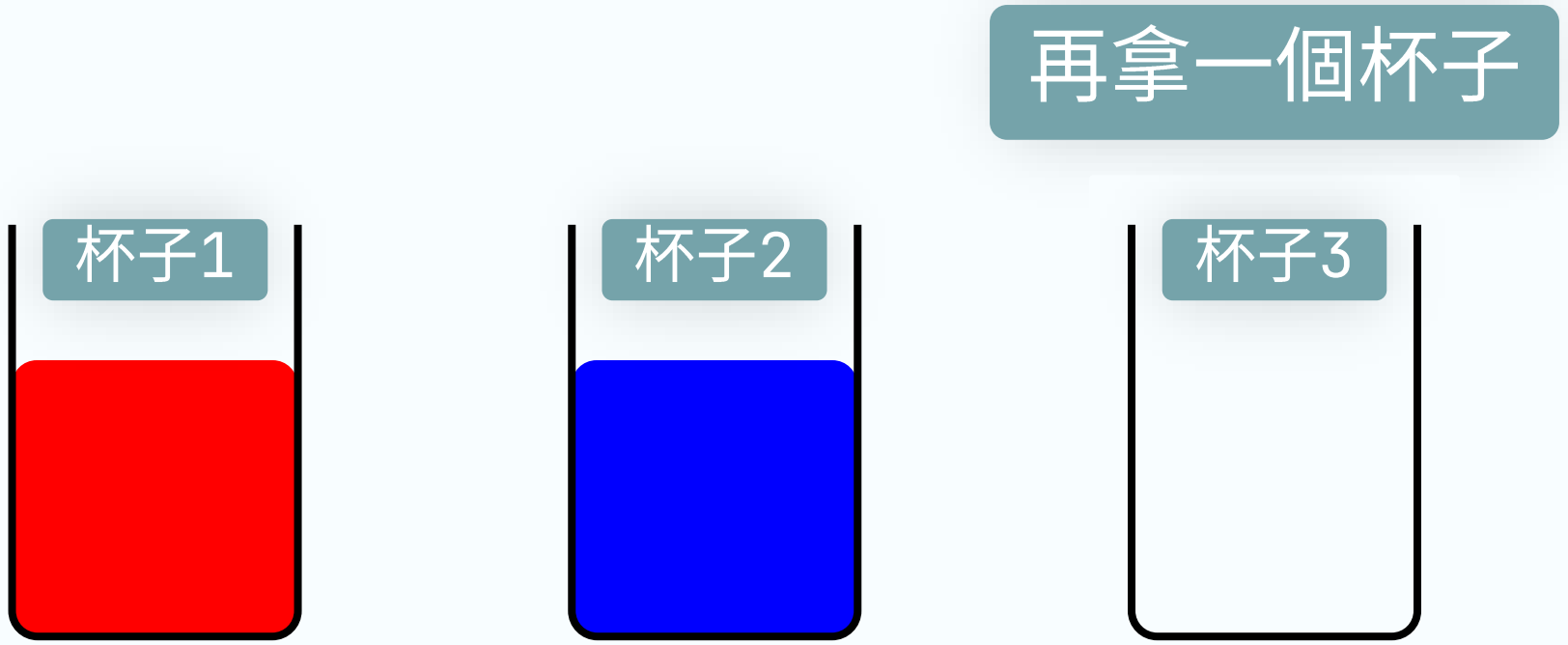
交換的概念

假設有二個杯子，一杯裝著紅色液體，一杯裝著藍色液體

要如何交換二個杯子裡的液體？

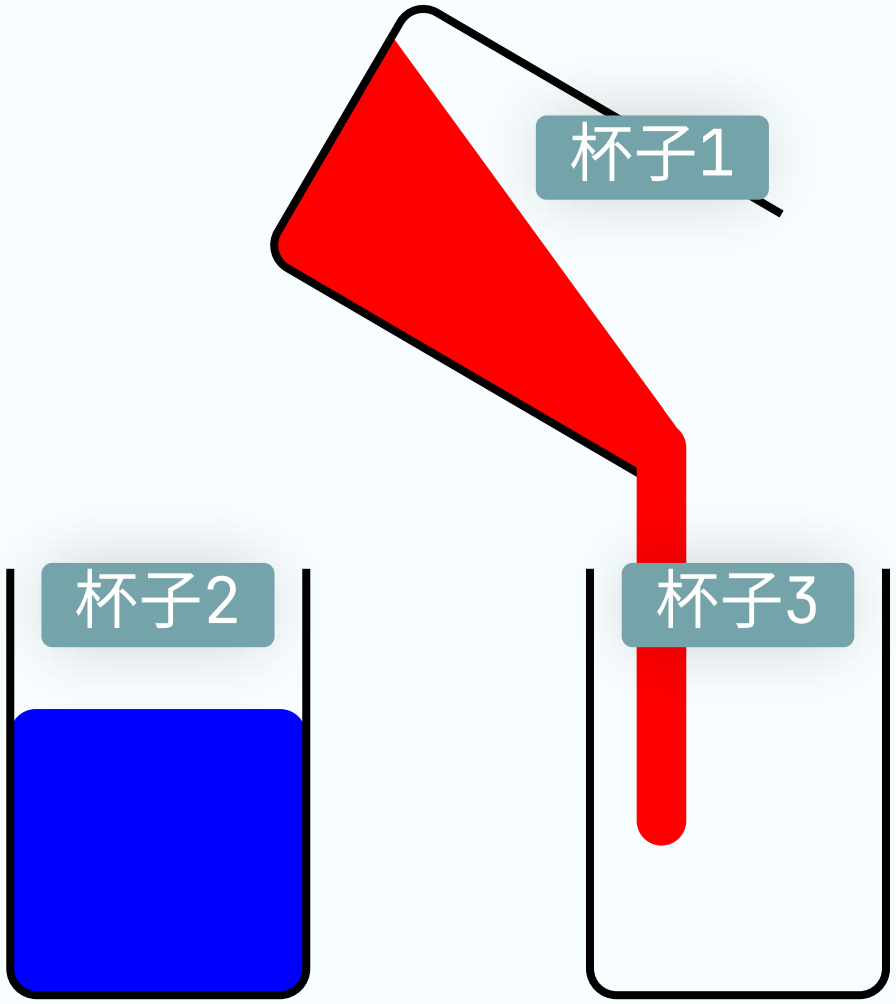


交換的概念



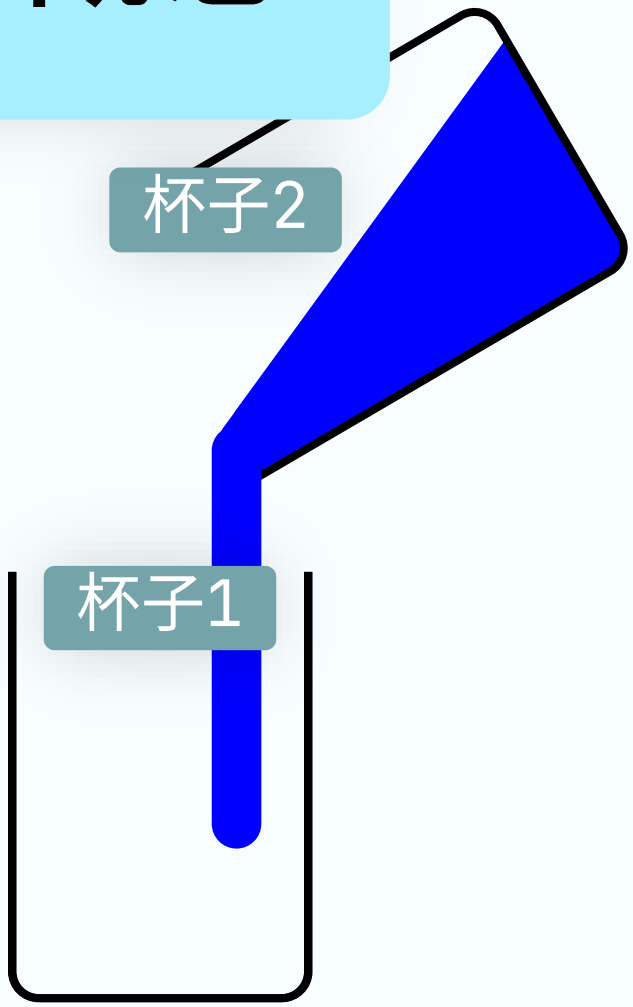
交換的概念

將一種液體倒入新杯子裡

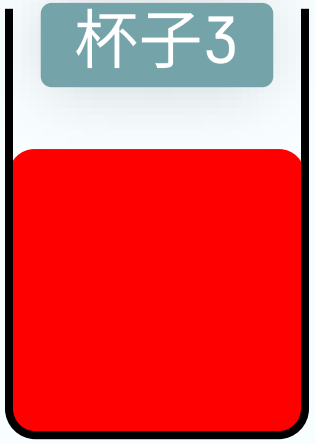


交换的概念

交換的概念



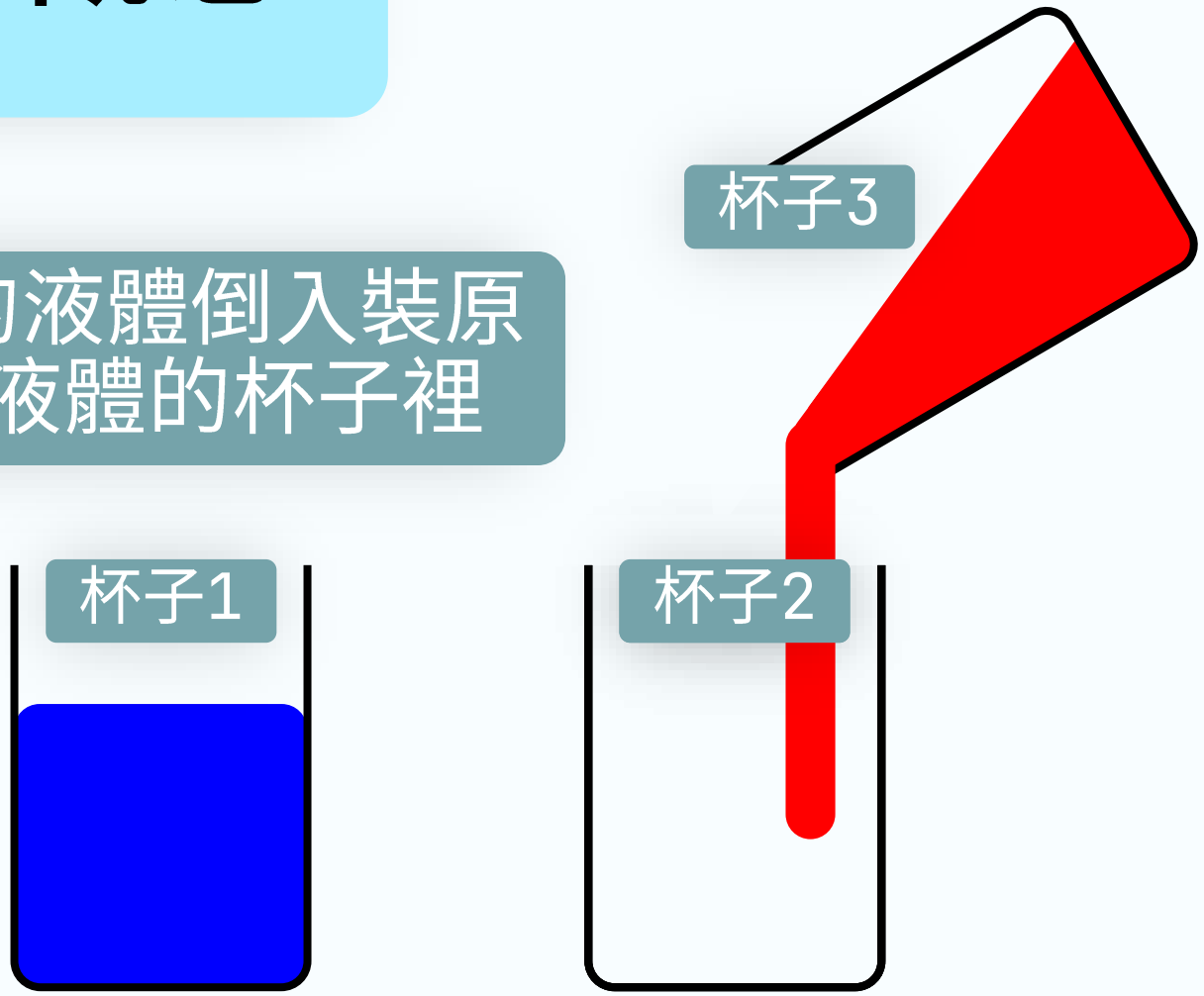
將另一種液體倒入裝
原本液體的杯子裡



交换的概念

交換的概念

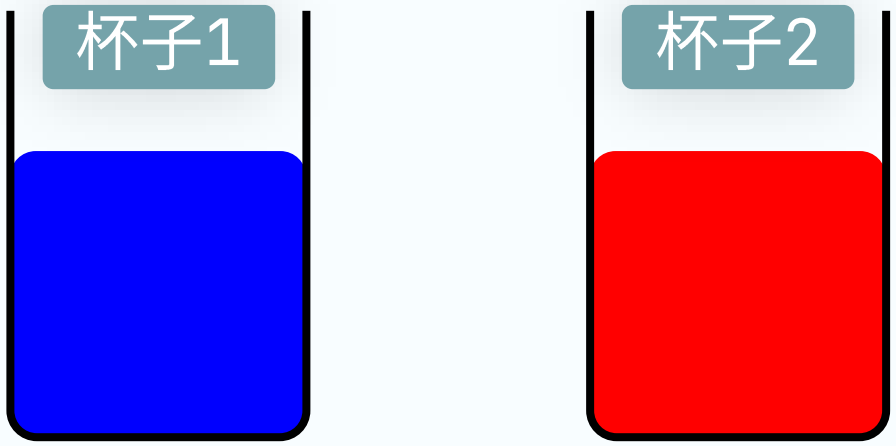
將在新杯子的液體倒入裝原本裝另一個液體的杯子裡



交换的概念

交換的概念

如此一來，就完成了兩種杯中液體的互換



選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

將未排序的第一個元素記錄下來

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

將最小值的元素覆寫到未排序的第一個元素的位置

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

將原本未排序的第一個元素覆寫到最小值的位置

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

如此一來，就可以完成交換兩元素

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    tmp = arr[to_sort];
    arr[to_sort] = arr[min_index];
    arr[min_index] = tmp;

}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
```

覺得太麻煩？

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    swap(arr[to_sort], arr[min_index]);
}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
cout << endl;

return 0;
```

覺得太麻煩？ 可以改寫成這樣！

選擇排序法

```
= 0; to_sort < arr_length - 1; to_sort++){
o_sort;
k = to_sort + 1; check < arr_length; check++){
        if (arr[check] < arr[min_index]){
            min_index = check;
        }
    }

    swap(arr[to_sort], arr[min_index]);
}

for (int i = 0; i < arr_length; i++){
    cout << arr[i] << " ";
}
cout << endl;

return 0;
```

覺得太麻煩？

可以改寫成這樣！

利用swap(a, b)交換a和b

選擇排序法

如果要用swap()，
需要加入標頭檔 algorithm

```
1   #include <iostream>
2   #include <algorithm>
3   using namespace std;
4
5   int main()
6   {
7       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
8
9       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
10          min_index = to_sort;
11          for (int check = to_sort + 1; check < arr_length; check++){
12              if (arr[check] < arr[min_index]){
```

選擇排序法

```
sort];
= arr[min_index];
] = tmp;

    }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

輸出

選擇排序法

```
sort];
= arr[min_index];
] = tmp;

    }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

利用for迴圈，使 i 從 0 到 陣列長度-1

選擇排序法

```
sort];
= arr[min_index];
] = tmp;

    }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

輸出arr[i]

選擇排序法

```
                sort];
                = arr[min_index];
                ] = tmp;

        }

        for (int i = 0; i < arr_length; i++){
            cout << arr[i] << " ";
        }
        cout << endl;

        return 0;
    }
```

換行

選擇排序法

```
sort];
= arr[min_index];
] = tmp;

    }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

如此一來，就完成了陣列的輸出

選擇排序法

```
sort];
= arr[min_index];
] = tmp;
```

```
19
20      }
21
22      f
23
24      }
25      c
26
27      r
28  }
```

selection_sort.exe

```
1 2 3 4 5

Process returned 0 (0x0)   execution time : 0.051 s
Press any key to continue.
```

輸出

選擇排序法

```
sort];
= arr[min_index];
] = tmp;
```

```
19
20      }
21
22      f
23
24      }
25      c
26
27      r
28  }
```

selection_sort.exe

```
1 2 3 4 5

Process returned 0 (0x0)   execution time : 0.051 s
Press any key to continue.
```

輸出

如果想要由大到小排列該怎麼做呢？

選擇排序法

```
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] < arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
17          arr[to_sort] = arr[min_index];
18          arr[min_index] = tmp;
```

改成 >

選擇排序法

```
5   {
6       int arr[5] = {5, 2, 1, 3, 4}, arr_length = 5, min_index, tmp;
7
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] > arr[min_index]){
12                  min_index = check;
13              }
14          }
15
16          tmp = arr[to_sort];
17          arr[to_sort] = arr[min_index];
18          arr[min_index] = tmp;
```

改成 >

選擇排序法

```
selection_sort.exe

5 4 3 2 1

Process returned 0 (0x0)   execution time : 0.041 s
Press any key to continue.
```

選擇排序法

假設要排N筆資料

選擇排序法

假設要排N筆資料

- 排序循環N-1次

```
8       for (int to_sort = 0; to_sort < arr_length - 1; to_sort++){
9           min_index = to_sort;
10          for (int check = to_sort + 1; check < arr_length; check++){
11              if (arr[check] > arr[min_index]){
12                  min_index = check;
13              }
14          }
15
```

選擇排序法

假設要排N筆資料

- 比較N*(N-1)/2次

第1次排序循環，要比較N-1次
第2次排序循環，要比較N-2次

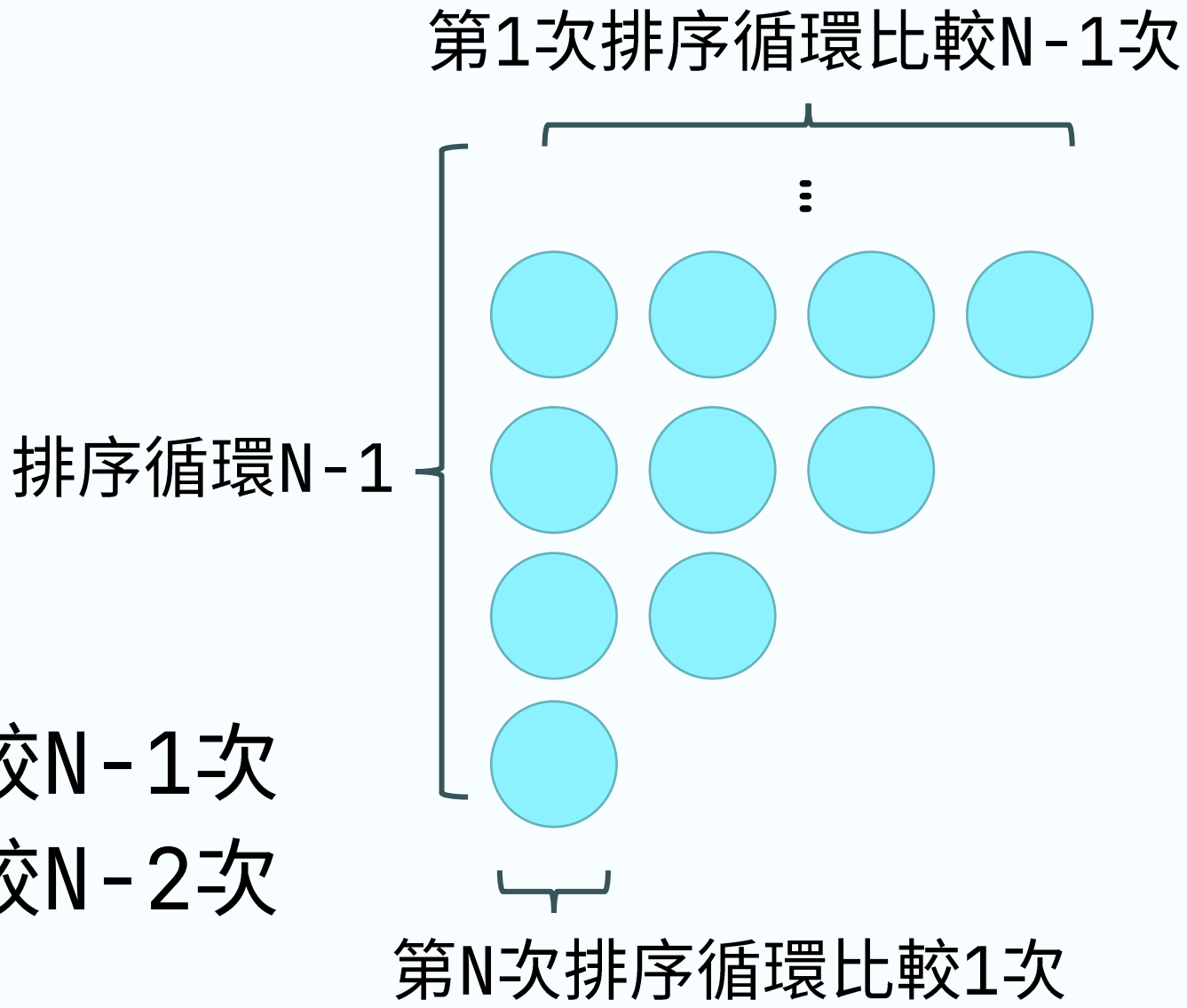
……
第N次排序循環，要比較1次

選擇排序法

假設要排N筆資料

- 比較N*(N-1)/2次

第1次排序循環，要比較N-1次
第2次排序循環，要比較N-2次
……
第N次排序循環，要比較1次



可使用梯形公式計算

選擇排序法

假設要排N筆資料

- 最多交換次數N-1次

假設每次排序循環都需要交換，則最多的交換次數為N-1次

選擇排序法

假設要排N筆資料

- 排序循環N-1次
- 比較N*(N-1)/2次
- 最多交換次數N-1次

氣泡排序法

Bubble Sort

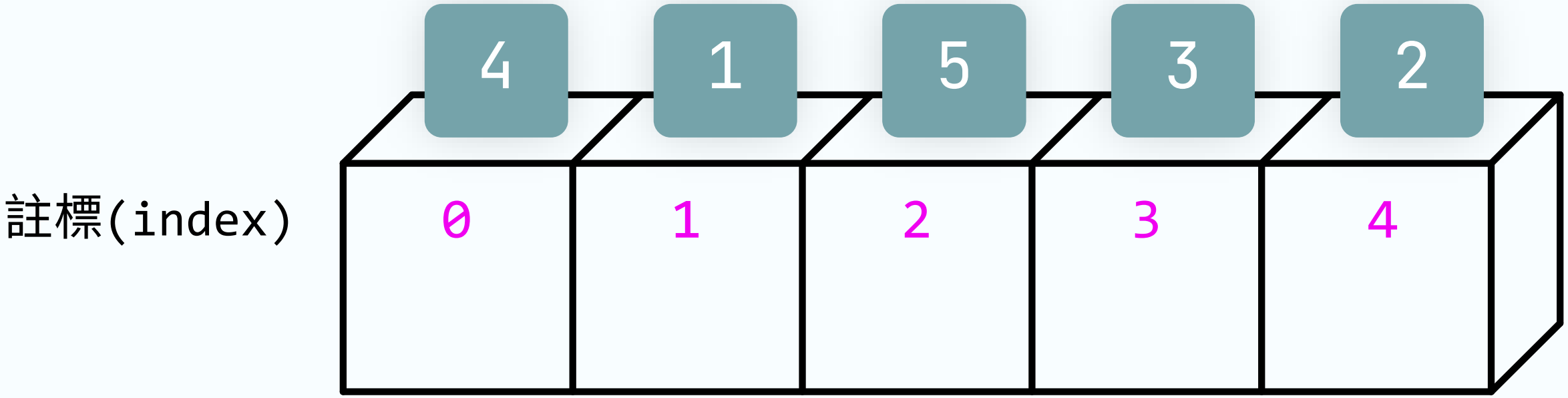
氣泡排序法

精神：

- 氣泡上浮
- 每次比較相鄰2個元素，排序錯誤就交換

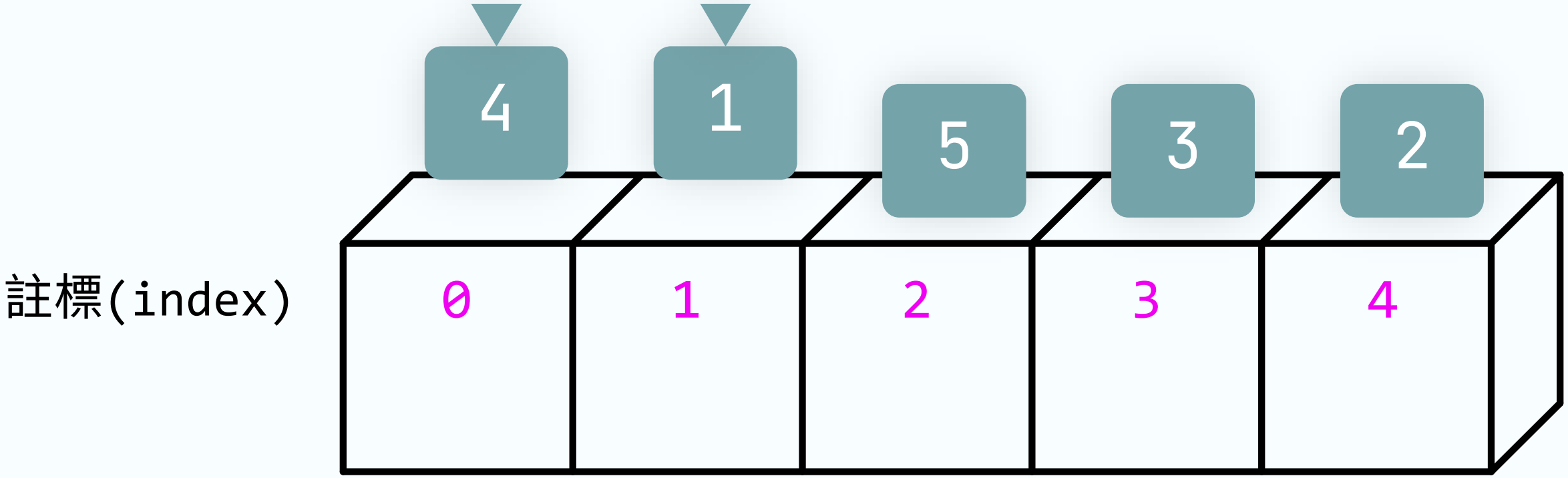
氣泡排序法

假設陣列中有5個數，
要從小到大排列



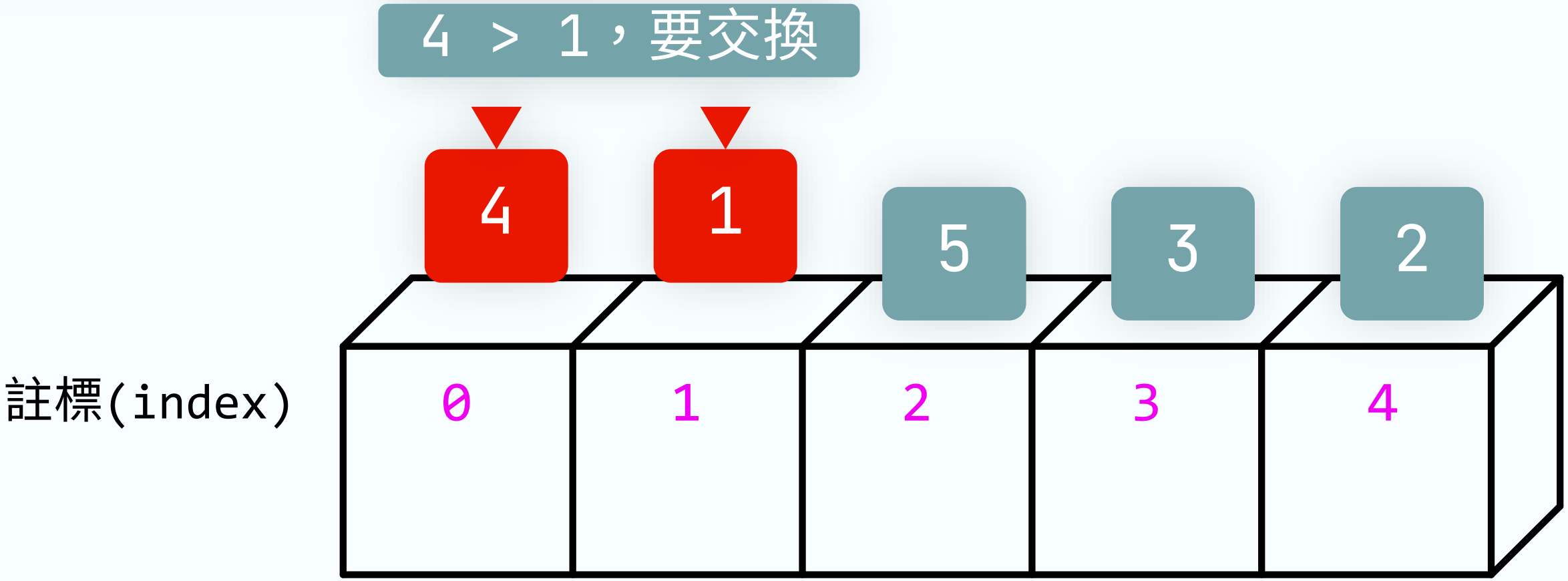
氣泡排序法

假設陣列中有5個數，
要從小到大排列



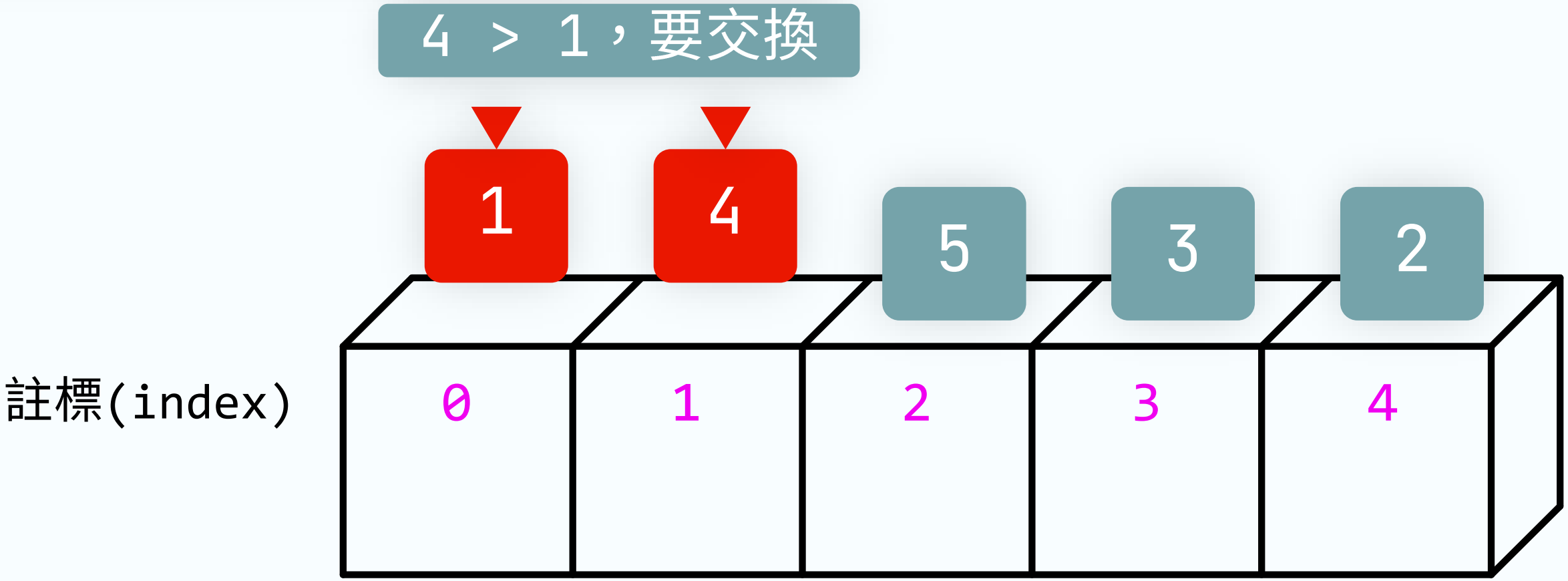
氣泡排序法

假設陣列中有5個數，
要從小到大排列

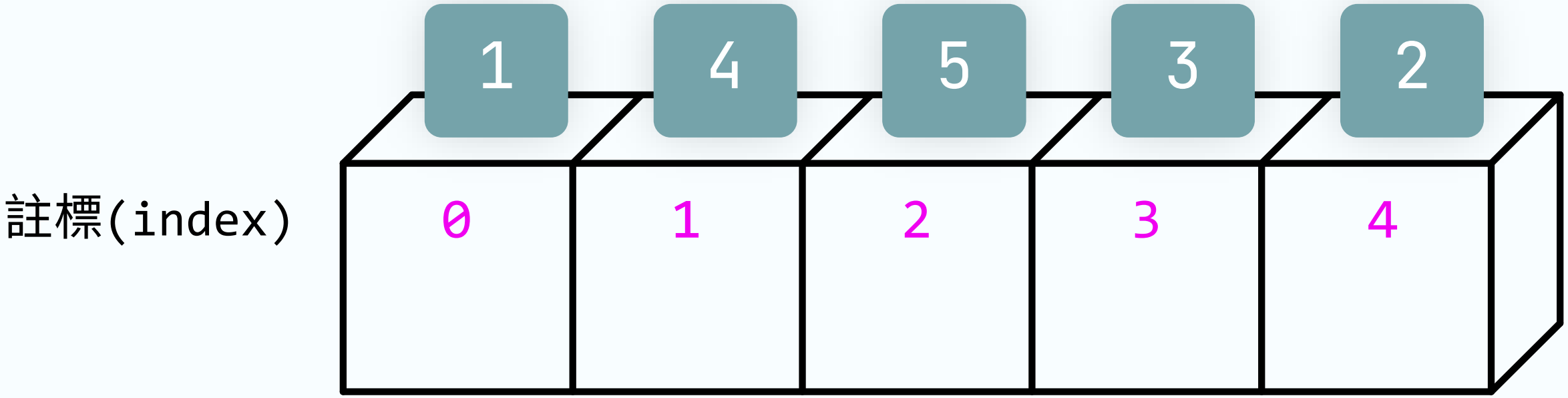


氣泡排序法

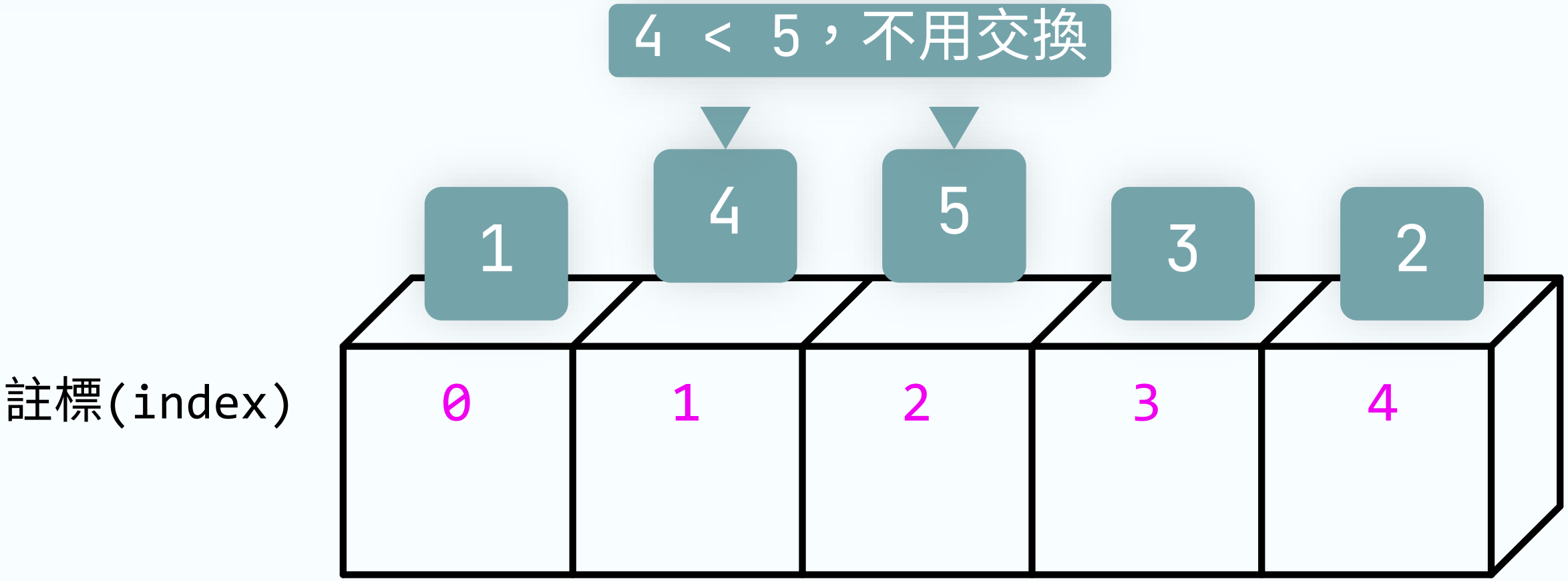
假設陣列中有5個數，
要從小到大排列



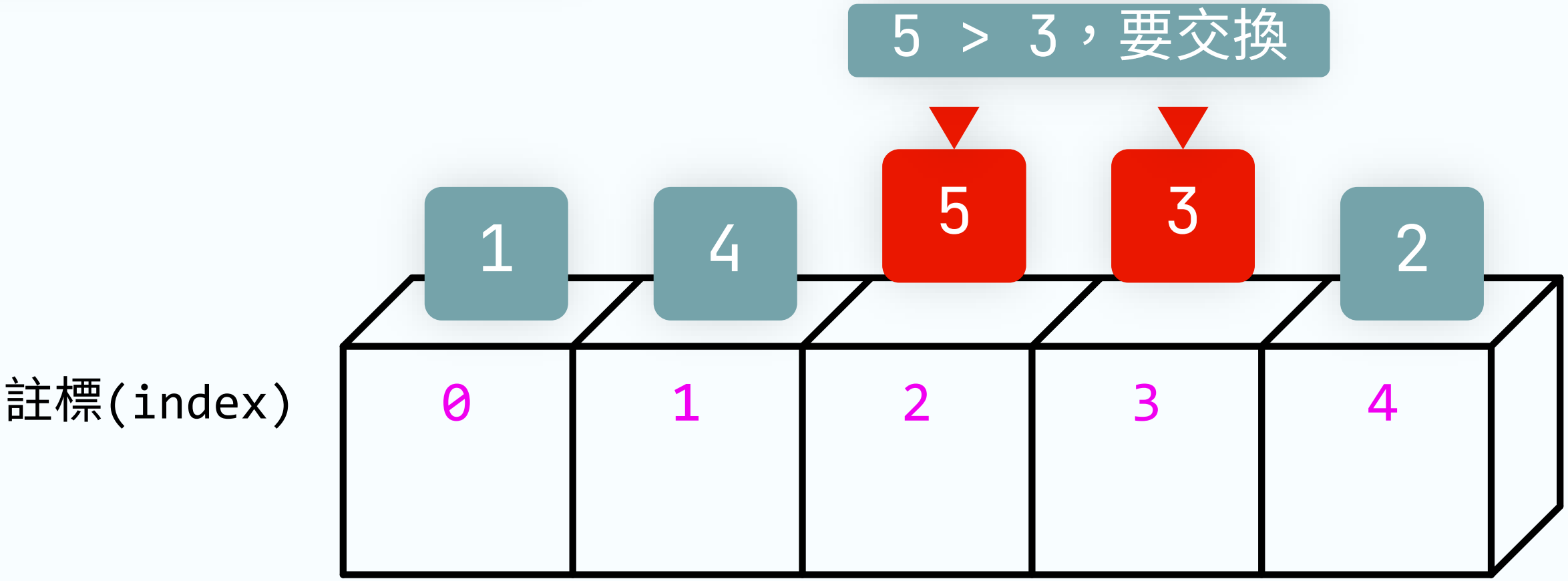
氣泡排序法



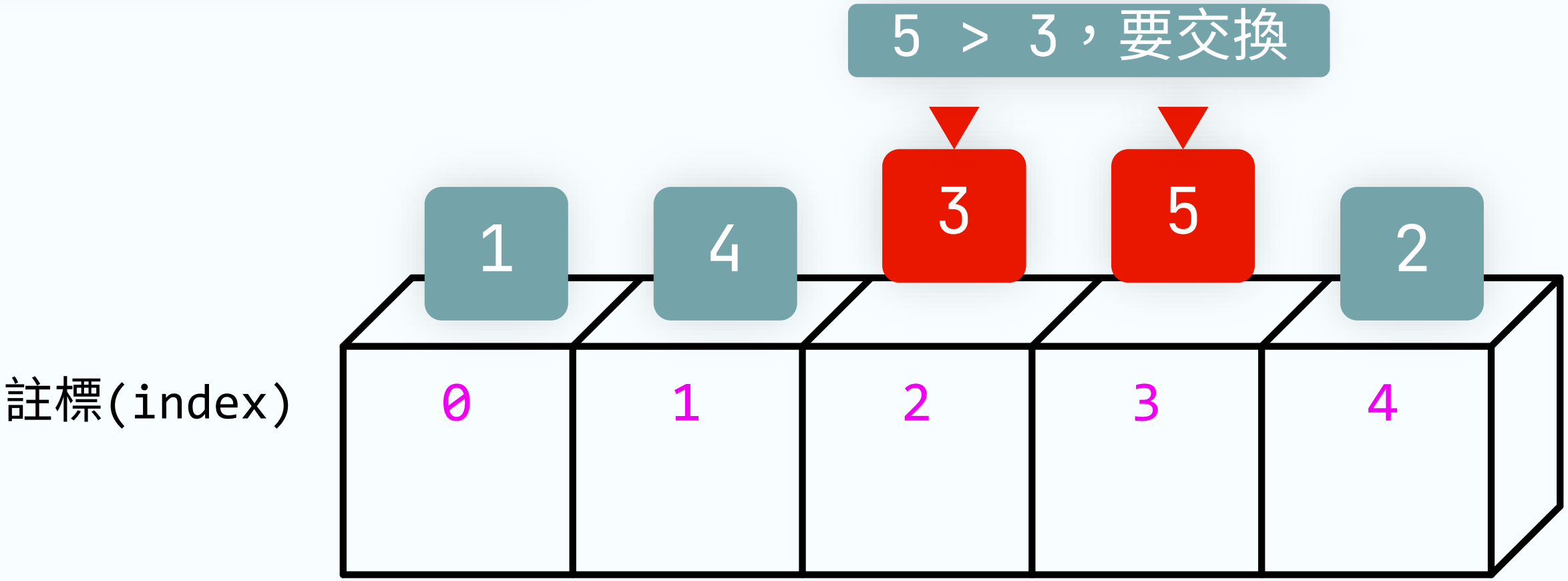
氣泡排序法



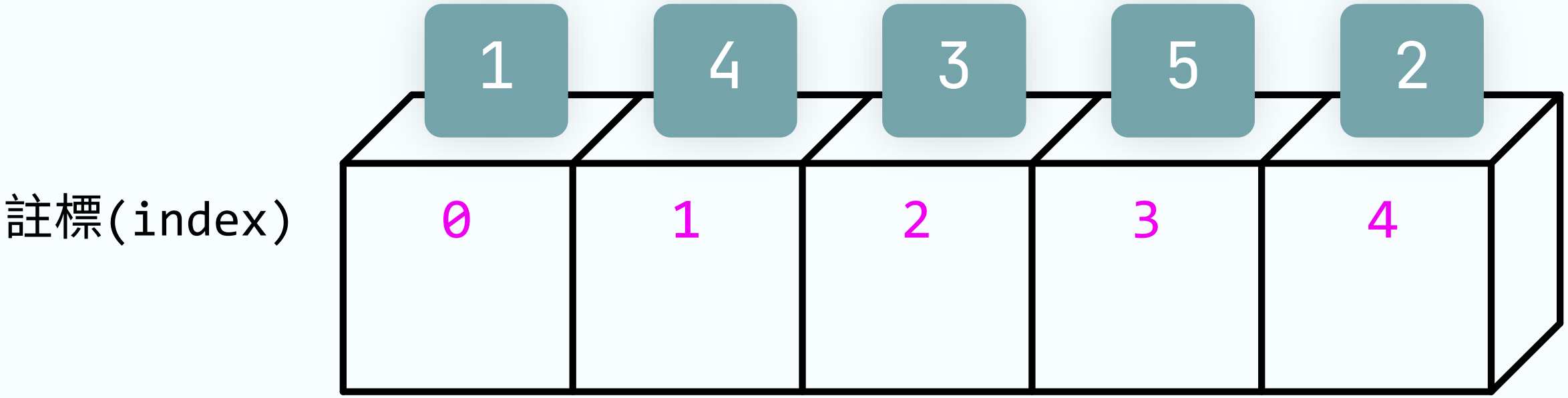
氣泡排序法



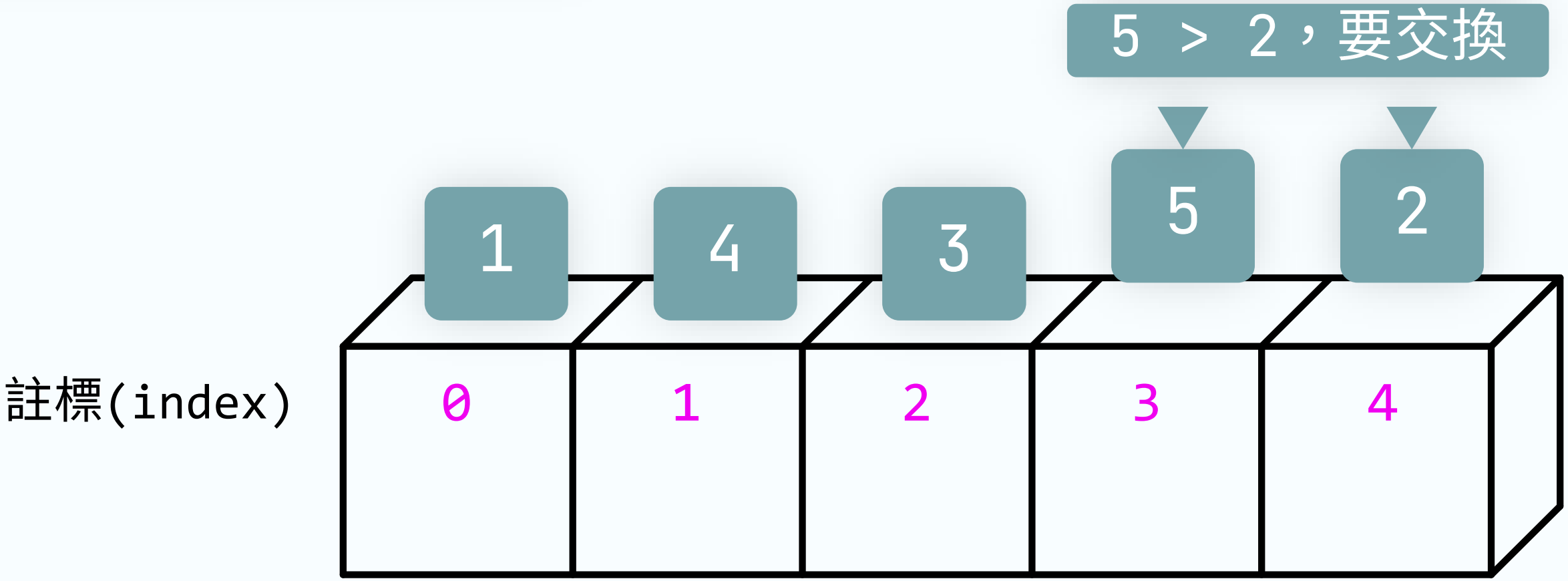
氣泡排序法



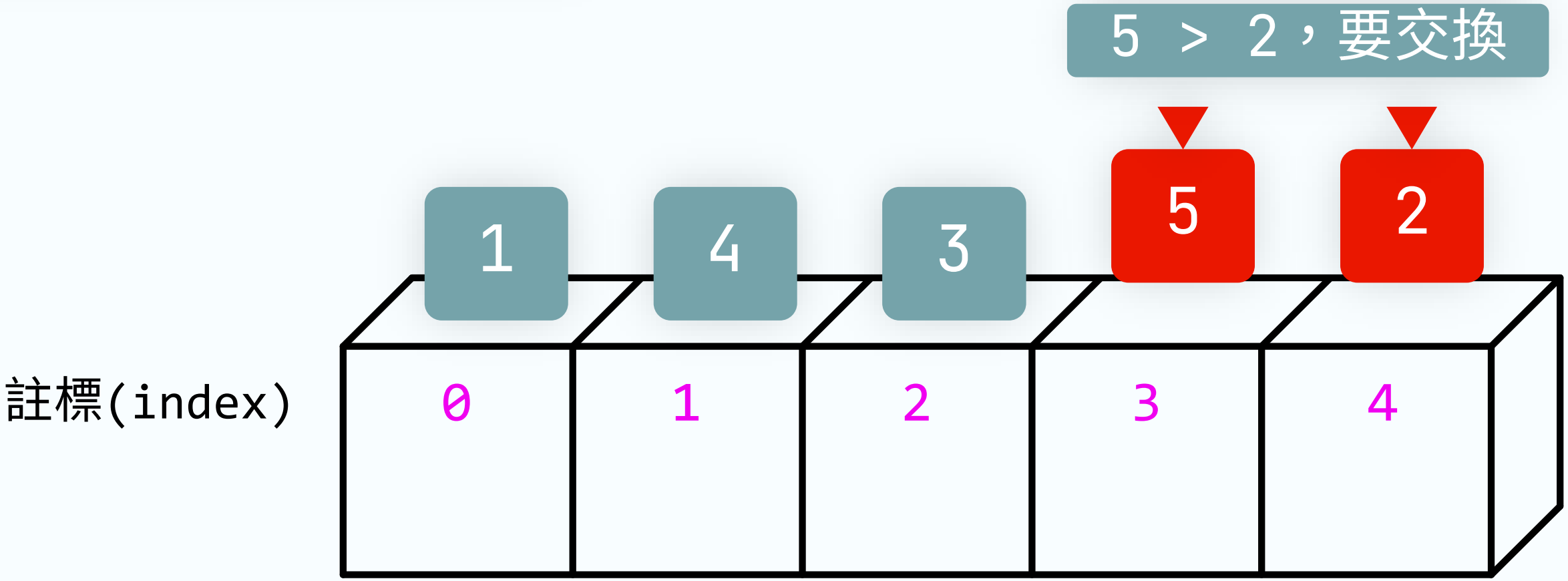
氣泡排序法



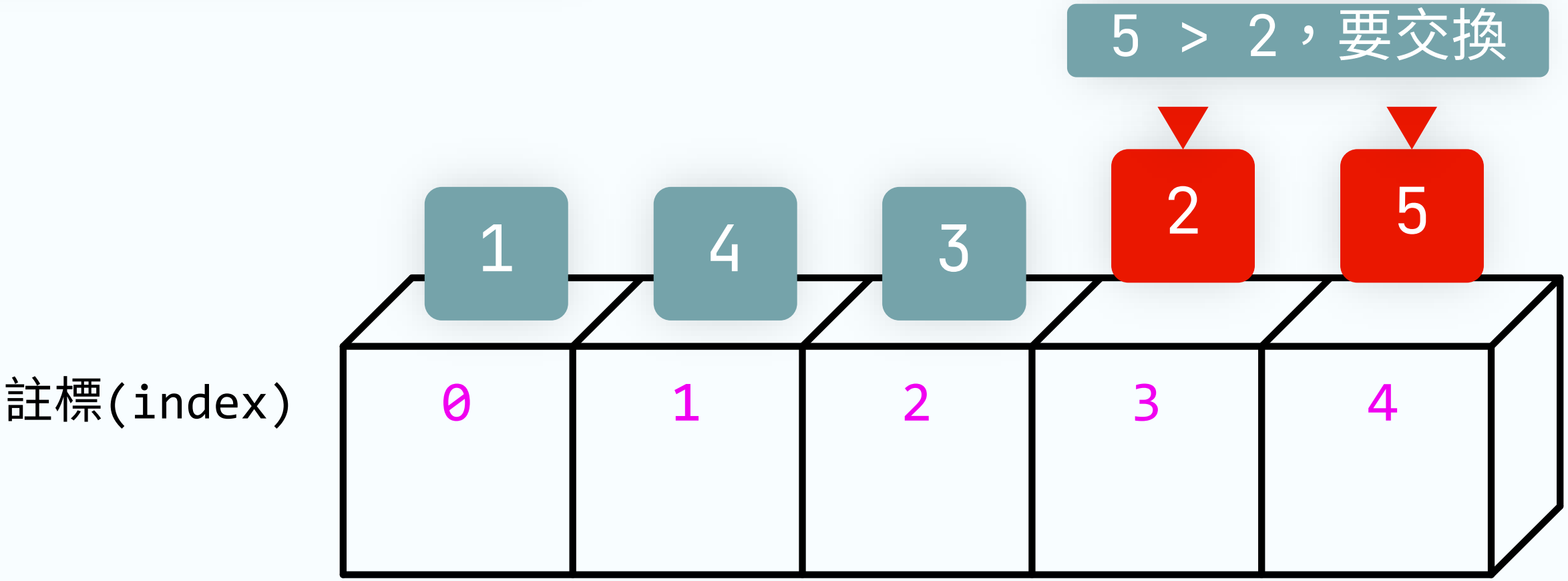
氣泡排序法



氣泡排序法



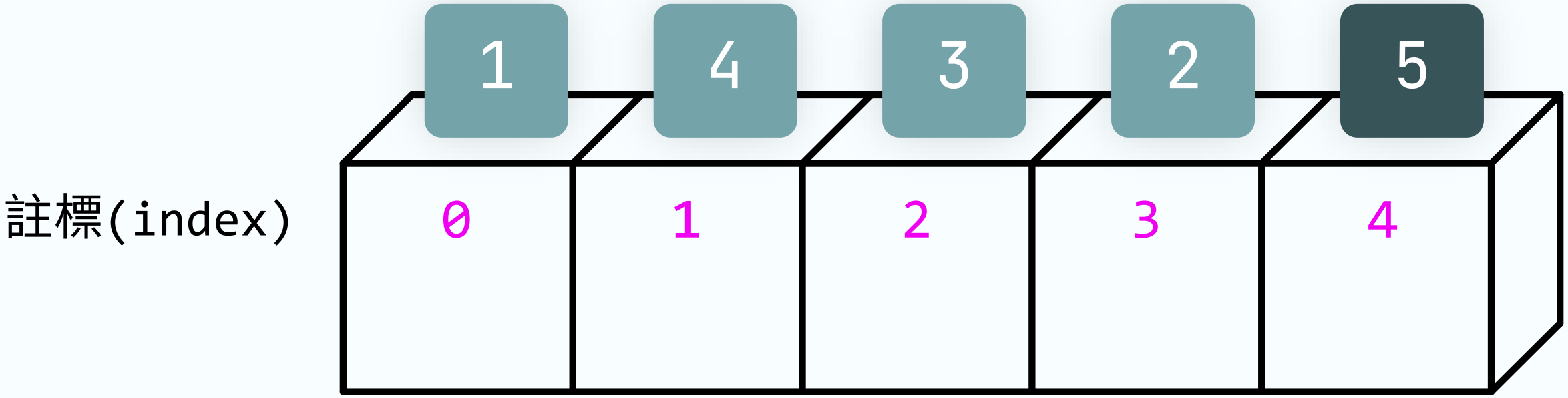
氣泡排序法



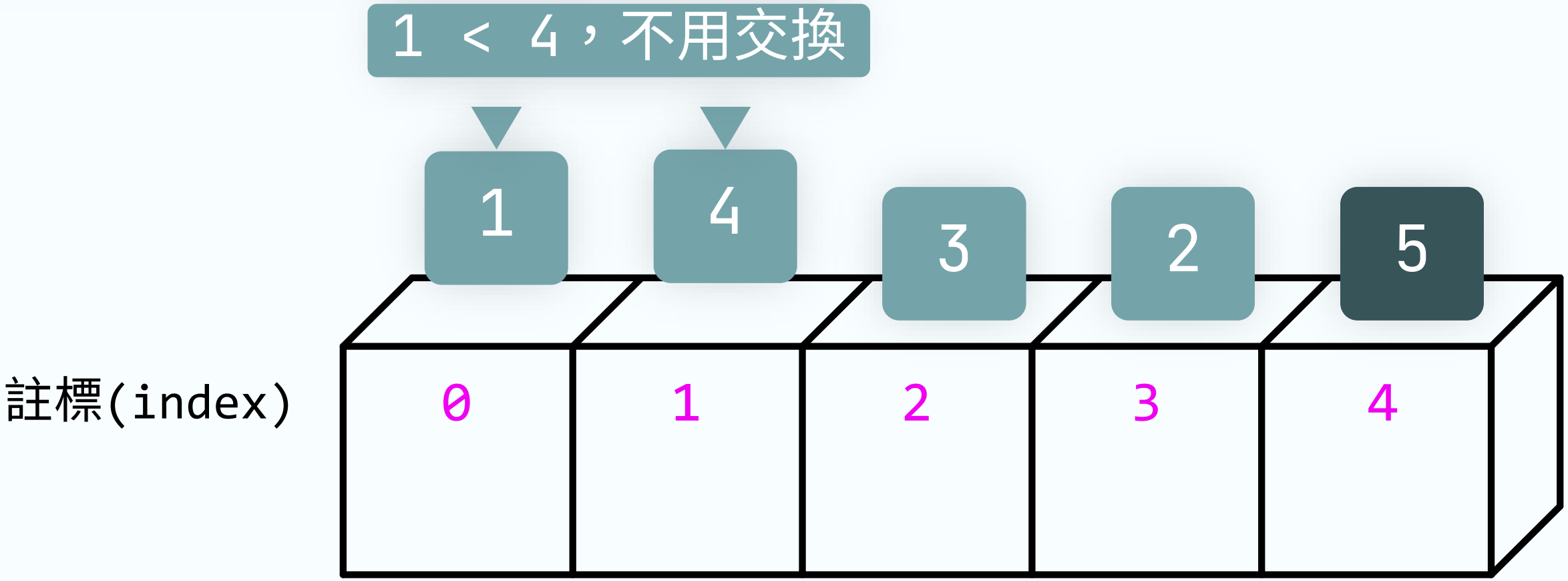
氣泡排序法

如同氣泡上浮般

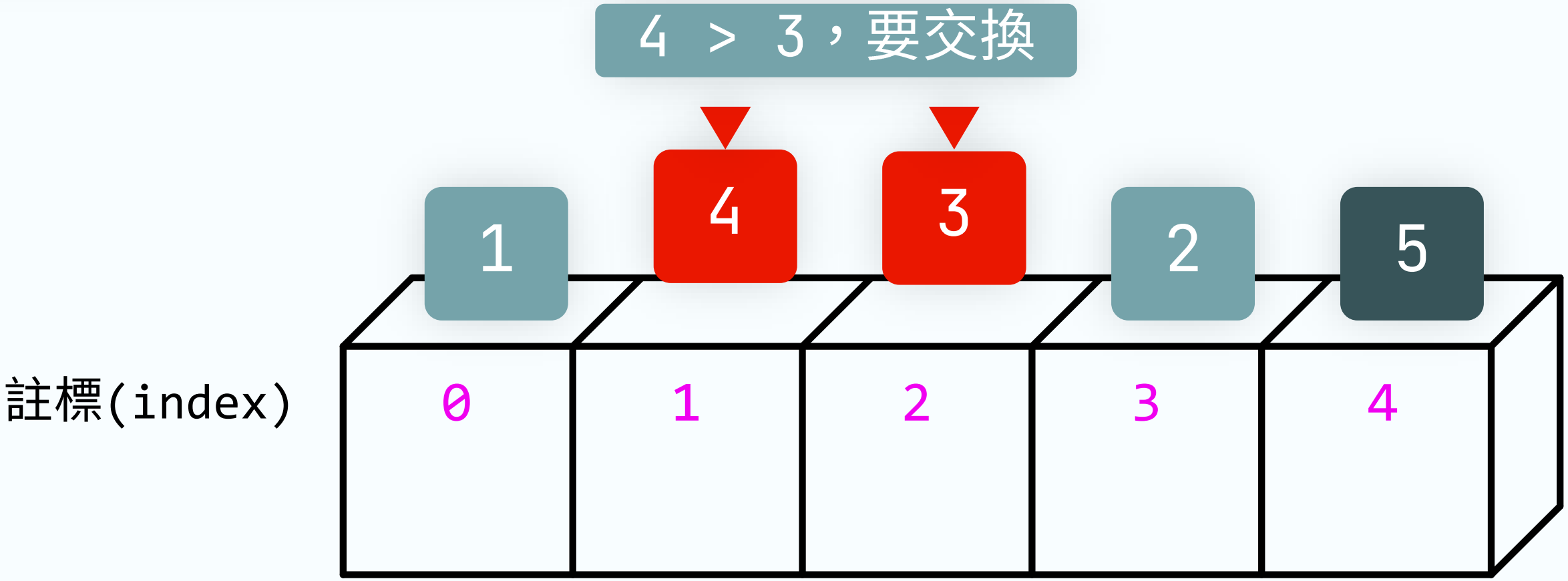
排好了



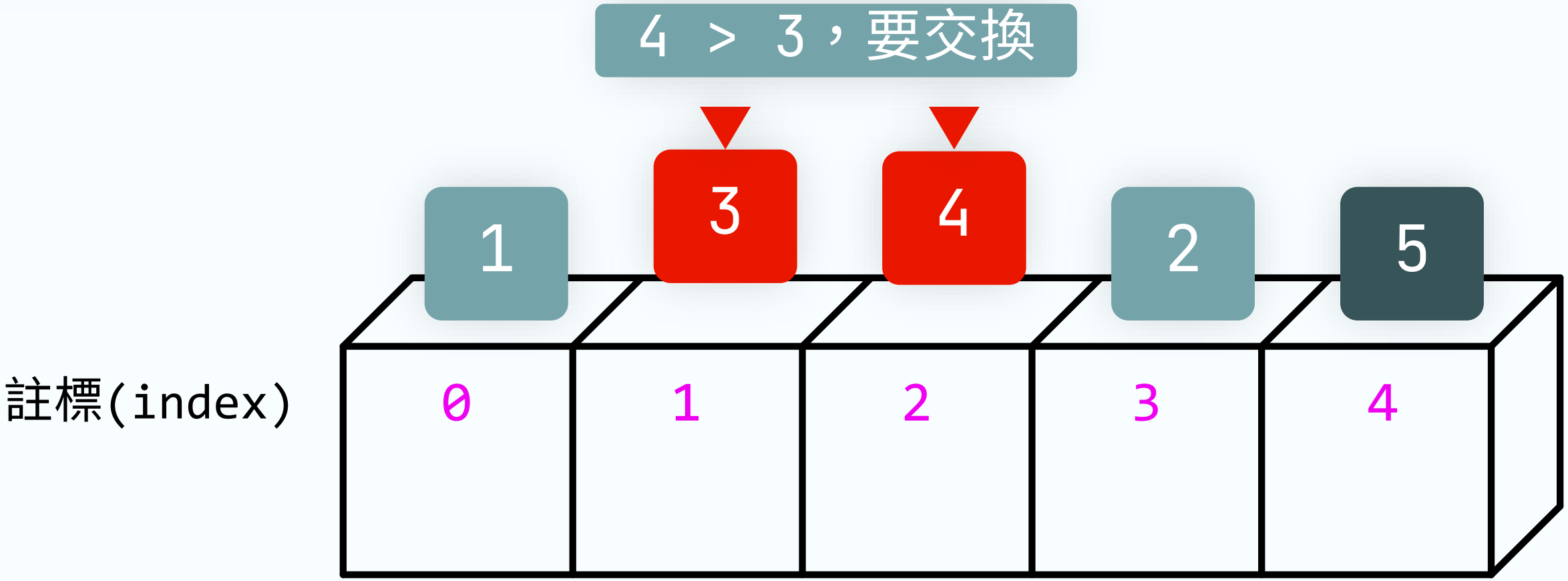
氣泡排序法



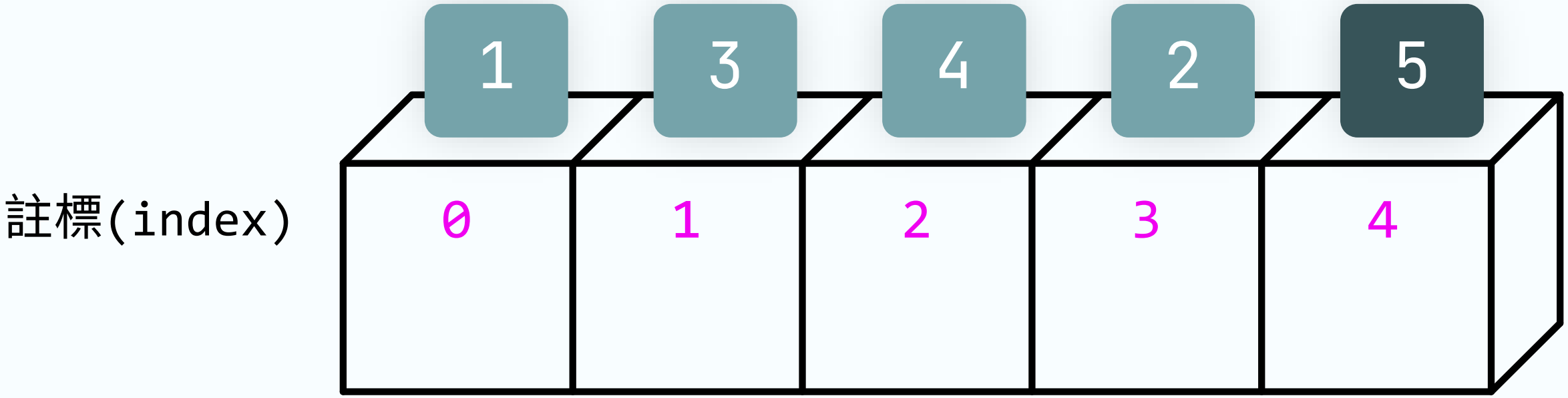
氣泡排序法



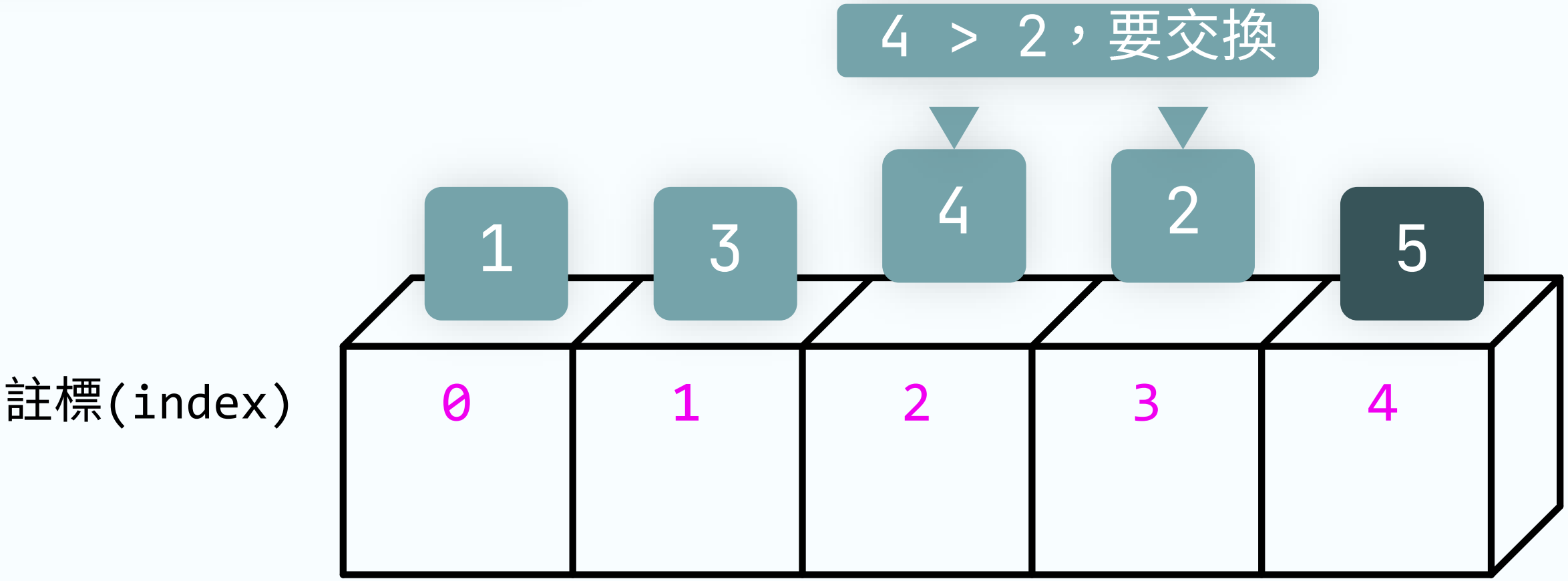
氣泡排序法



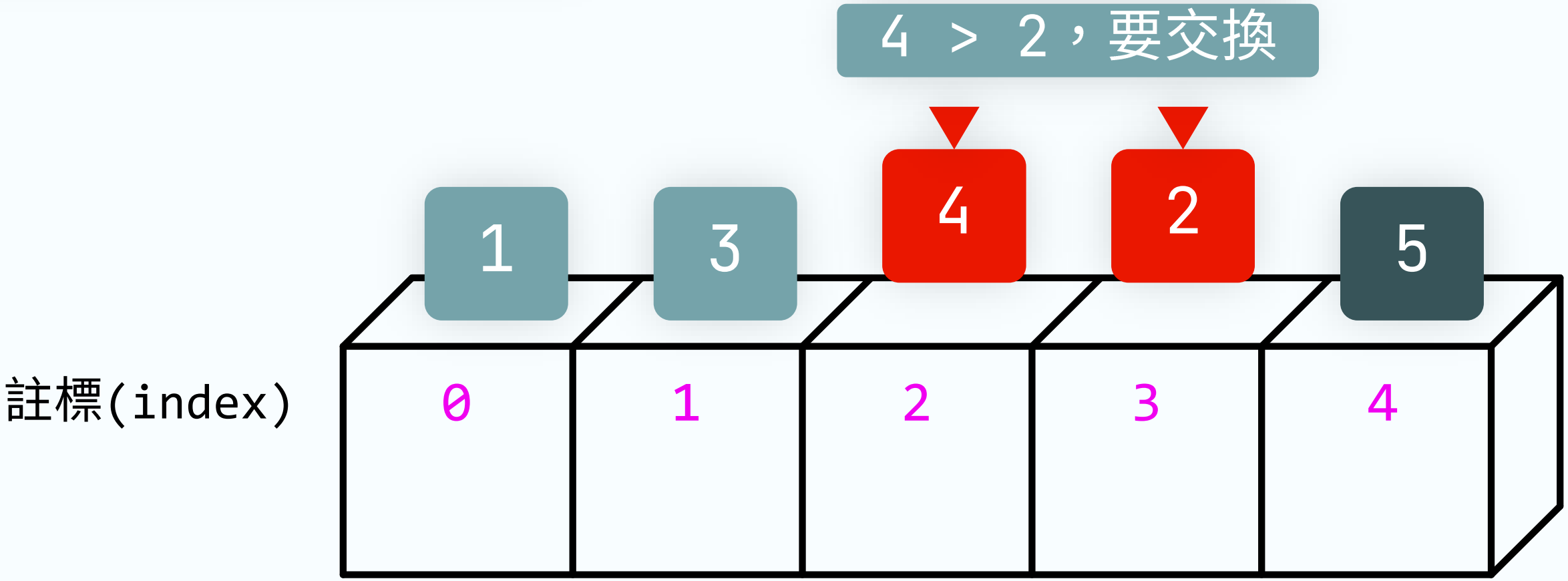
氣泡排序法



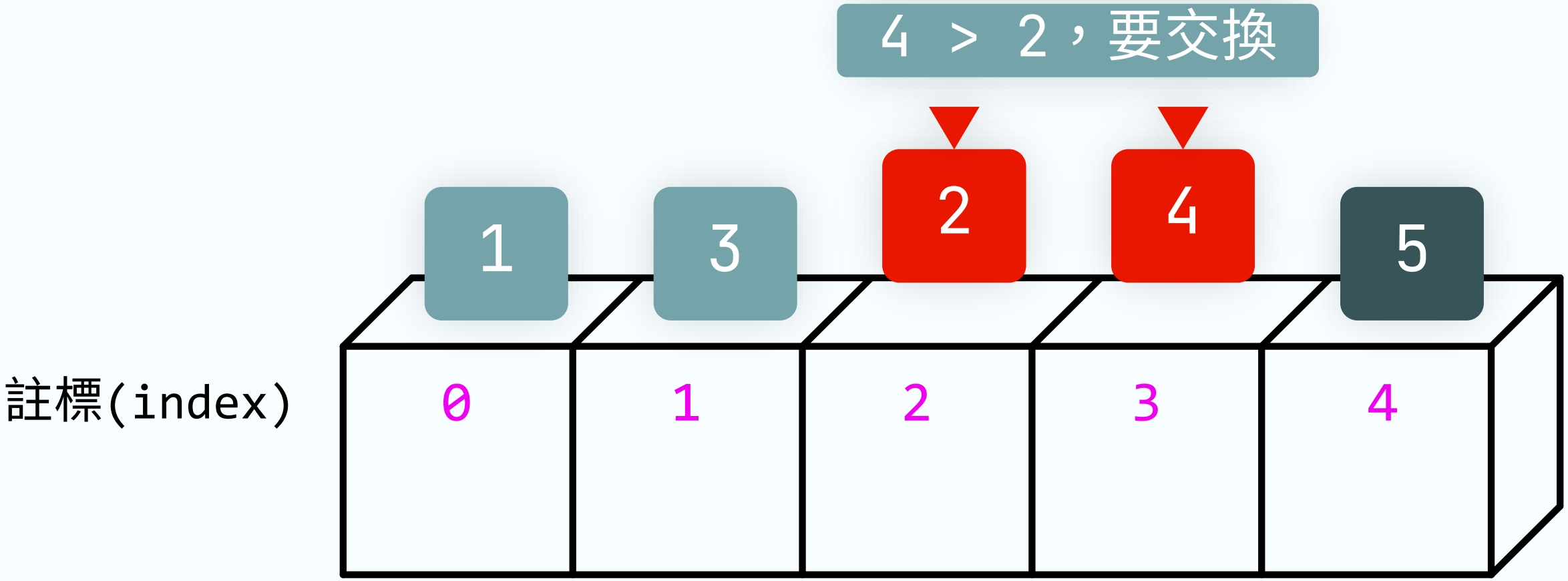
氣泡排序法



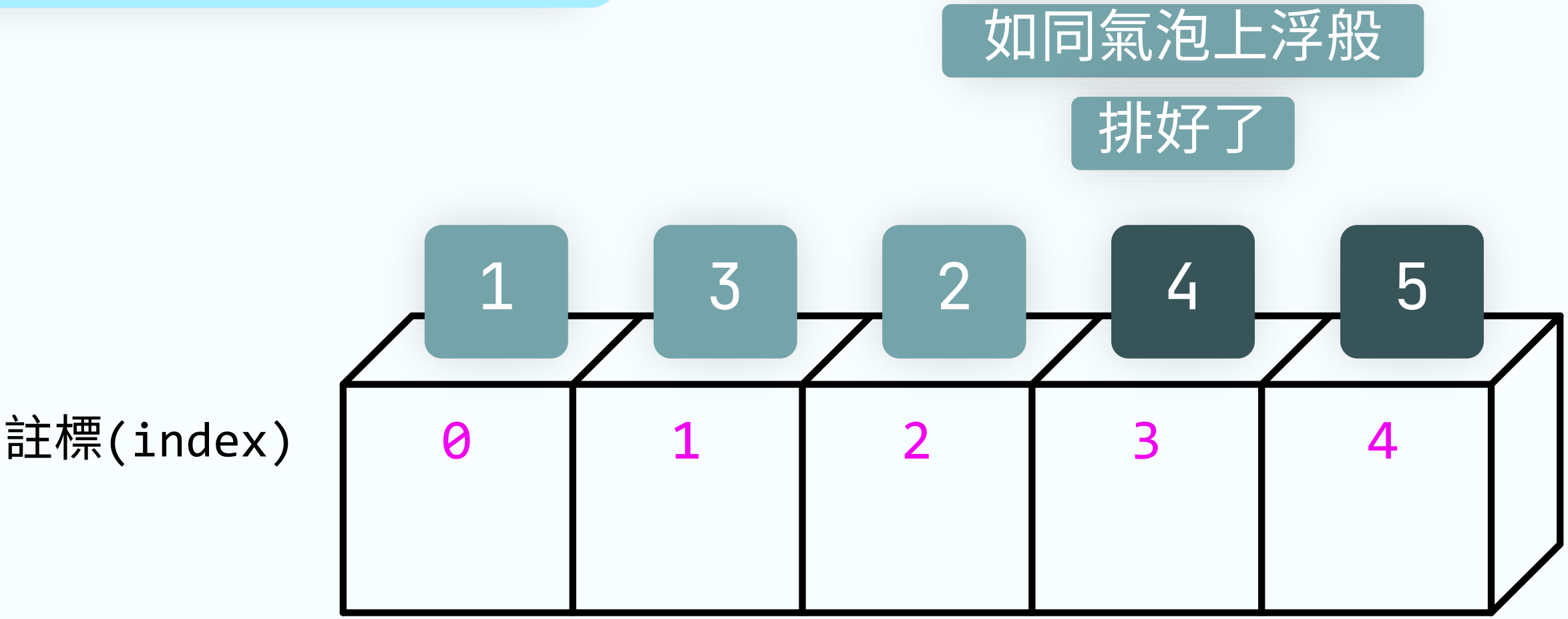
氣泡排序法



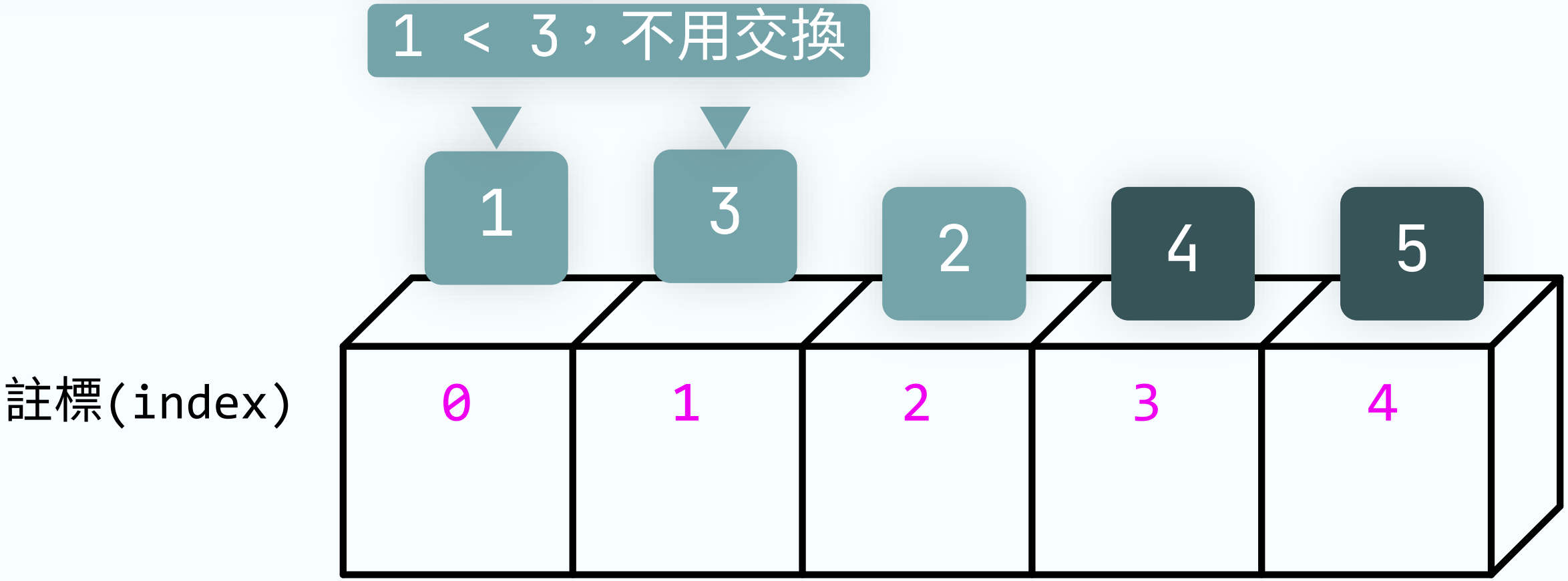
氣泡排序法



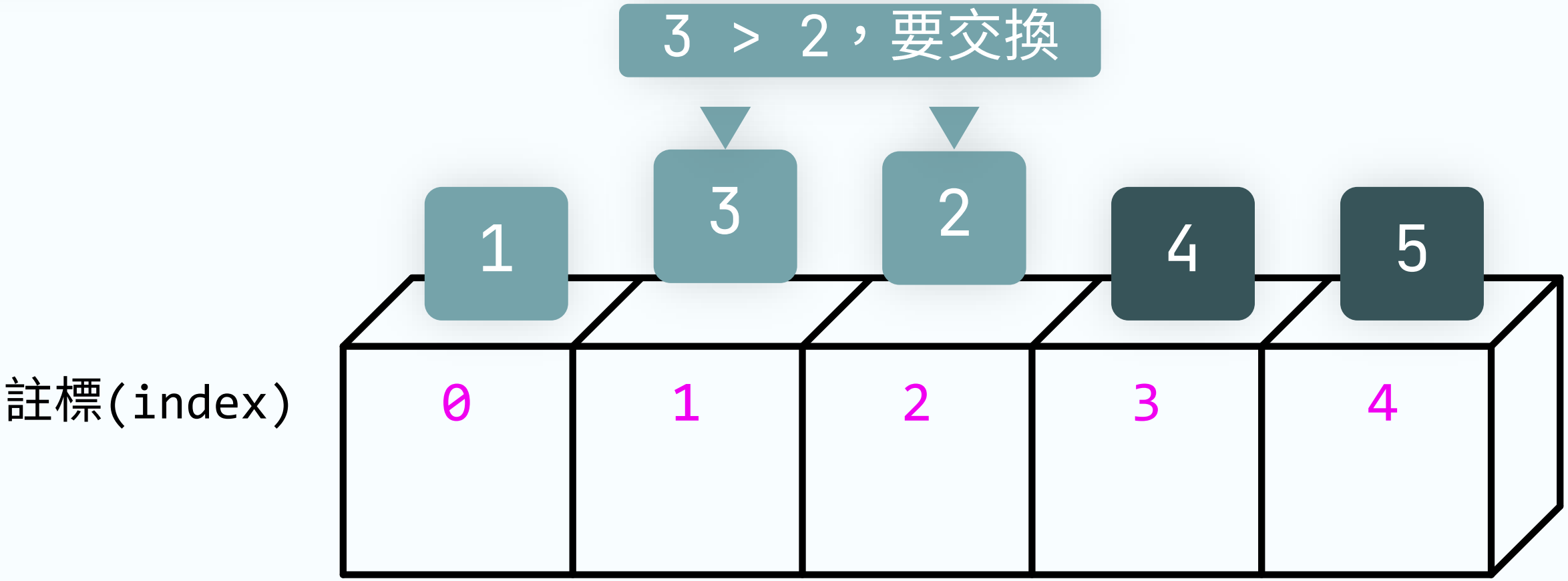
氣泡排序法



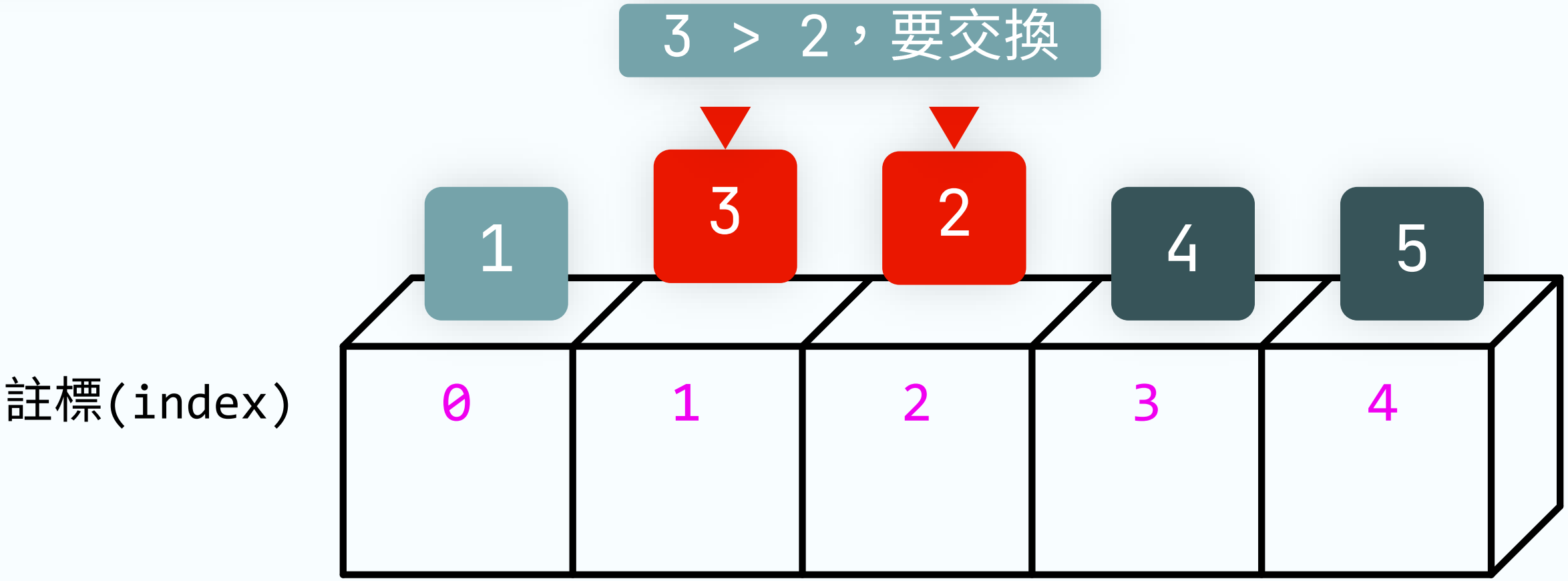
氣泡排序法



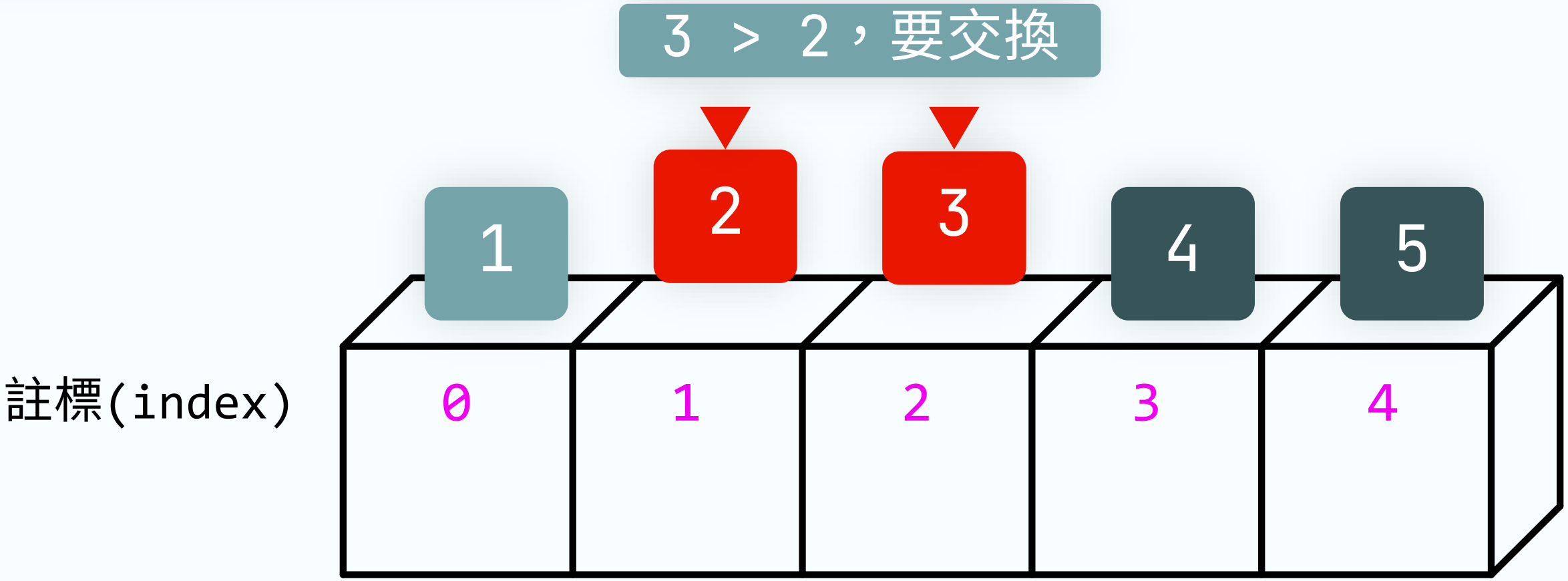
氣泡排序法



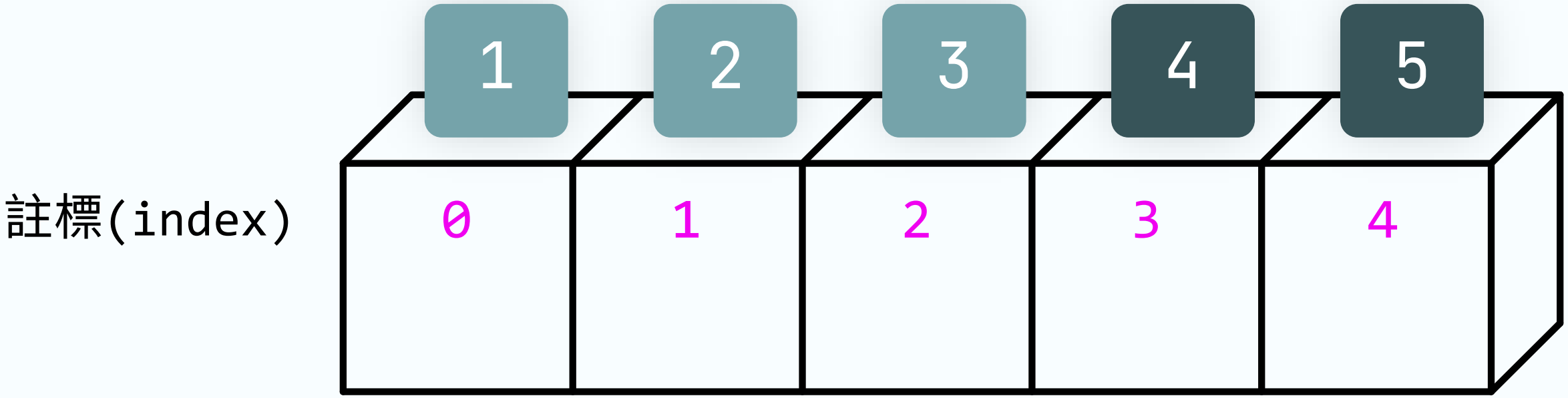
氣泡排序法



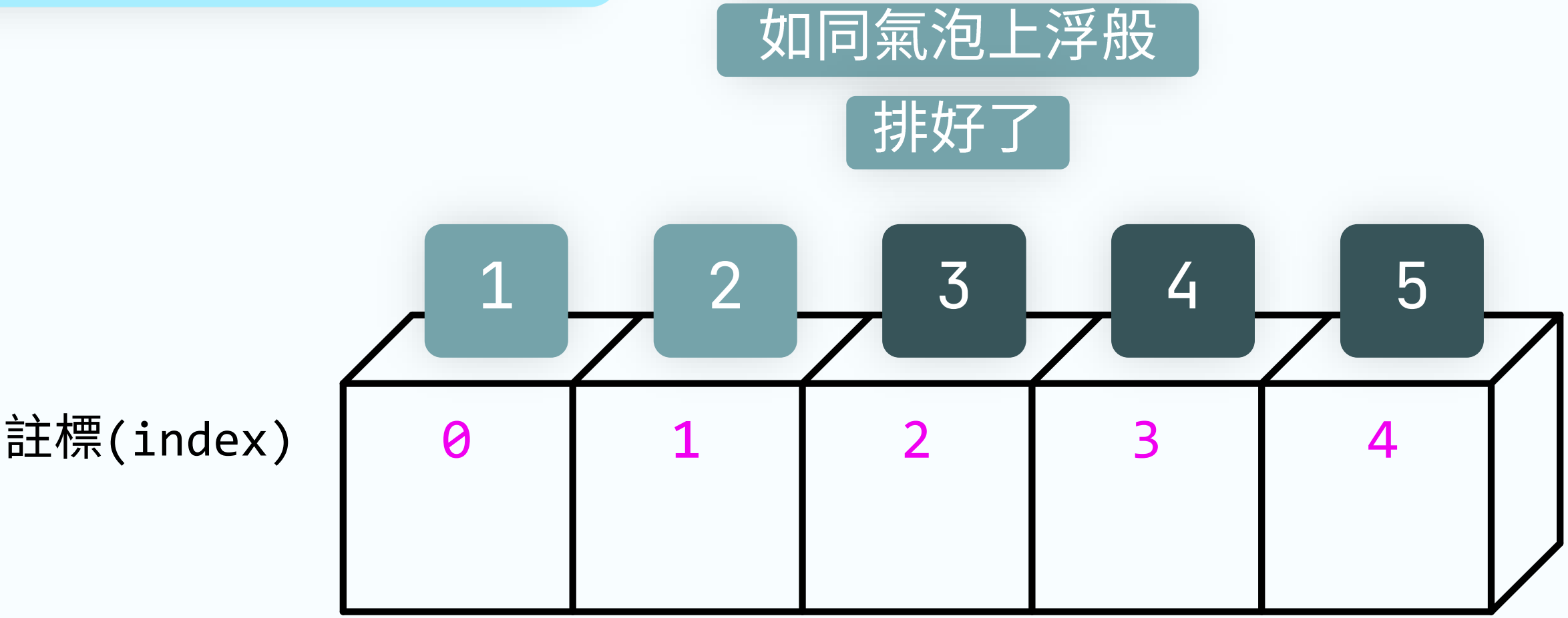
氣泡排序法



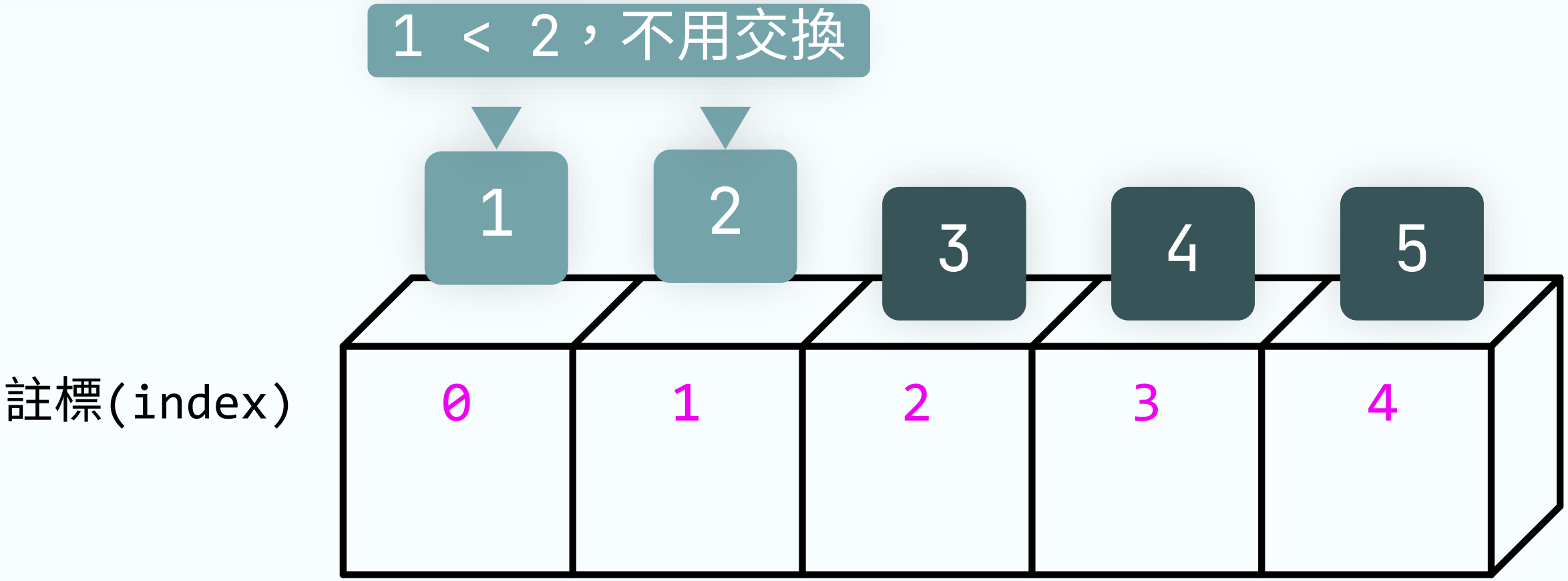
氣泡排序法



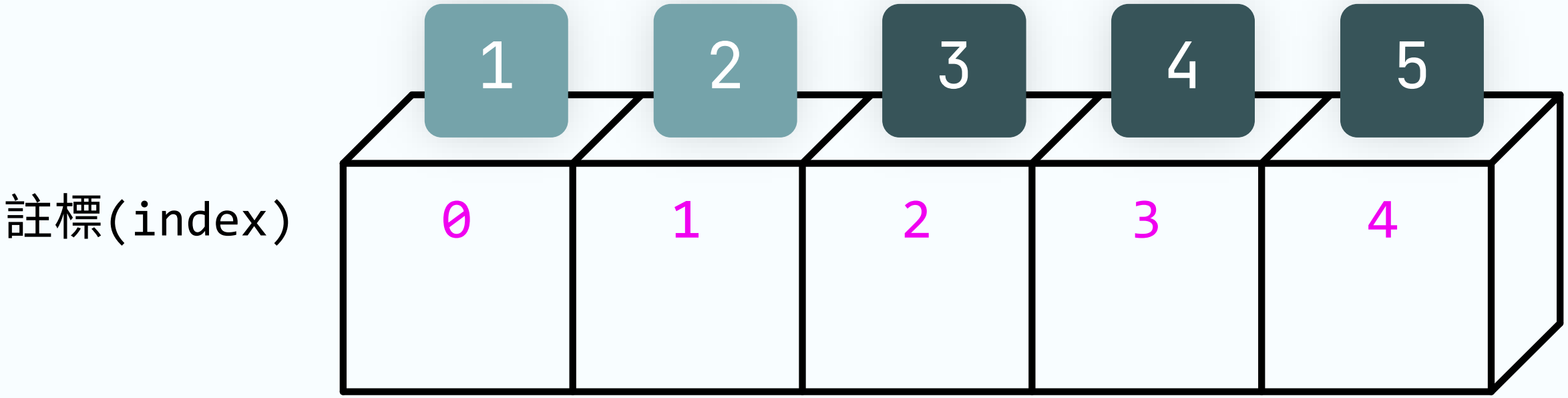
氣泡排序法



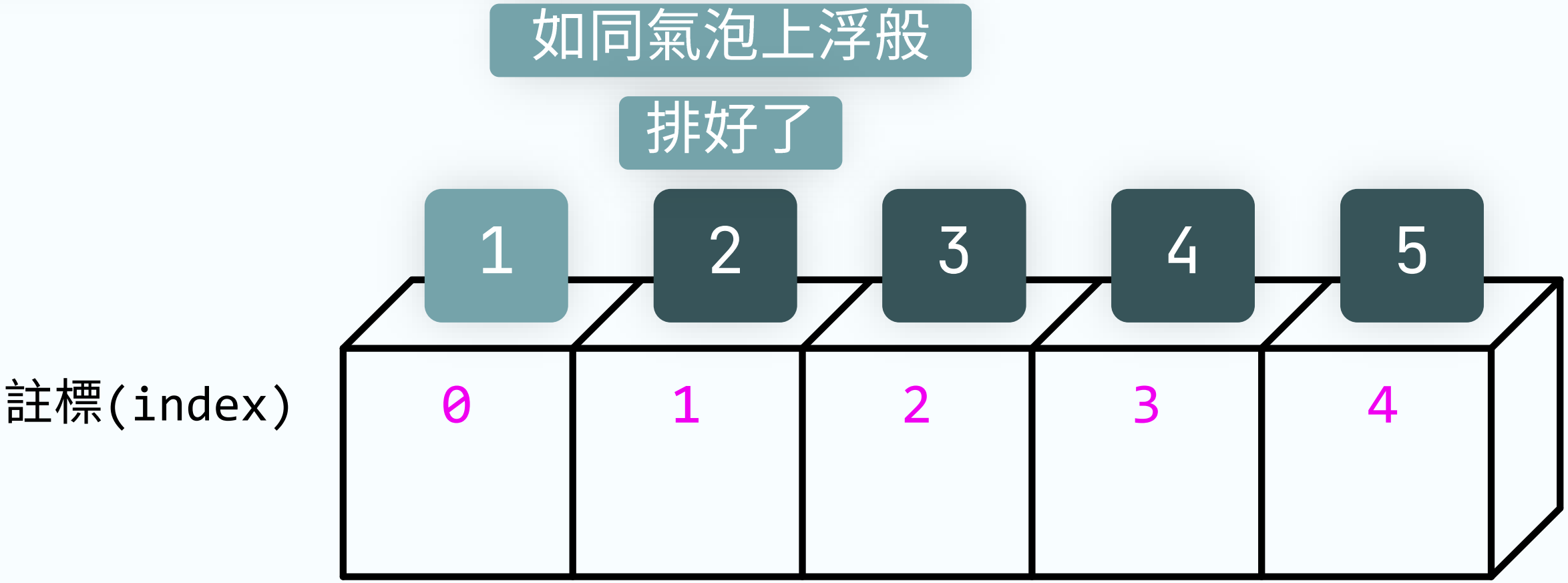
氣泡排序法



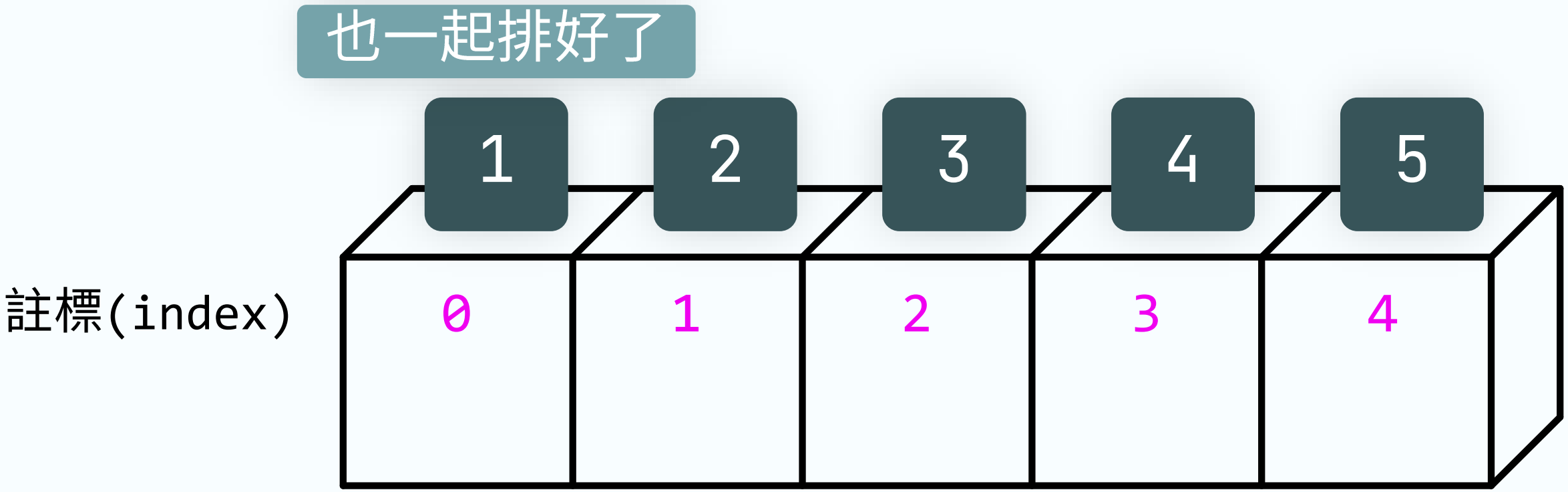
氣泡排序法



氣泡排序法

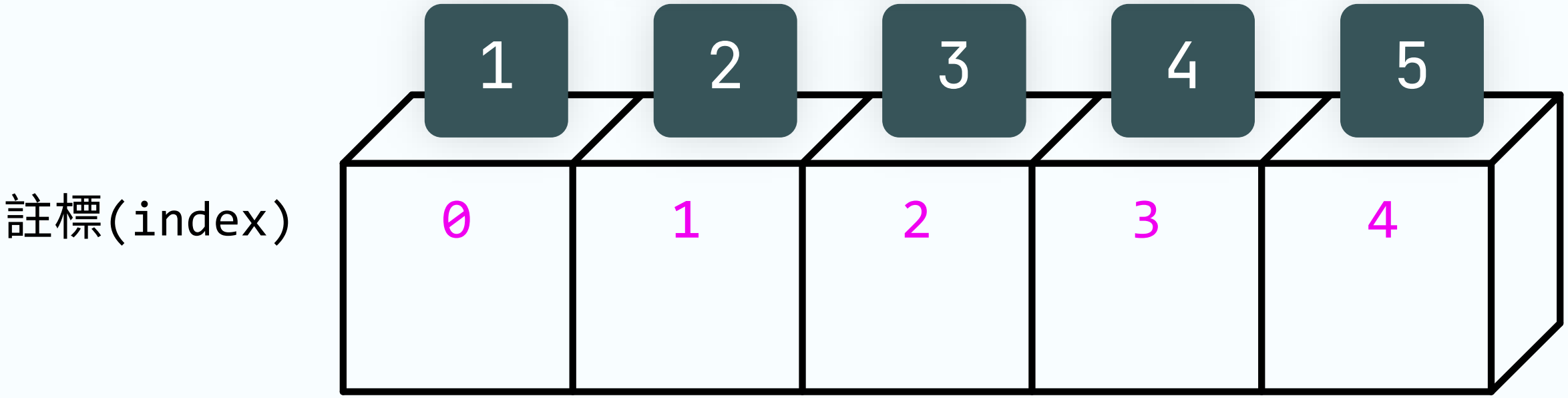


氣泡排序法



氣泡排序法

如此一來，就排好資料們了



氣泡排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
18     for (int i = 0; i < arr_length; i++){
19         cout << arr[i] << " ";
20     }
21     cout << endl;
22
23     return 0;
24 }
```

比較相鄰2元素

氣泡排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
18     for (int i = 0; i < arr_length; i++){
19         cout << arr[i] << " ";
20     }
21     cout << endl;
22
23     return 0;
24 }
```

比較相鄰2元素

交換

氣泡排序法

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
18     for (int i = 0; i < arr_length; i++){
19         cout << arr[i] << " ";
20     }
21     cout << endl;
22
23     return 0;
24 }
```

比較相鄰2元素

交換

輸出

氣泡排序法

宣告長度5的陣列arr

```
int
{
    int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;

    for (int i = 0; i < arr_length - 1; i++){
        for (int j = 0; j < arr_length - i - 1; j++){
            if (arr[j] > arr[j + 1]){
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }

```

氣泡排序法

宣告長度5的陣列arr

arr_length
記錄陣列arr長度

```
3
4   int
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

氣泡排序法

宣告長度5的陣列arr

arr_length
記錄陣列arr長度

tmp交換
暫存用

```
int
{
    int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;

    for (int i = 0; i < arr_length - 1; i++){
        for (int j = 0; j < arr_length - i - 1; j++){
            if (arr[j] > arr[j + 1]){
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }

```

氣泡排序法

```
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
```

比較相鄰2元素

氣泡排序法

利用for迴圈，使i從0跑到arr_length - 2

```
int main()
{
    for (int i = 0; i < arr_length - 1; i++){
        for (int j = 0; j < arr_length - i - 1; j++){
            if (arr[j] > arr[j + 1]){
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
```

氣泡排序法

```
3
4   int main()
5   {
6
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

利用for迴圈，使i從0跑到arr_length - 2

利用for迴圈，使j從0跑到arr_length - i - 2

氣泡排序法

```
3
4   int main()
5   {
6
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10
11
12
13
14          }
15      }
16  }
17
```

利用for迴圈，使i從0跑到arr_length - 2

利用for迴圈，使j從0跑到arr_length - i - 2

j會從0跑到尚未排序的倒數第2格

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

如果arr[j] > arr[j+1]

如果arr目前這元素>下一元素

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

如果arr[j] > arr[j+1]

如果arr目前這元素>下一元素

則交換2元素

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

交換

氣泡排序法

```
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
```

arr[j]儲存到tmp

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

arr[j]儲存到tmp

arr[j+1]複寫到arr[j]

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

arr[j]儲存到tmp

arr[j+1]複寫到arr[j]

tmp(原本的arr[j]的值)
複寫到arr[j]

氣泡排序法

```
3
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
```

如此一來，就完成交換了

氣泡排序法

```
3
4  int main()
5  {
6      int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8      for (int i = 0; i < arr_length - 1; i++){
9          for (int j = 0; j < arr_length - i - 1; j++){
10             if (arr[j] > arr[j + 1]){
11                 tmp = arr[j];
12                 arr[j] = arr[j+1];
13                 arr[j+1] = tmp;
14             }
15         }
16     }
17
```

如此一來，就完成排序了

氣泡排序法

```
                    ] > arr[j + 1]){
                    arr[j];
                    ] = arr[j+1];
                    arr[j+1] = tmp;
                }
            }
        }

        for (int i = 0; i < arr_length; i++){
            cout << arr[i] << " ";
        }
        cout << endl;

        return 0;
    }
```

輸出

氣泡排序法

```
                ] > arr[j + 1]){
                arr[j];
                ] = arr[j+1];
                arr[j+1] = tmp;
            }
        }

    for (int i = 0; i < arr_length; i++){
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

利用for迴圈，使 i 從 0 到 陣列長度-1

氣泡排序法

```
                    ] > arr[j + 1]){
                    arr[j];
                    ] = arr[j+1];
                    arr[j+1] = tmp;
                }
            }
        }

        for (int i = 0; i < arr_length; i++){
            cout << arr[i] << " ";
        }
        cout << endl;

        return 0;
    }
```

輸出arr[i]

氣泡排序法

```
                    ] > arr[j + 1]){
                    arr[j];
                    ] = arr[j+1];
                    arr[j+1] = tmp;
                }
            }
        }

        for (int i = 0; i < arr_length; i++){
            cout << arr[i] << " ";
        }
        cout << endl;

        return 0;
    }
```

換行

氣泡排序法

```
                    ] > arr[j + 1]){
                    arr[j];
                    ] = arr[j+1];
                    arr[j+1] = tmp;
                }
            }
        }

        for (int i = 0; i < arr_length; i++){
            cout << arr[i] << " ";
        }
        cout << endl;

        return 0;
    }
```

如此一來，就完成了陣列的輸出

氣泡排序法

```
] > arr[j + 1]){
arr[j];
] = arr[j+1];
arr[j+1] = tmp;
}

for
}
cou

ret
}
```

bubble_sort.exe

```
1 2 3 4 5

Process returned 0 (0x0)   execution time : 0.028 s

Press any key to continue.
```

的輸出

氣泡排序法

```
] > arr[j + 1]){
arr[j];
] = arr[j+1];
arr[j+1] = tmp;
}

for
}
cou

ret
}
```

bubble_sort.exe

```
1 2 3 4 5

Process returned 0 (0x0)   execution time : 0.028 s

Press any key to continue.
```

如果想要由大到小排列該怎麼做呢？

的輸出

氣泡排序法

```
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
18      for (int i = 0; i < arr_length; i++){
```

改成 <

氣泡排序法

```
4   int main()
5   {
6       int arr[5] = {4, 1, 5, 3, 2}, arr_length = 5, tmp;
7
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] < arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
16      }
17
18      for (int i = 0; i < arr_length; i++){
```

改成 <

氣泡排序法

```
4   int m
5   {
6
7
8
9
10
11
12
13
14
15
16
17
18      for (int i = 0; i < arr.length; i++){
```

bubble_sort.exe

```
5 4 3 2 1

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

氣泡排序法

氣泡排序法可以優化：

- 若每次循環中沒有資料進行交換，則表示已完成排序，可以直接跳出迴圈。

氣泡排序法

假設要排N筆資料

- 排序循環N-1次

```
8       for (int i = 0; i < arr_length - 1; i++){
9           for (int j = 0; j < arr_length - i - 1; j++){
10              if (arr[j] > arr[j + 1]){
11                  tmp = arr[j];
12                  arr[j] = arr[j+1];
13                  arr[j+1] = tmp;
14              }
15          }
```

氣泡排序法

假設要排N筆資料

- 比較N*(N-1)/2次

第1次排序循環，要比較N-1次
第2次排序循環，要比較N-2次

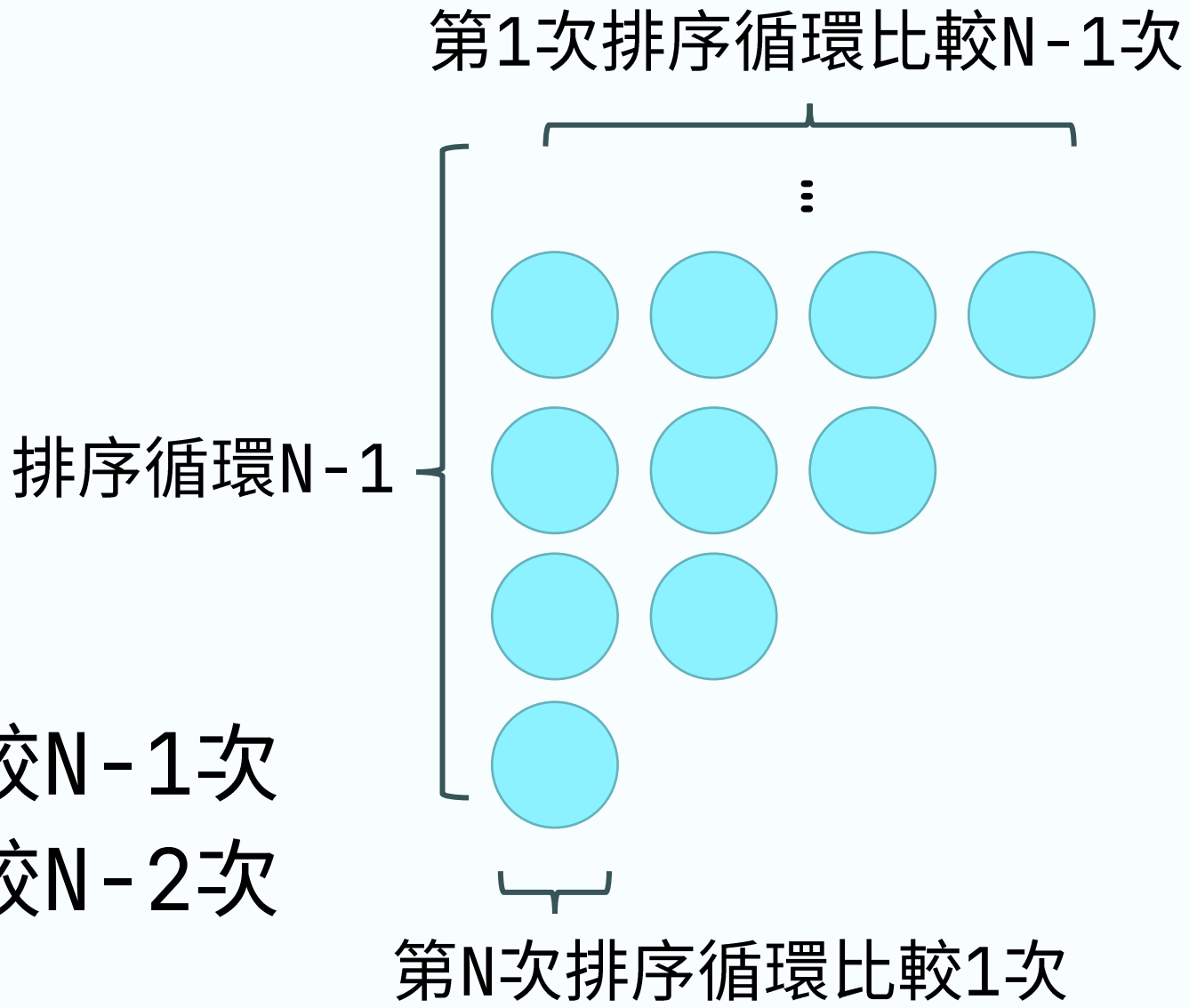
……
第N次排序循環，要比較1次

氣泡排序法

假設要排N筆資料

- 比較N*(N-1)/2次

第1次排序循環，要比較N-1次
第2次排序循環，要比較N-2次
……
第N次排序循環，要比較1次



可使用梯形公式計算

氣泡排序法

假設要排N筆資料

- 排序循環N-1次
- 比較N*(N-1)/2次

每次排列都需要這麼麻煩嗎？

<algorithm>

- C++標準函式庫<algorithm>
- 包含許多常用演算法(如：最大公因數、最小公倍數、最大值、最小值、交換、排序……等)

<algorithm>

- C++標準函式庫<algorithm>
- 包含許多常用演算法(如：最大公因數、最小公倍數、最大值、最小值、交換、排序……等)

<algorithm> sort

- sort(first, last);
- 將first ~ last - 1之間的元素由小到大排列

```
int arr[5] = {4, 1, 5, 3, 2};
sort(arr, arr + 5);
```

即可將arr陣列從第0~4格排列

<algorithm> sort

- sort(first, last, cmp);
- 將first ~ last - 1之間的元素依照自訂比照函式排列

```
bool cmp(int a, int b){
    return a > b;
}

int arr[5] = {4, 1, 5, 3, 2};
sort(arr, arr + 5, cmp);
```

如果a>b回傳true，反之false

即可將arr陣列從第0~4格大到小排列

<algorithm> stable_sort

- `stable_sort(first, last);`
- 將first ~ last - 1之間的元素由小到大排列
- 保證相同數值的元素，在排序後關係不變