STL

temmie

前言

STL 課程時間較少，部分細節我會跳過。

有任何問題請盡快提出！

Slido 連結

詹凱智（temmie）

- 個人簡介
  - 就讀北市立成淵高中
  - 特殊選才錄取清大資工系
  - Discord：temmieowo
- 競賽 / 社群經驗
  - APCS 實作 5、觀念 4 級分
  - 北市資訊能力競賽三等獎
  - 北臺灣學生資訊社群負責人

什麼是 STL？

一堆好用的**容器**跟**函式**。

注意事項

- 內建工具非常方便，熟練後的 CP 值很高。
- 資料結構有好的性質，在之後的課程中會用到。
- 內建工具固然方便，但仍然還是要了解一些東西的原理。
- STL 很多細節，記得多練習，才不會忘記用 or 在賽中踩到坑。
- 部分練習題的題單

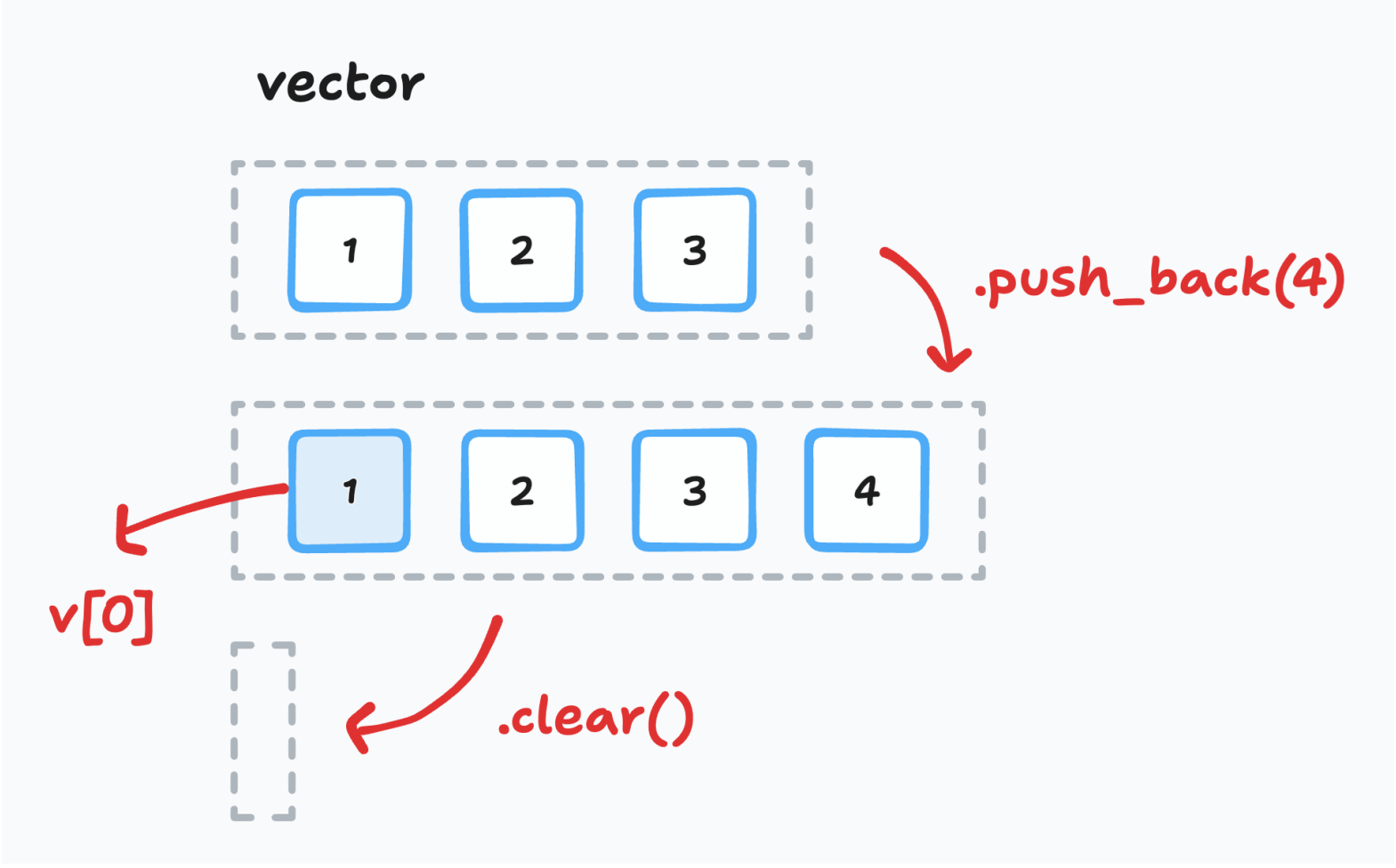
```
container<type> name
```

- container：STL 容器名稱
- type：容器內元素的型別
- name：該容器的名稱

```
name.method(arg...)
```

- method：函式名稱
- arg：函式參數

vector



可以動態改變大小的陣列。

```
container<type> name

vector<int> v(size, value);
```

- `size`：陣列的初始大小
- `value`：陣列的初始值

```cpp
vector<int> v1; // v1 = {} 空的陣列
vector<char> v2; // v2 = {} 空的陣列
vector<int> v3(5); // v3 = {0, 0, 0, 0, 0}
vector<int> v4(3, 17); // v4 = {17, 17, 17}
vector<int> v5 = {4, 8, 7};
```

```
name.method(arg...)

vector.method(arg...)
```

- `.push_back(val)`：將 val 加進陣列的最後，$O(1)$
- `.pop_back()`：將陣列最後的元素刪除，$O(1)$
- `.size()`：回傳陣列的大小**（無號整數）**，$O(1)$
- `.empty()`：回傳陣列是否為空，$O(1)$
- `.clear()`：將陣列清空，$O(size)$
- `.resize(size, value)`：

  將陣列的大小設為 size，**多出來的部分設為** val，$O(\Delta size)$

```cpp
vector<int> v; // v = {}
v.push_back(5); // v = {5}
v.push_back(2); // v = {5, 2}

v.resize(5, 3); // v = {5, 2, 3, 3, 3}
v.size(); // 5
v.empty(); // false

v.pop_back(); // v = {5, 2, 3, 3}
v.clear(); // v = {}
```

大多 STL 容器支援一個特殊的遍歷方法：Range-based for loop

```
1 vector<int> v = {4, 8, 7, 6, 3}
2 for (int i=0 ; i<5 ; i++){
3     cout << v[i] << " "; // 4 8 7 6 3
4 }
5
6 for (int x : v){
7     cout << x << " "; // 4 8 7 6 3
8 }
```

同時也可以透過 & 修改數值

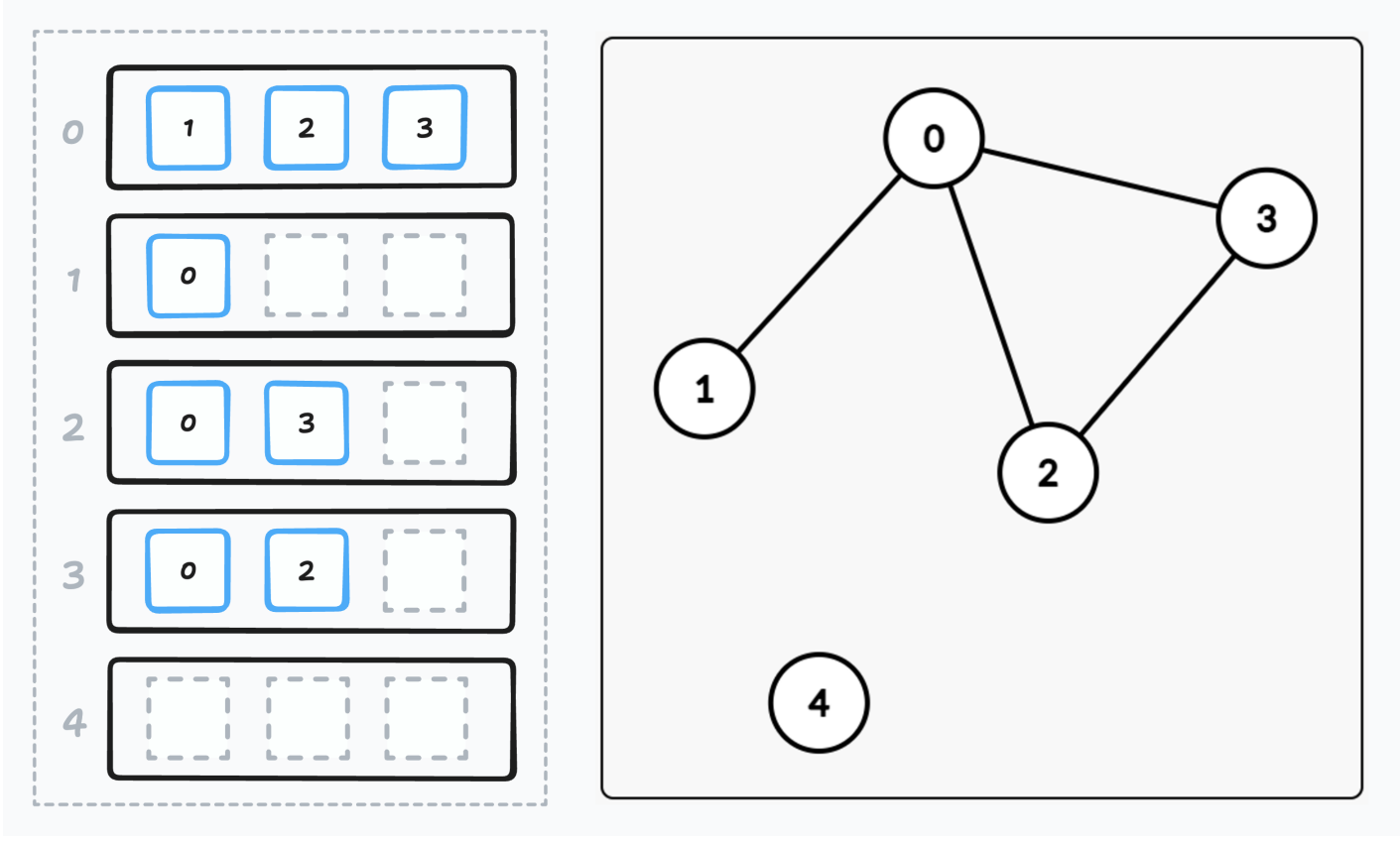
```
vector<int> v = {4, 8, 7, 6, 3}
for (int i=0 ; i<5 ; i++){
    v[i] = 2
}
// v = {2, 2, 2, 2, 2}

for (int &x : v){
    x = 5
}
// v = {5, 5, 5, 5, 5}
```

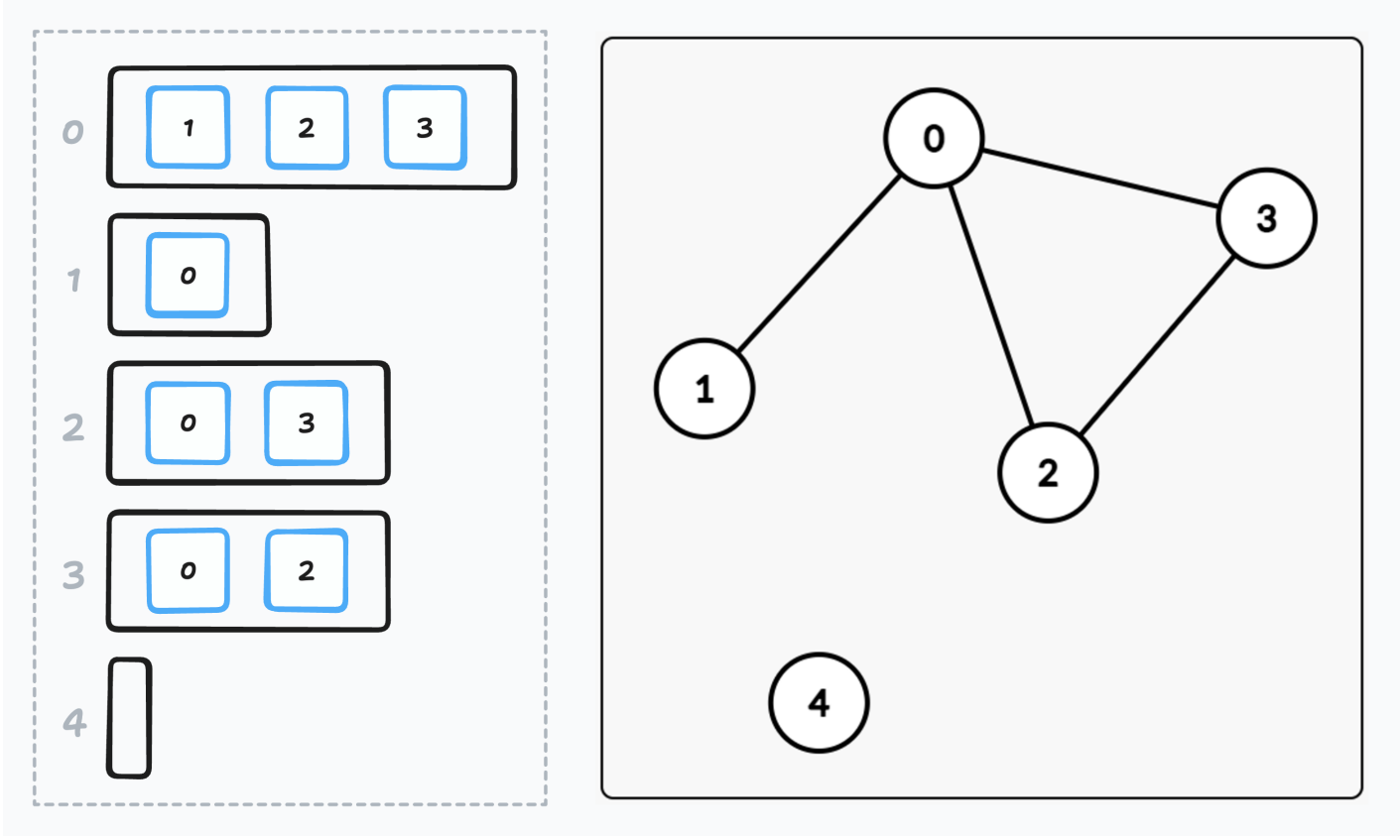
練習題 pA. 一維 vector 練習

給一個長度為 N (N 為偶數) 的陣列，請依序輸出該陣列奇數的項以及偶數的項。

二維 vector



用普通的矩陣會浪費很多空間



當矩陣的每一行有不同數量的元素，就是二維 vector 的出場時機

```
container<type> name

vector<vector<type>> name(row_size, row_val)

row_val ⟹ vector<type>(column_size, val)
```

- row_size：行數
- column_size：列數
- val：初始值

```cpp
vector<vector<int>> v1; // v1 = {} 空的矩陣

vector<vector<int>> v2(3);
// v2 = { {}, {}, {} } 矩陣內有三個空的陣列

vector<vector<int>> v3(3, vector<int>(2));
// v3 = { {0, 0}, {0, 0}, {0, 0} }

vector<vector<int>> v4(3, vector<int>(2, 5));
// v4 = { {5, 5}, {5, 5}, {5, 5} }
```

```cpp
vector<vector<int>> v(3); // v = { {}, {}, {} }
v[1].push_back(1); // v = { {}, {1}, {} }
v.push_back({3, 4}); // v = { {}, {1}, {}, {3, 4} }
```

練習題 pB. 二維 vector 練習

給一個長度為 $N \times M$ 的矩陣，以及一倍率 k，請輸出矩陣每個數乘上 k 的結果。

練習題 pC. 朋友關係

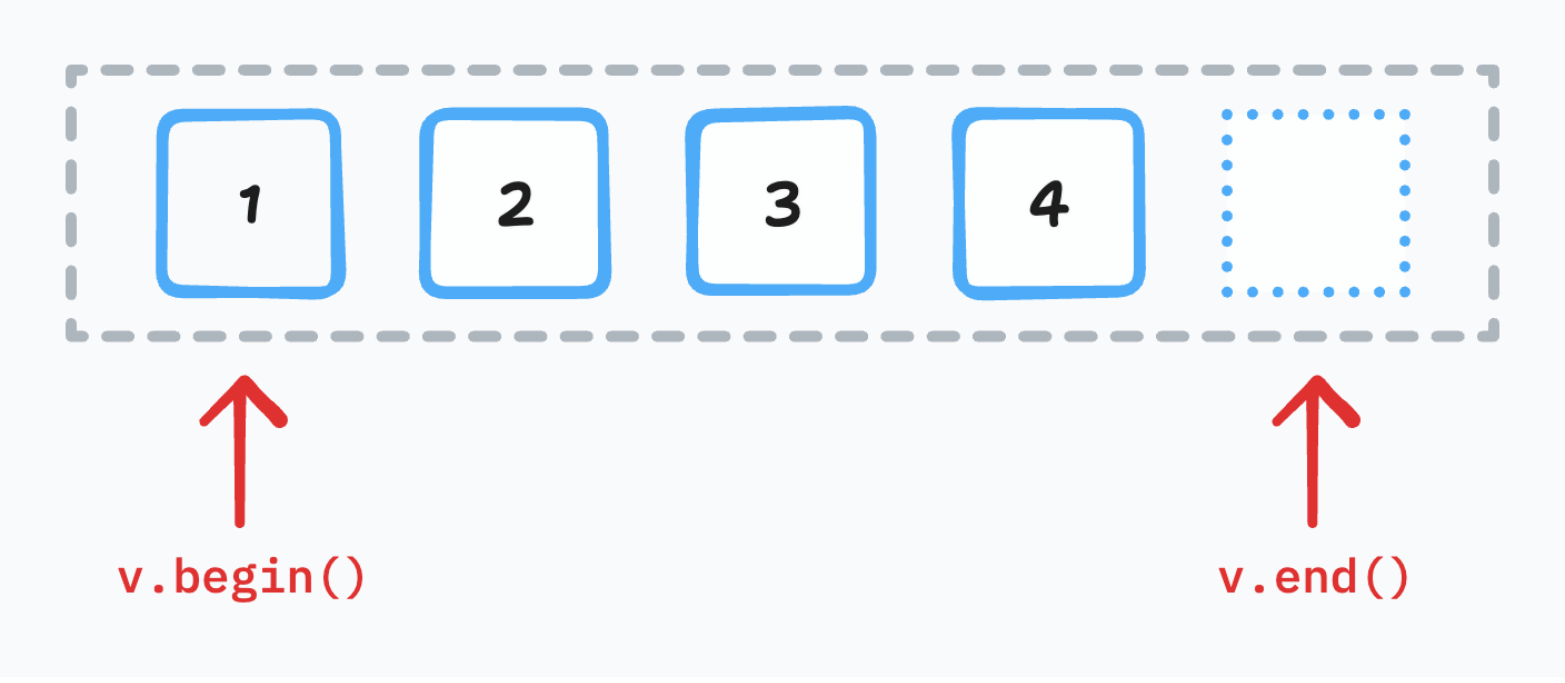
給 N 個人，以及 M 個朋友關係，每個朋友關係代表 a_i 與 b_i **彼此**是朋友，請對於每個人 $0 \sim n-1$，先輸出一數 k_i 代表他的朋友人數，接下來 k_i 個數代表他所有的朋友。

iterator

iterator（迭代器）

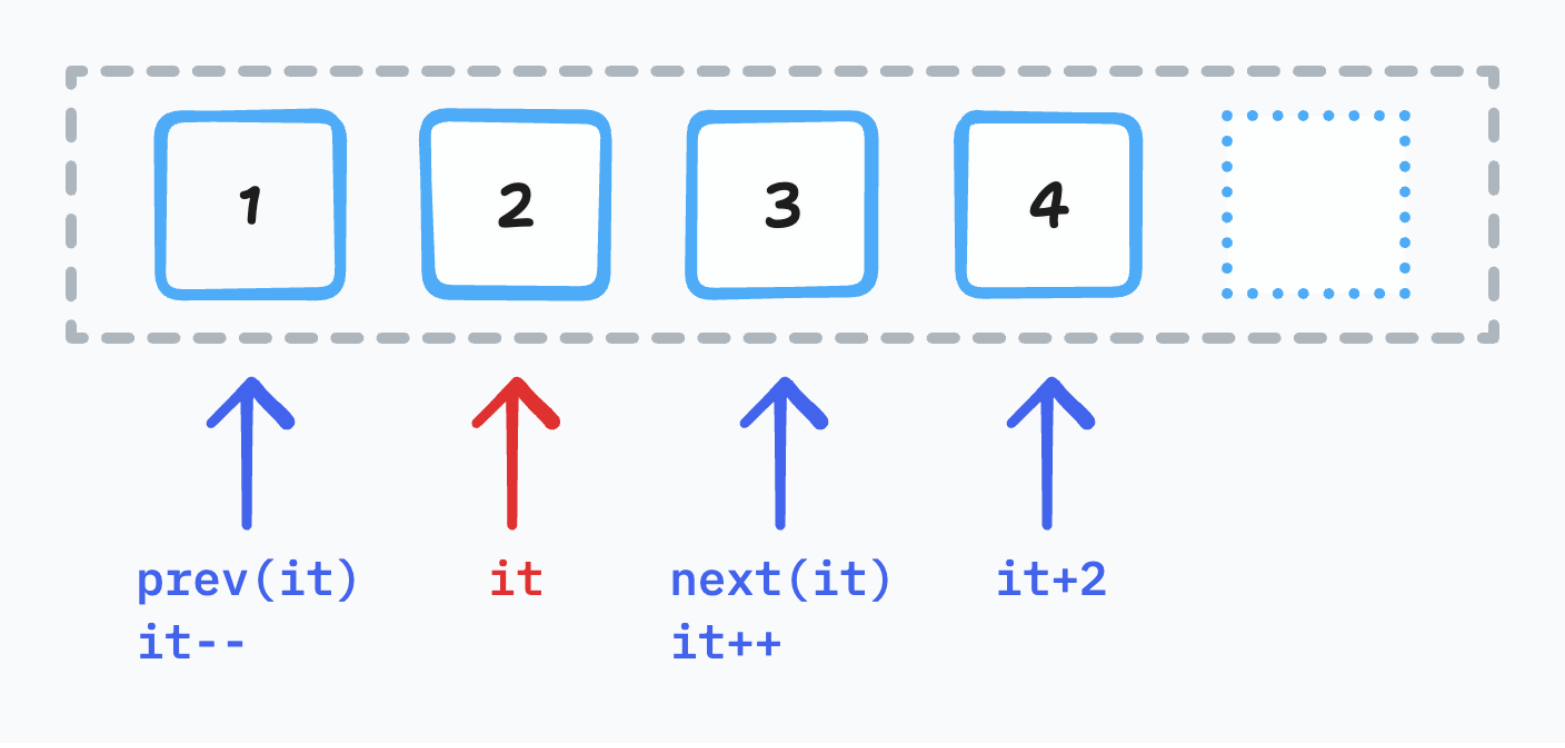
概念：STL 容器中的**指標**。

1. 容器的開頭與結尾



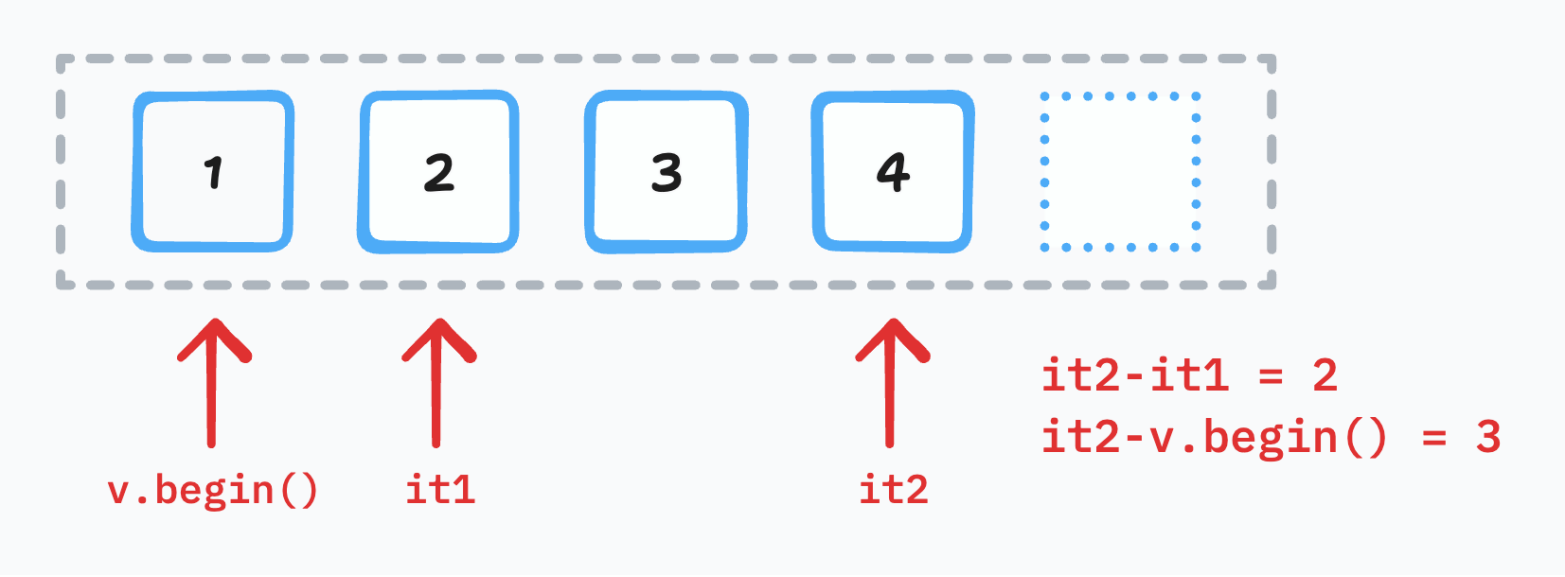
待會只要沒有特別註明，**開頭 (L)** 與**結尾 (R)** 都是左閉右開形式。

2. 左右移動



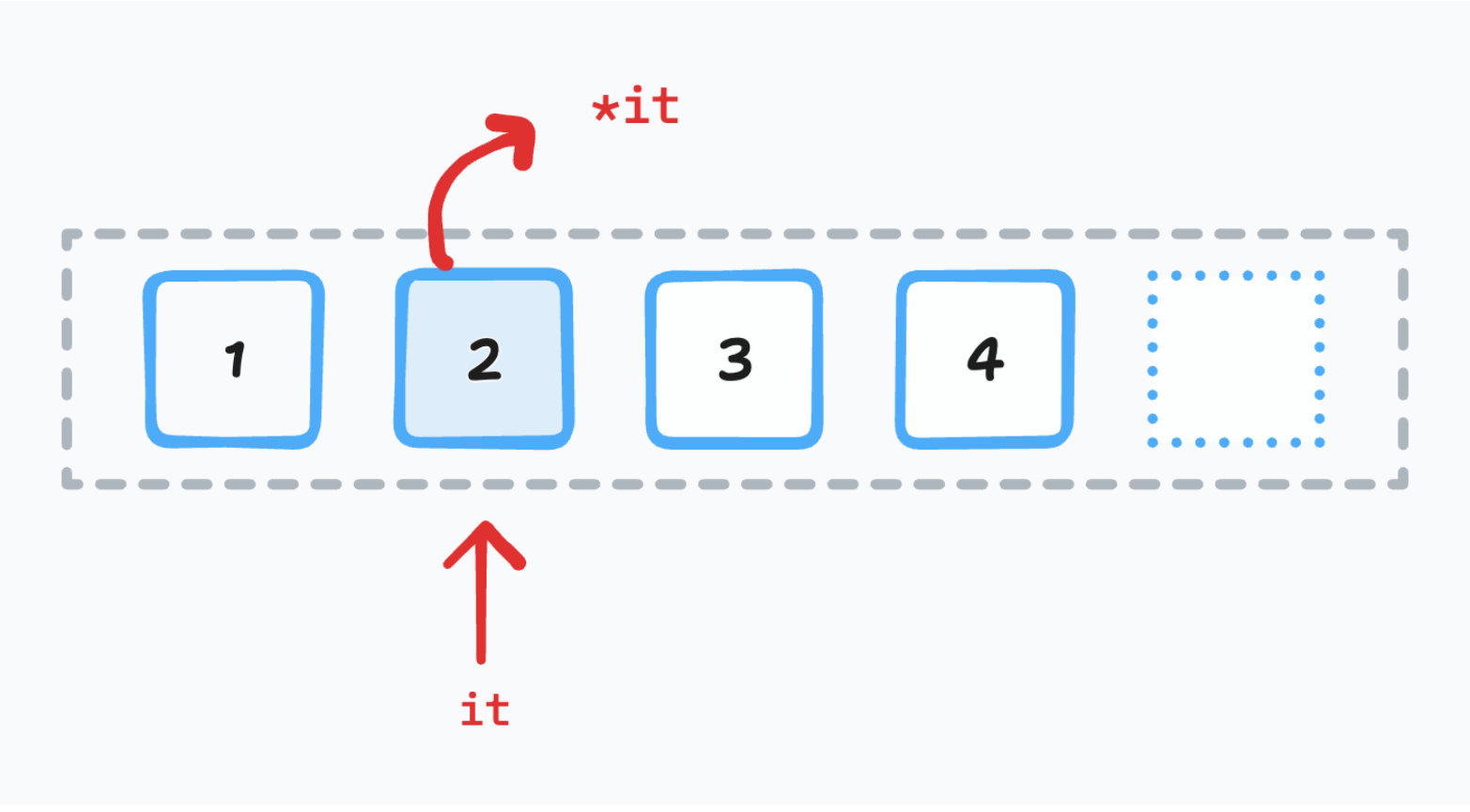
某些容器沒辦法這樣使用，之後會註明。

3. 計算距離



某些容器沒辦法這樣使用，之後會註明。

4. 存取 / 修改內容



`*it` 就是指向的元素，修改它就可以修改元素。

指標有很多型別（e.g. `int *`），iterator 也有。

`container<type>::iterator name=value;`

```
1 vector<int> v1;
2 vector<int>::iterator it1; // vector<int> 的 iterator
3 vector<int>::iterator it2 = v1.begin();
4
5 vector<int> v2;
6 auto it3 = v2.begin(); // 一定要丟初始值
```

內建函式

`function(L, R, arg)`

用 `sort(L, R)` 以及 `vector<int> v={4, 8, 7, 6, 3}` 當例子。

- `sort(`L`, `R`)`：將陣列的 $[L, R)$ 由小到大排序
- `sort(v.begin(), v.end())`：`[3, 4, 6, 7, 8]`
- `sort(v.begin(), v.begin()+3)`：`[4, 7, 8, 6, 3]`
- `sort(v.begin()+3, v.begin())`：RE

`function(L, R, arg)`，作用於 $[L, R)$

- `sort(L, R)`：將陣列從小到大排序，$O(n \log n)$
- `sort(L, R, greater<int>())`：將陣列從大到小排序
- `reverse(L, R)`：將陣列反轉，$O(n)$
- `fill(L, R, val)`：將陣列所有元素設為 `val`，$O(n)$

`function(L, R, arg)`，作用於 $[L, R)$

- `min_element(L, R)`：陣列最**小**元素的 iterator，$O(n)$
- `max_element(L, R)`：陣列最**大**元素的 iterator，$O(n)$
- `binary_search(L, R, val)`：

  要先排序，回傳是否有找到 `val`，$O(\log n)$
- `lower_bound(L, R, val)`：

  要先排序，陣列數值**大於等於** `val` 的 iterator，$O(\log n)$
- `upper_bound(L, R, val)`：

  要先排序，陣列數值**大於** `val` 的 iterator，$O(\log n)$

練習題 pD. 陣列反轉

給一個長度為 N 的陣列，請輸出該陣列反轉的結果。

練習題 pE. 字母排序

給 $N(1 \leq N \leq 10^5)$ 個字串 $S_i(1 \leq |S_i \leq 20)$，請輸出他們按照字典序排序的結果。

練習題 pF. 迴文數

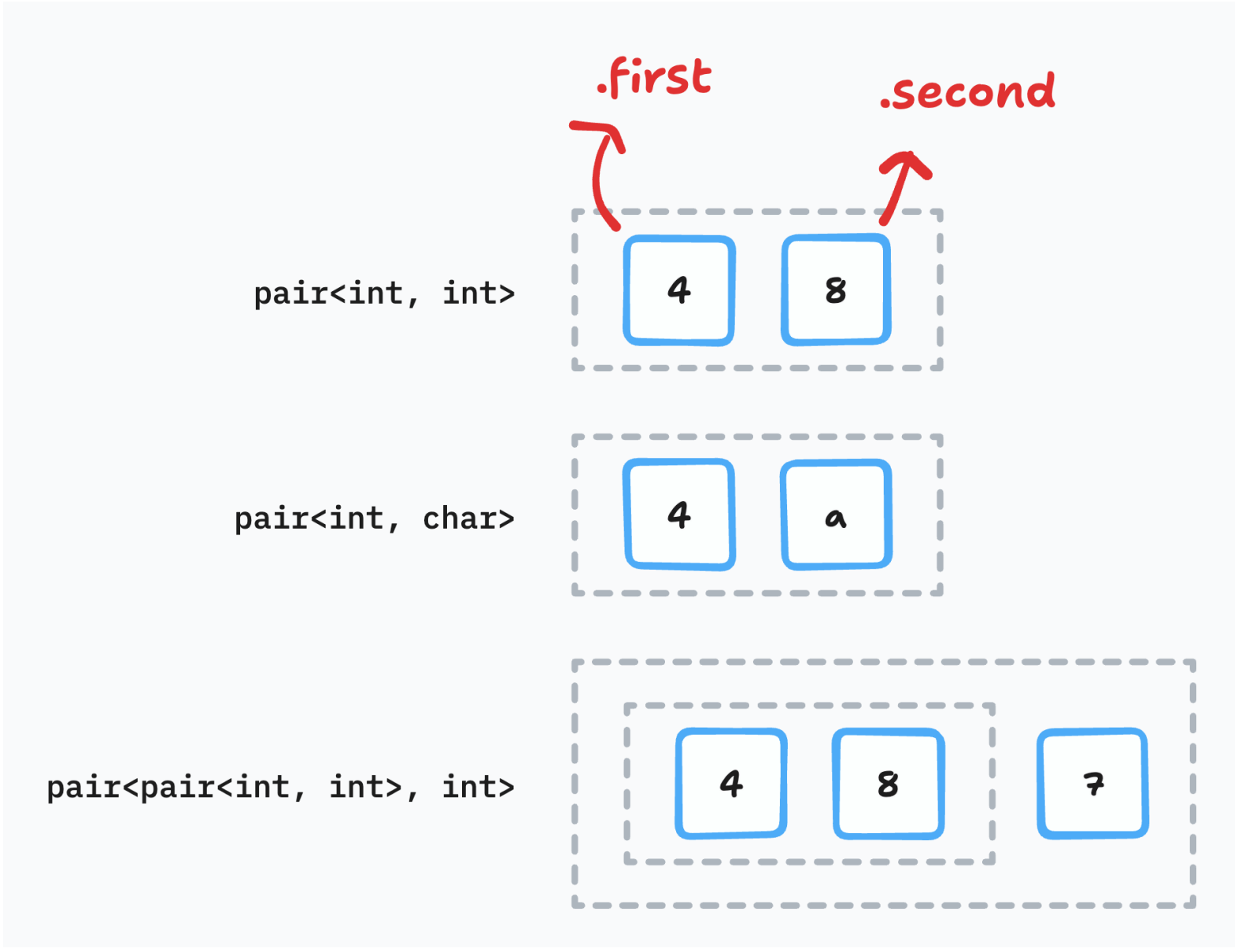
給一個數字 N，保證不會有零出現，一數被稱作**迴文數**，若他從左到右看跟從右到左看都相同，例如 12321 跟 999 都是迴文數，123 跟 554 則不是，請你判斷 n 是否為迴文數。

包裝容器

包裝容器

本章節會介紹 pair

「包裝容器」是我自己創的名詞，概念是將**多個元素包裝**起來。


.first
.second
pair<int, int>
4
8
pair<int, char>
4
a
pair<pair<int, int>, int>
4
8
7

`pair<type1, type2> name`

- `type`：元素型別（可以不同）

- `name.first`：取得前面的元素
- `name.second`：取得後面的元素

```cpp
pair<int, int> p1; // p1 = { , } 空的 pair
pair<int, int> p2(4, 8); // p2 = {4, 8}
pair<int, int> p3 = {4, 8};
p3.first; // 4
p3.second; // 8

pair<pair<int, int>, int> p4 = {{4, 8}, 7};
p4.first; // {4, 8}
p4.first.first; // 4
p4.second; // 7
```

練習題 pG. 星星排序-1

給 N $(1 \le N \le 10^5)$ 個座標點 $(x_i,\ y_i)$，排序方式為「x 比較小的在前，如果一樣則 y 比較小的在前」，輸出排好序的座標點。

練習題 pH. 星星排序-2

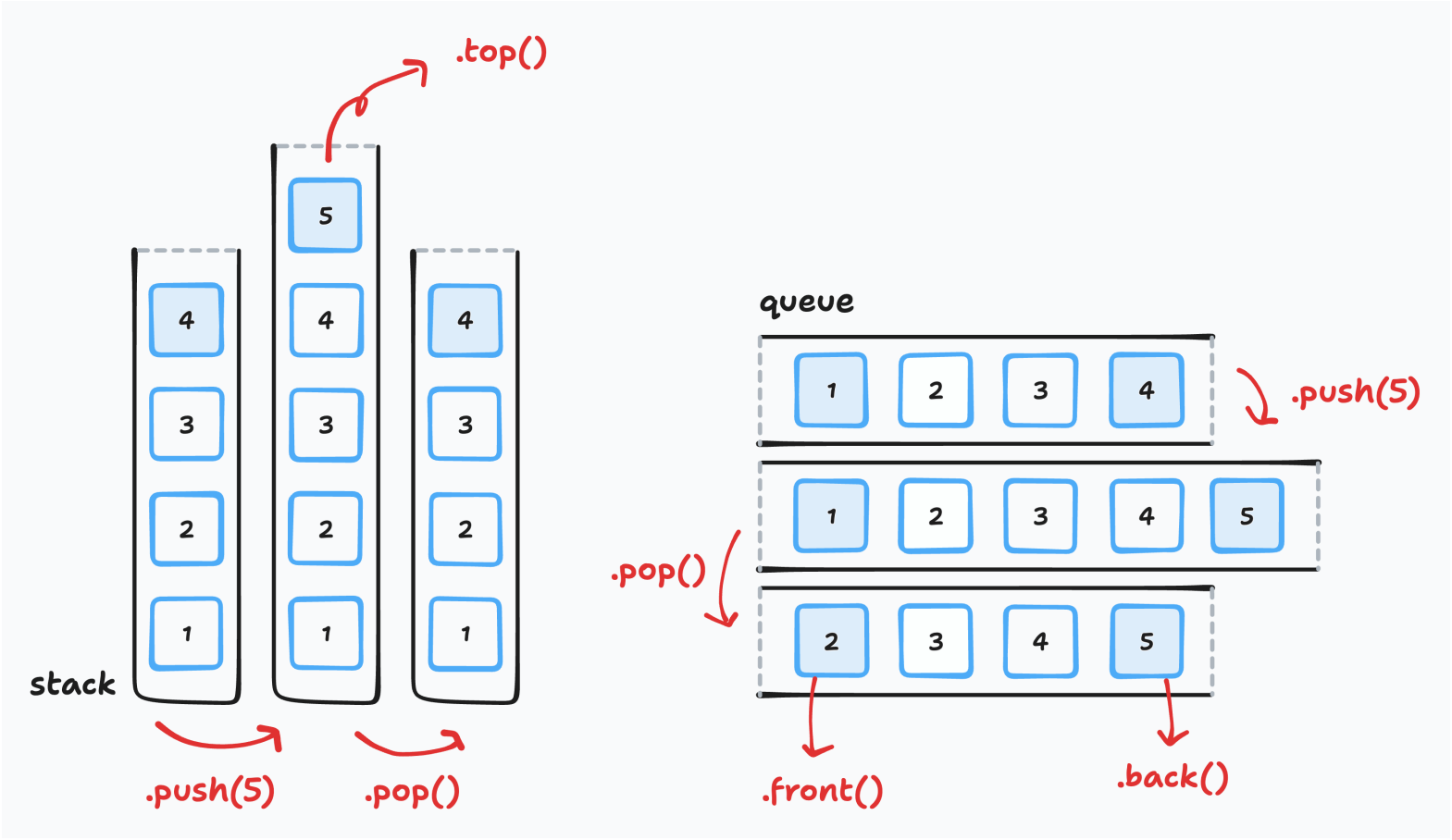
給 N $(1 \le N \le 10^5)$ 個座標點 $(x_i,\ y_i)$，排序方式為「x 比較小的在前，如果一樣則 y 比較**大**的在前」，輸出排好序的座標點。

線性容器與 pq

線性容器與 pq

本章節會介紹 `stack`、`queue` 與 `priority_queue`

他們特點在於**加入與刪除元素**的方式有所限制。


.top()
5
4
3
2
1
4
3
2
1
4
3
2
1
stack
.push(5)
.pop()
queue
1
2
3
4
.push(5)
1
2
3
4
5
.pop()
2
3
4
5
.front()
.back()

通用函式

- `.size()`
- `.empty()`
- 沒有 iterator（也就是沒有 `.begin()`、`.end()`）

```
stack.method(arg...);
```

- `.push(val)`：將 val 加進 stack 的最上面，$O(1)$
- `.pop()`：將 stack 最上面的元素刪除，$O(1)$
- `.top()`：取得 stack 最上面的元素，$O(1)$

```
queue.method(arg...);
```

- `.push(val)`：將 val 加進 queue 的最後面，$O(1)$
- `.pop()`：將 queue 最前面的元素刪除，$O(1)$
- `.front()`：取得 queue 最前面的元素，$O(1)$
- `.back()`：取得 queue 最後面的元素，$O(1)$

使用時機：

stack

- 括號序列
- DFS
- 單調隊列
- 四則運算

queue

- BFS
- 單調佇列

練習題 pI. Stack 練習

模擬 stack 的常用操作。

ZJ b838. 括號問題

給 t $(1 \leq t < 1000)$ 個字串 S_i $(1 \leq |\ S_i\ | \leq 20)$，如果 S_i 是合法的括號序列就輸出括號的對數，否則輸出 0。

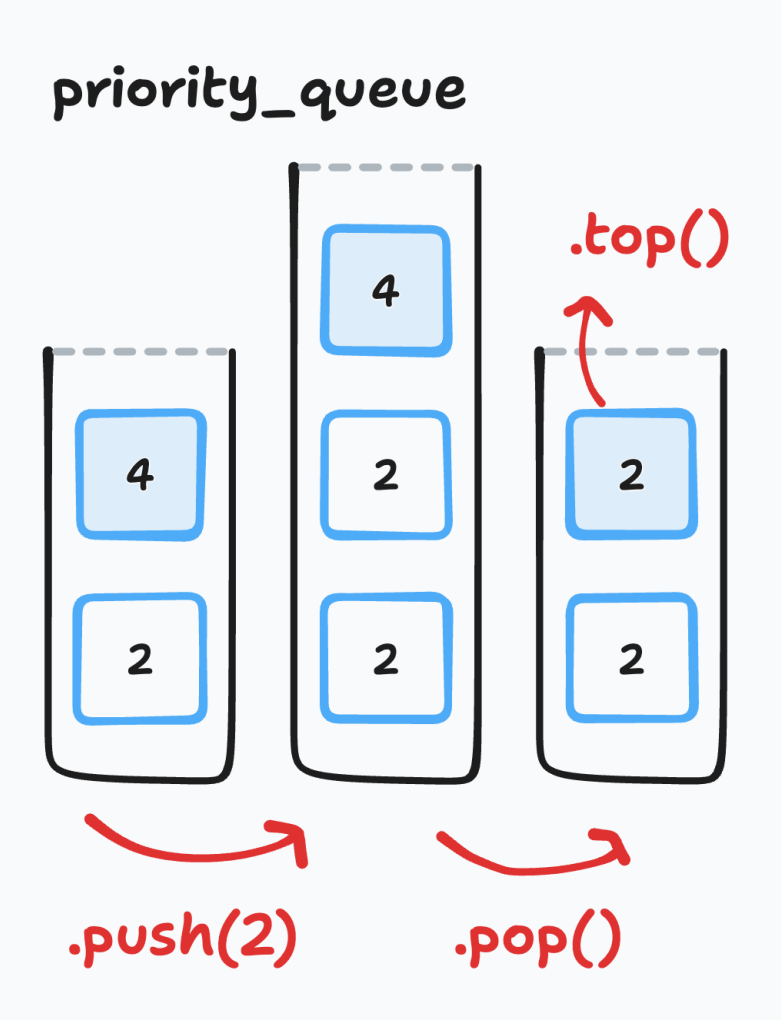
練習題 pJ. Queue 練習

模擬 queue 的常用操作。

ABC 247D. Cylinder

給 q $(1 \le q \le 2 \times 10^5)$ 筆查詢，輸入為兩種其中一個：

- $1\ x\ c$：將寫有 x 的 c 顆球加入 queue 最後面
- $2\ c$：將 queue 最前面的 c 顆球取出，並輸出數字總和



(這些結構本質上是樹狀的)

最小值在頂端的 pq：

```
priority_queue<type, vector<type>, greater<type>>;
priority_queue.method(arg...);
```

- `.push(val)`：將 val 加進 pq 裡面，$O(\log n)$
- `.pop()`：將 pq 最上面的元素刪除，$O(1)$
- `.top()`：取得 pq 最上面的元素，$O(1)$

使用時機：

priority_queue

- $10^6 \leq n \leq 10^7$ 的動態排序題
- 維護前 k 大的數
- 貪心法
- 最短路

練習題 pK. Priority Queue 練習

模擬 priority_queue 的常用操作。

ABC 217E. Sorting Queries

給 q $(1 \le q \le 2 \times 10^5)$ 筆查詢，輸入為三種其中一個：

- $1\ x\ c$：將寫有 x $(1 \le x \le 10^9)$ 的數字加入陣列最後面
- 2：輸出陣列最前面的數字，保證陣列一定會有數字
- 3：將陣列由小到大排序

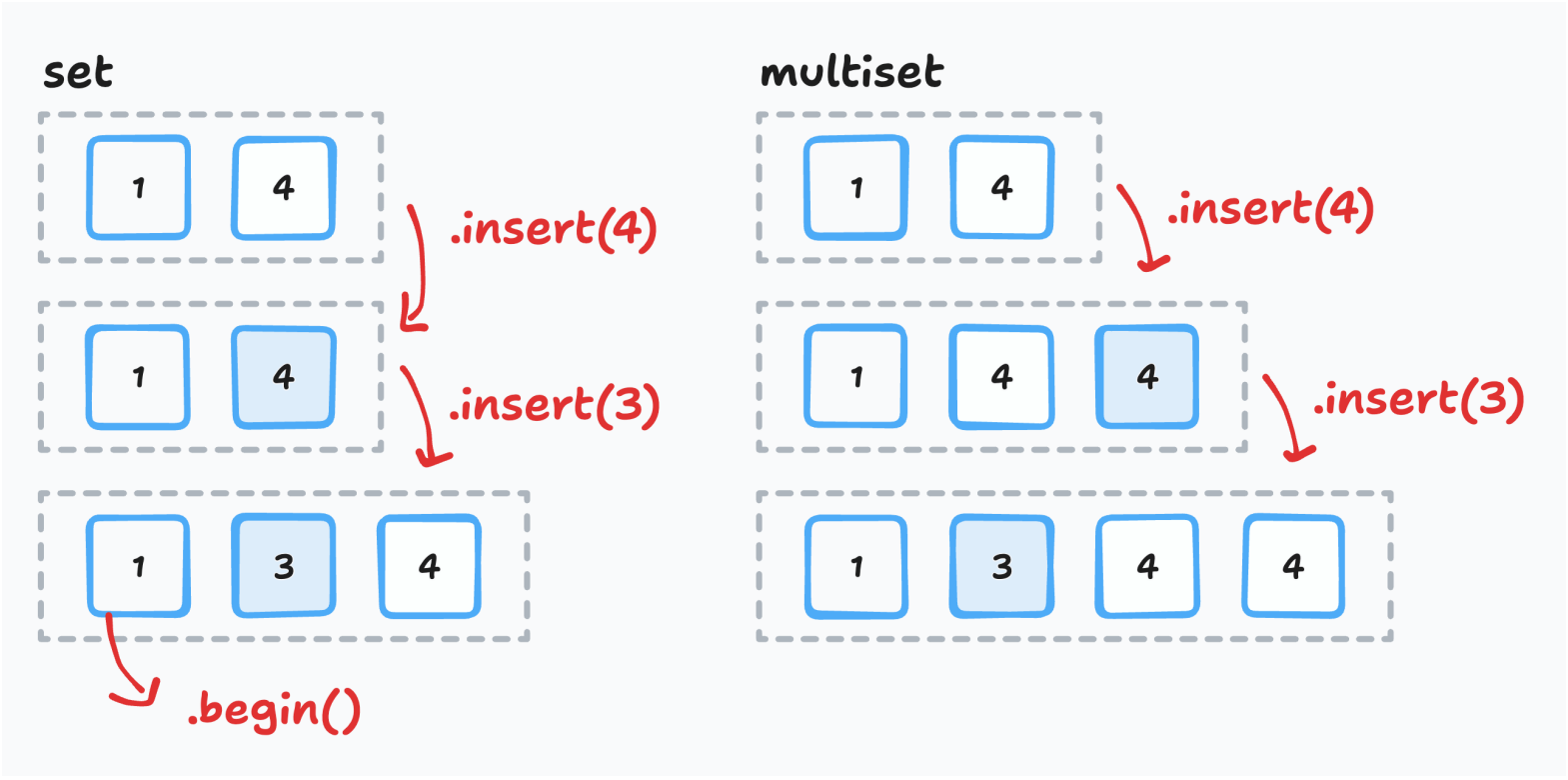
樹狀容器

樹狀容器

本章節會介紹 `set`、`multiset` 與 `map`

他們特點在於**容器內的元素都是排序的**，操作幾乎都是 $O(n \log n)$

請注意他們的差異，以及成員函式的回傳值



（這些結構本質上是樹狀的）

通用函式

- `.size()`
- `.empty()`
- `.begin()`
- `.end()`

雖然有支持迭代器，但不支援計算兩迭代器的距離。

```
set.method(arg...);
```

- `.insert(val)`：將 val 加進 set 內，$O(\log n)$
  - 回傳 `pair<it, bool>`，代表插入的位置、是否插入成功
- `.erase(iterator)`：將元素刪除，$O(1)$
  - 回傳 `it`，代表刪除元素的下一個位置
- `.erase(val)`：將元素刪除，$O(\log n)$
  - 回傳 `bool`，代表是否成功刪除
- `.count(val)`：尋找元素個數，$O(\log n)$
  - 回傳 `int`，代表 `val` 的數量，但實際上只會有 0 或 1

```
set.method(arg...);
```

- `.find(val)`：尋找 val 的位置，$O(\log n)$
- `.lower_bound(val)`：尋找不小於 val 的元素的位置，$O(\log n)$
- `.upper_bound(val)`：尋找不小於等於 val 的元素位置，$O(\log n)$

- 上面函式都回傳 `it`，如果有多個符合則回傳最前面的，否則回傳 `set.end()`

`multiset.method(arg...);`

- `.insert(val)`：將 val 加進 set 內，$O(\log n)$
  - 回傳 `it`，代表插入的位置（必定會成功）
- `.erase(iterator)`：將 iterator 指向的元素刪除
  - 回傳 `it`，代表刪除元素的下一個位置，$O(1)$
- `.erase(val)`：將所有值為 val 的元素刪除
  - 回傳 `int`，代表有多少元素被刪除，$O(k + \log n)$
  - 如果只想刪除一個值，應該用 `set.erase(set.find(val))`
- `.count(val)`：尋找元素個數，$O(k + \log n)$
  - 回傳 `val` 的數量，請注意時間複雜度與匹配到的數成正比

```
multiset.method(arg...);
```

- `.find(val)`：尋找 val 的位置，$O(\log n)$
- `.lower_bound(val)`：尋找不小於 val 的元素的位置，$O(\log n)$
- `.upper_bound(val)`：尋找不小於等於 val 的元素位置，$O(\log n)$

- 上面函式如果找到則回傳它的迭代器位置，如果有多個則回傳最前面的，否則回傳 `set.end()`

使用時機：

set / multiset

- 去重（set）
- $n \leq 10^6$ 的動態排序題
- 維護前 k 大的數
- 動態尋找排序陣列中，某元素的鄰居

練習題 pI. Set 練習

模擬 set 的常用操作。

CSES 1091. Concert Tickets

給 N $(1 \le N \le 2 \times 10^5)$ 張票以及 M $(1 \le M \le 2 \times 10^5)$ 位購票者，每張票有他的價錢 h_i，每位購票者也有預算 t_i，請**依序**輸出每位訂票者他們能買的最貴的票，並且該票之後**不能重複賣出**，若沒有則輸出 `-1`。

NHDK TPR 11. 藤原千花與字串

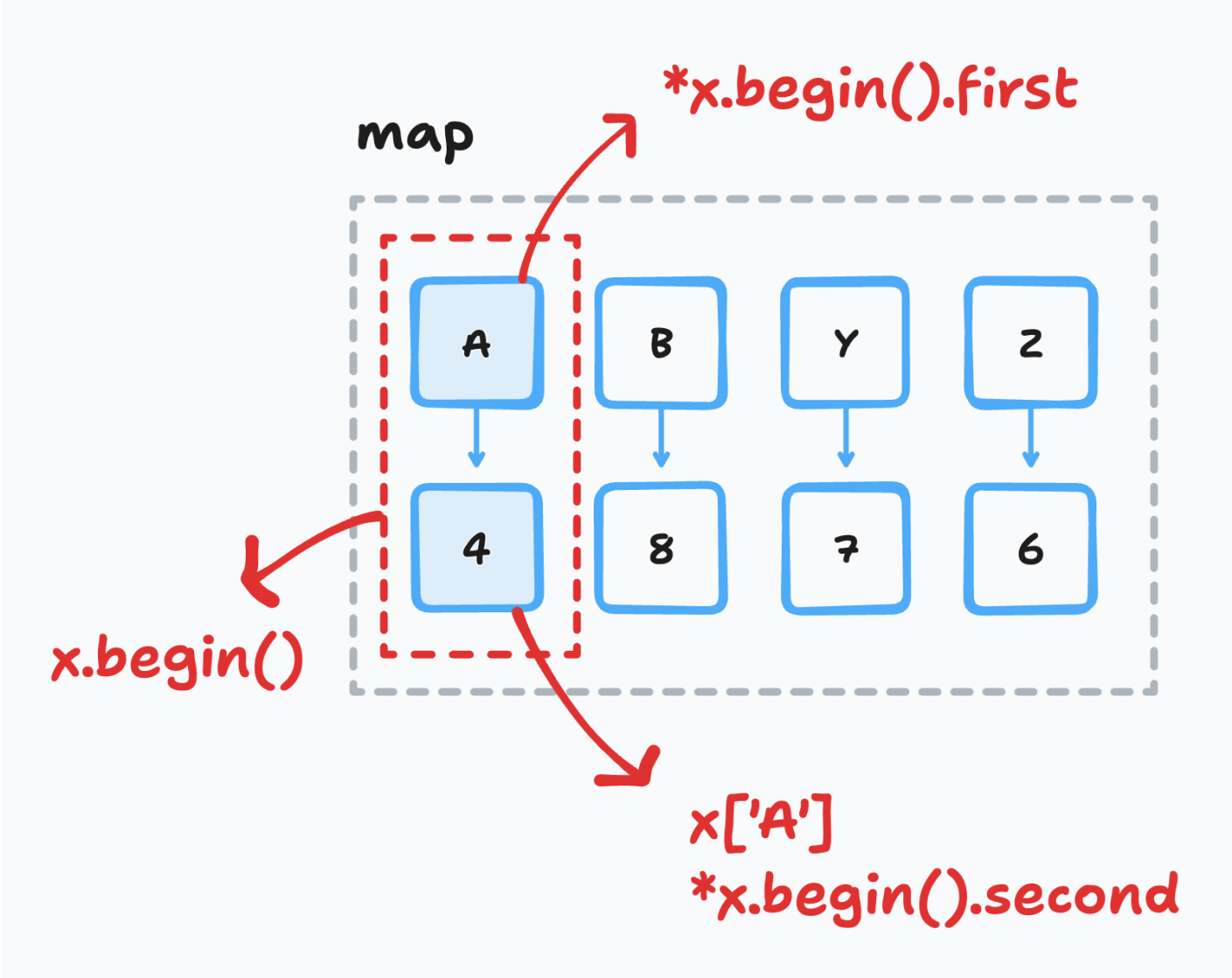
給 N $(1 \leq N \leq 2 \times 10^5)$ 個操作，每個操作會記錄一個字串

S_i $(1 \leq |\ S_i\ | \leq 10$，**可能重複** $)$，請對於每次操作輸出以前所有操作中，

字典序不超過目前字串中，字典序最大的字串。

ZJ f607. 切割費用

[現場 Demo]



(這些結構本質上是樹狀的)

```
map<key_type, value_type> ss;
```

- `ss[val1]` 把它想像成一個變數
  - 宣告：`ss[key]`
  - 賦值：`ss[key] = value`
  - 計算：`ss[key] += value`
  - 只要存取 `ss[key]` 就會多出一個元素

    (將增加 map 的儲存元素，以及查詢的時間複雜度)

假設 `it = map` 某個元素的迭代器

- `*it`：一個 `pair`
- `(*it).first`：該元素的 key
- `(*it).second`：該元素的 value

```
map.method(arg...);
```

- `.erase(iterator)`：將 iterator 指向的元素刪除
  - 回傳 `it`，代表刪除元素的下一個位置，$O(1)$
- `.erase(key)`：將索引為 key 的元素刪除
  - 回傳 `int`，代表有多少元素被刪除，$O(\log n)$
- `.find(key)`：尋找索引為 key 元素
  - 回傳 `it`，代表找到的位置，若找不到則回傳 `map.end()`

使用時機：

map

- 字串對應數字
- 索引值過大或有負數
- hash

ABC 261 C. NewFolder(1)

給 $N(1 \leq N \leq 2 \times 10^5)$ 個新增資料夾的操作，每次操作會有要新增的資料夾名稱 S_i，若該名稱的資料夾從未出現過，則該資料夾名稱為 S_i，否則該資料夾名稱為 $S_i(x)$，其中 x 為該資料夾在為新增前出現的次數。

補充

本次 STL 課程其實還少了一些有機會用到的容器，這部分就請學員自行研究 >_<

- deque
- list
- unordered 系列
- bitset

各位可以參考的學習資料

- 從零到一：那些演算法競賽會用到的基礎語法
- Cplusplus.com