Data Structure

Lecture by nella17
Credit by casperwang, yp155136
2023 / 03 / 04

Sprout

課程內容

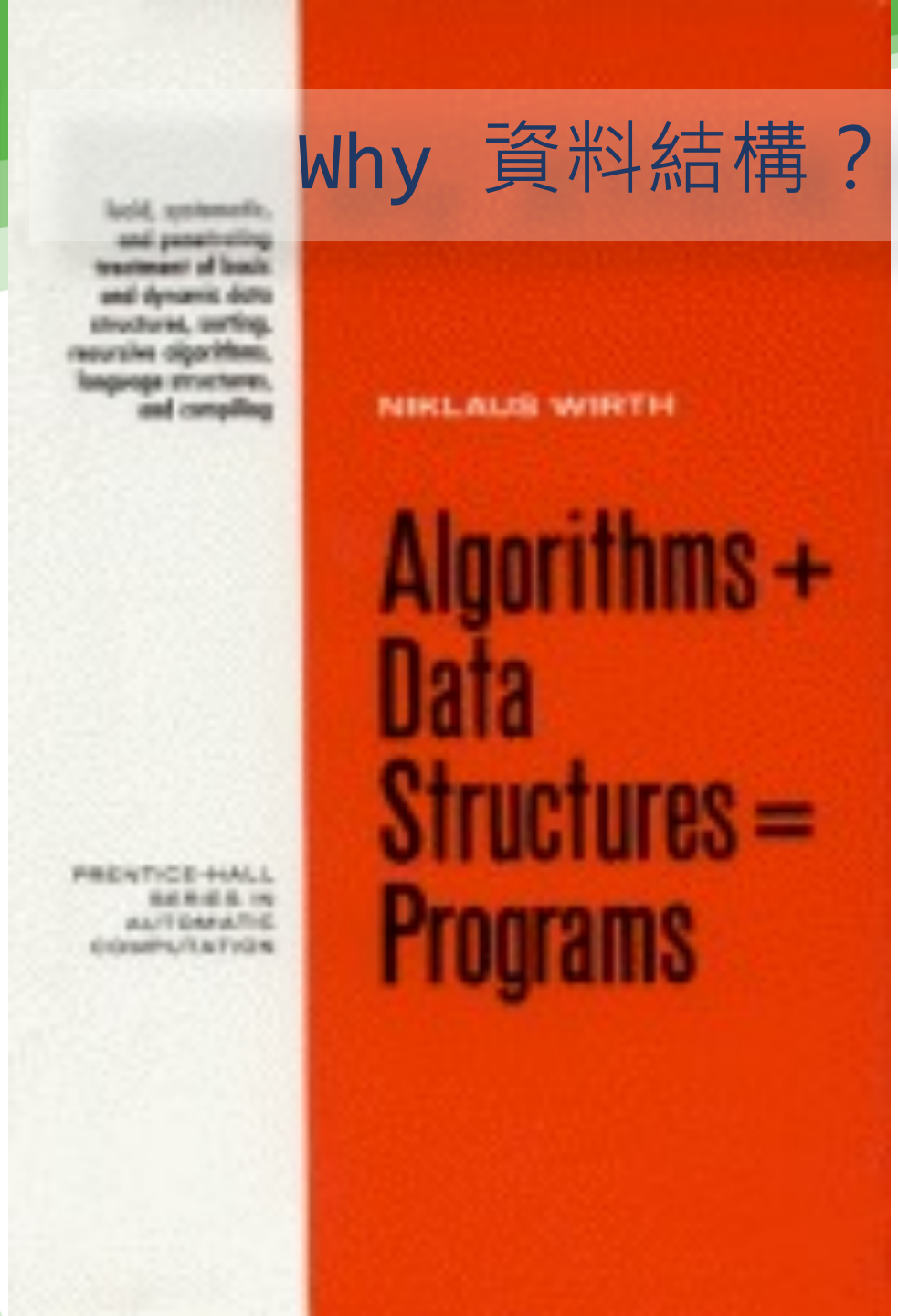
- 什麼是資料結構？
- Stack
- Queue
- Deque
- Linked-list
- 例題討論

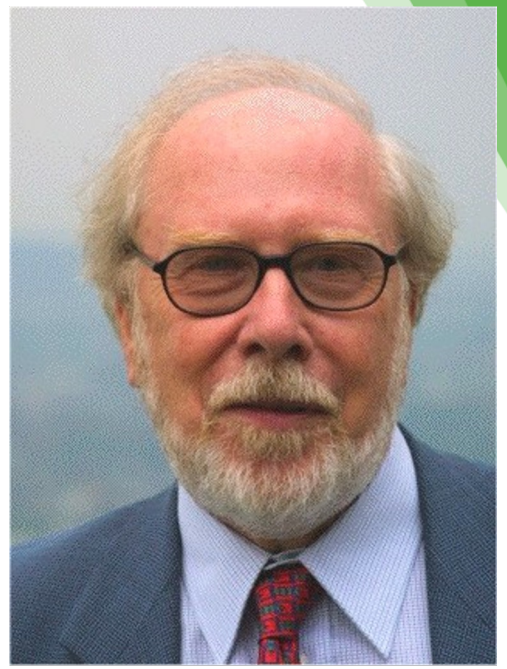
什麼是資料結構？

資料結構

- 是電腦中儲存、組織資料的方式
- 好的資料處理方式，能讓程式節省時間與空間
  - 操作：查詢、修改
- 例如：「陣列」就是一種資料結構

Why 資料結構？





Niklaus Wirth

the designer of Pascal

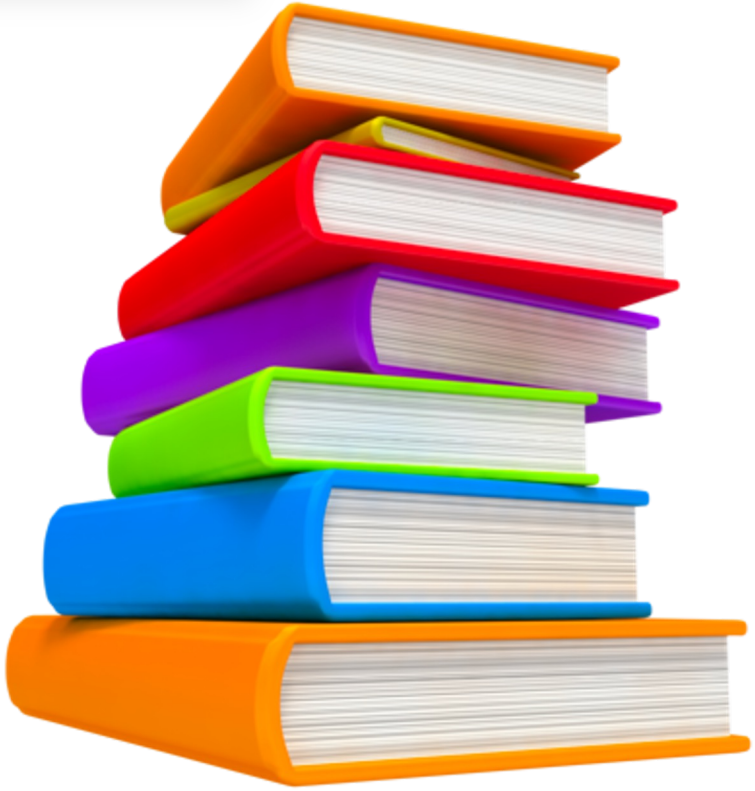
常見的資料結構

- Stack, Queue, Deque
- Linked-list
- Heap
- Set, Map
  - Self-balanced Binary Search Tree
- Disjoint Set
- BIT, Segment Tree, Sparse Table
- ……

Stack

Stack

- 要怎麼拿到紫色的那本書？



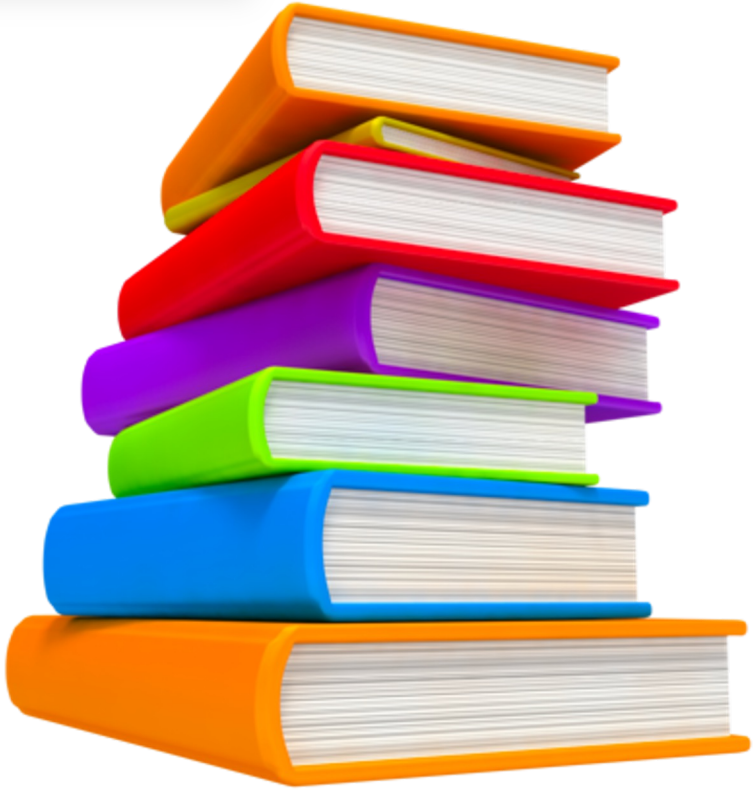
Stack

- 要怎麼拿到紫色的那本書？

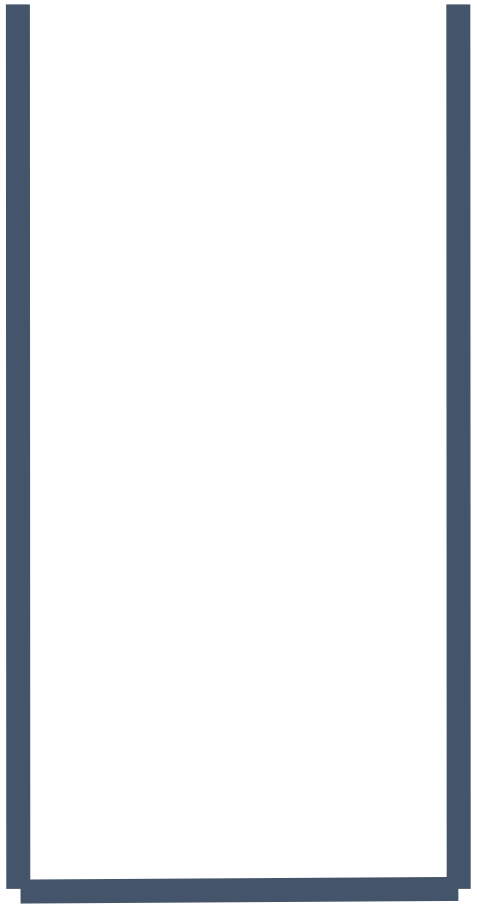
先依序把橘、黃、紅的書拿起來

拿到紫色的書

再依序將紅、黃、橘的書放回去

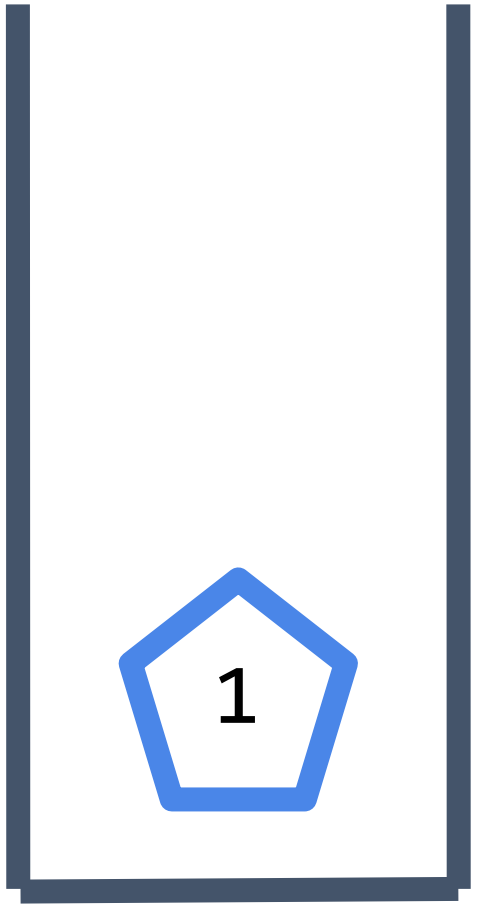


Stack（堆疊）



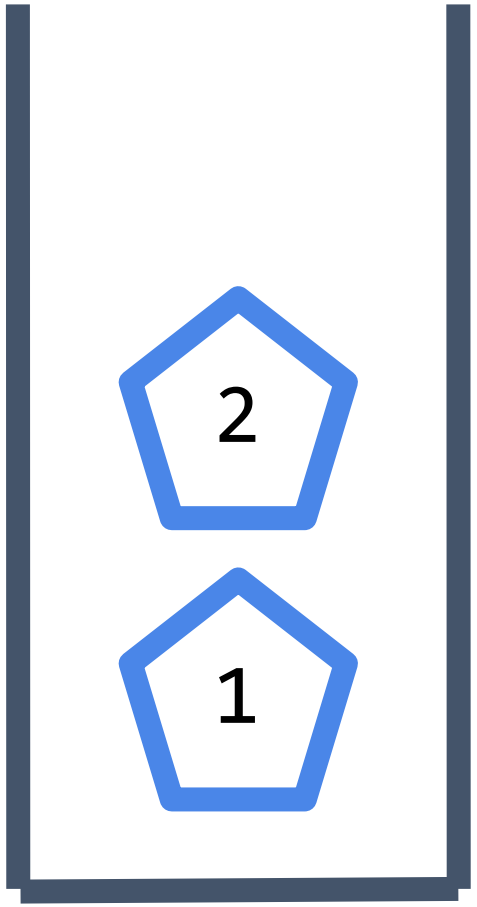
Empty

Stack（堆疊）



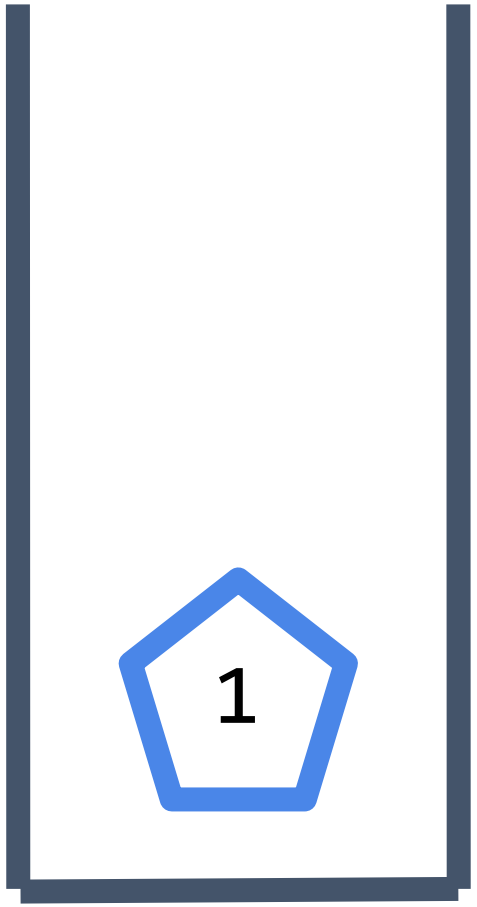
加入新資料 1

Stack（堆疊）



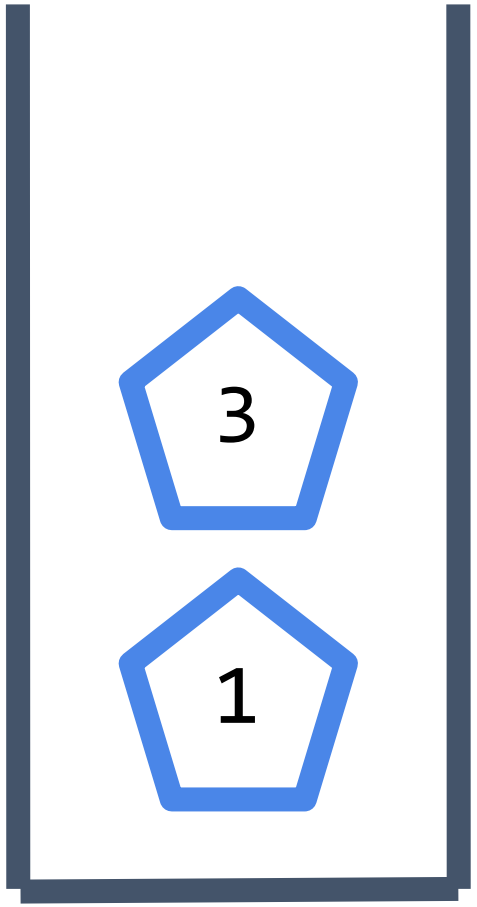
加入新資料 2

Stack（堆疊）



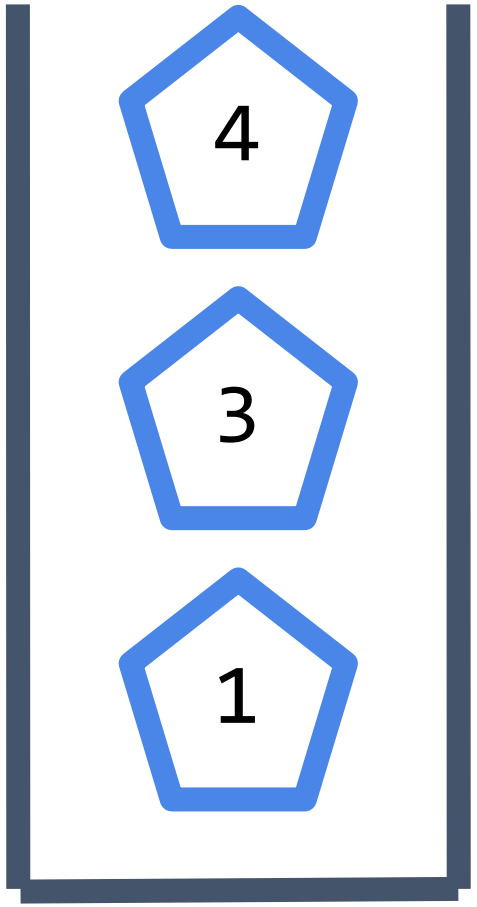
刪除最頂端資料

Stack（堆疊）



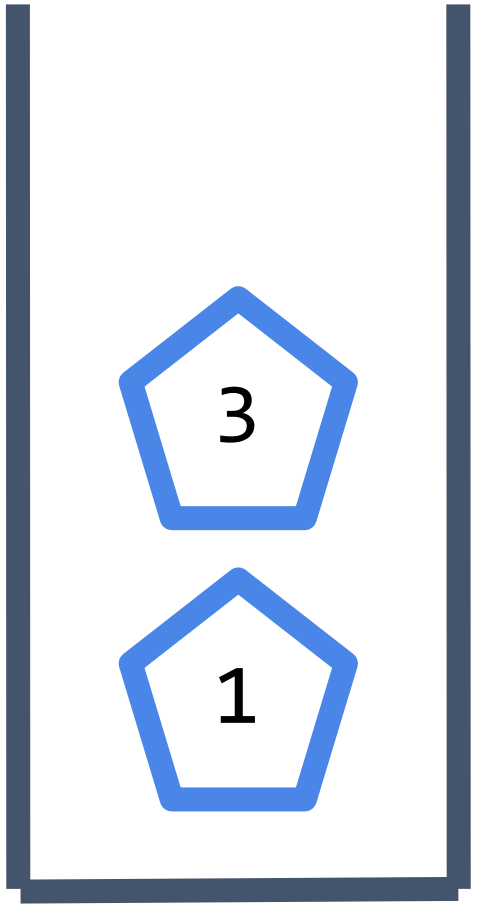
加入新資料 3

Stack（堆疊）



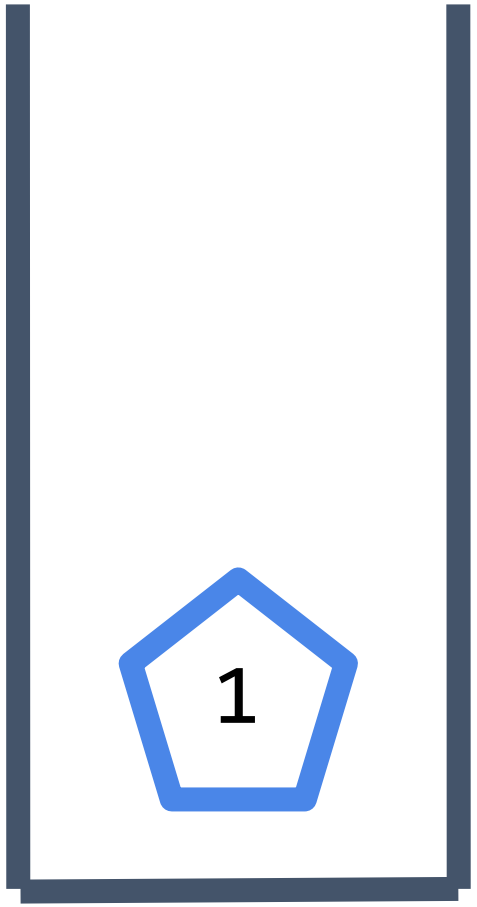
加入新資料 4

Stack（堆疊）



刪除最頂端資料

Stack（堆疊）



刪除最頂端資料

Stack 的功能

- 存取排在 stack 最頂端的資料
- 刪除排在 stack 最頂端的資料
- 新增資料到 stack 的最頂端

Stack 的特性

- 只能從最頂端存取、刪除、新增資料
- 後進先出(Last In First Out, LIFO)

用陣列實作 Stack

- top() 回傳 stack 最頂端的值
- pop() 刪除 stack 最頂端的資料
- push(x) 將一個新的值 x 加入 stack
- size() 回傳 stack 的大小

用陣列代表 Stack

- 假設任意時刻 stack 裡的資料筆數不會超過陣列大小

用變數 now 記錄頂端位置

- 如果 now = 0，代表 stack 是空的
- push 時 now++
- pop 時 now--

```cpp
template<typename T>
struct Stack{
    int now;
    T arr[MAXN];
    Stack(): now(0) {}
    // 回傳 stack 最頂端的值
    T& top() { return arr[now-1]; }
    // 刪除 stack 最頂端的資料
    void pop() { now--; }
    // 將一個新的值加入 stack 的最頂端
    void push(T val) { arr [now++] = val; }
    // 回傳 stack 的大小
    int size() { return now; }
};
```

Stack有什麼用？

遞迴是用 Stack 實作的

- 範例：F(n) = F(n-1) + F(n-2)

```cpp
int F(int x) {
    if (x < 2) return x;
    return F(x-1) + F(x-2);
}
cout << F(10) << endl;
```

也許有天可以在 stack 上二分搜。

```cpp
struct Node {
  int v, i = 0, s = 0;
};
Stack<Node> st{};
st.push(Node{ .v = 10 });
while (true) {
  while (st.top().i == 2)
    if (st.size() == 1) break;
    else {
        int s = st.top().s;
        st.pop();
        st.top().s += s;
    }
  if (st.top().i == 2) break;
  int v = st.top().v;
  if (v < 2) {
    st.pop();
    st.top().s += v;
  } else {
    int x = v - (++st.top().i);
    st.push(Node{ .v = x });
  }

cout << st.top().s << endl;
```

練習時間

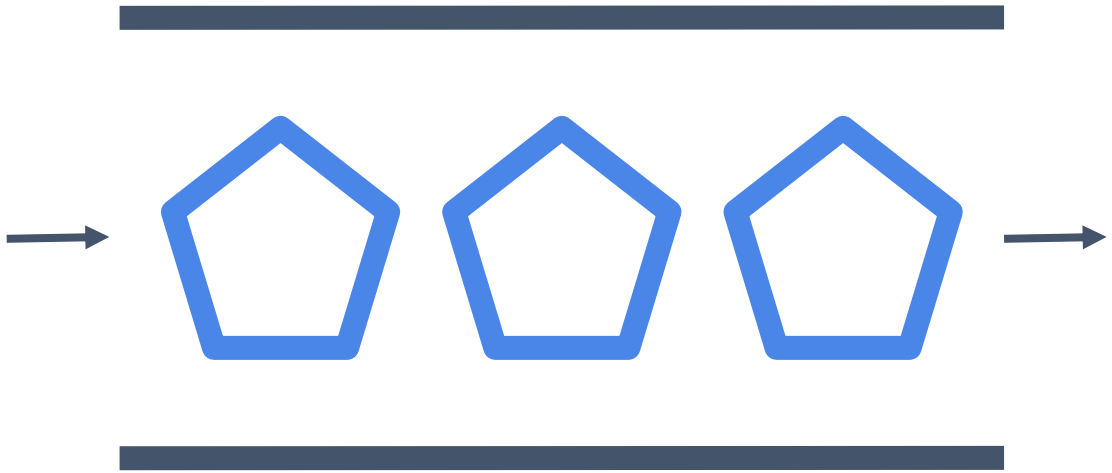
- NEOJ #36 https://neoj.sprout.tw/problem/36/
- 如果輕鬆水掉的話，可以挑戰看看 NEOJ #19, #22, #23, #512, #513

Queue

Queue

- 先到的先拿！

Queue（佇列）



Queue 的功能

- 存取排在 queue 最前端的資料
- 刪除排在 queue 最前端的資料
- 新增資料到 queue 的最後端

Queue 的特性

- 只能從最前端存取、刪除資料
- 只能從最後端新增資料
- 先進先出(First In First Out, FIFO)

用陣列實作 Queue

- front() 回傳 queue 最前端的值
- pop() 刪除 queue 最前端的資料
- push(x) 將一個新的值 x 加入 queue 的最後端
- size() 回傳 queue 的大小

和 stack 類似的方法

- 用變數 head, tail 記錄目前 queue 的開頭、結尾
- push 時 tail++
- pop 時 head++

但...可能會碰到問題

- 如果一直 push 東西進去後立刻 pop 出來？

但...可能會碰到問題

- 如果一直 push 東西進去後立刻 pop 出來？
- 雖然在過程中 queue 所存的東西數量不會超過上限，但？

Circular Queue

- 如果不斷進行 push 然後 pop 的操作會超過陣列限制

Circular Queue

- 如果不斷進行 push 然後 pop 的操作會超過陣列限制
- 碰到尾巴的話，就從頭再來一次！

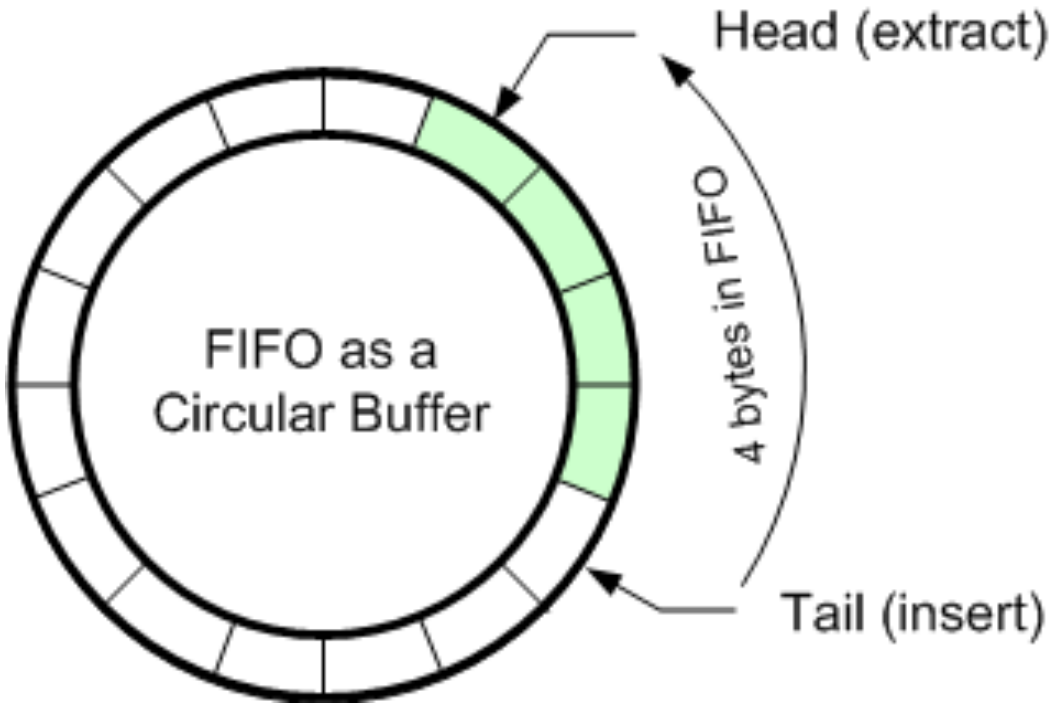


Image source: https://tex.stackexchange.com/questions/18363/diagrams-package-that-allows-to-draw-computer-buffers-memory-diagrams


```cpp
template<typename T>
struct Queue {
    int head, tail;
    T arr[MAXN];
    Queue(): head(0), tail (0) {}
    // 回傳 queue 最前端的值
    T& front() { return arr [head]; }
    // 刪除 queue 最前端的資料
    void pop() {
        head++;
        if (head == MAXN) head = 0;
    }
    // 將一個新的值加入 queue 的最後端
    void push (T val) {
        arr[tail++] = val;
        if (tail == MAXN) tail = 0;
    }
    // 回傳 queue 的大小
    int size() {
        return (tail + MAXN - head) % MAXN;
    }
};
```

練習時間

- NEOJ #37 https://neoj.sprout.tw/problem/37/
- 如果輕鬆水掉的話，可以挑戰看看 NEOJ #20
- 特別注意到宣告一個 queue 就會佔用一些記憶體了，再寫 NEOJ #20 的時候，注意不要開 10^5 量級個 queue (?)
  - source: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=77524

Deque

Deque（雙端佇列）

- 有些人喜歡念成「de-queue」

Deque（雙端佇列）

- ~~有些人喜歡念成「de-queue」~~
- usually pronounced like "deck" — by CPP reference

Deque 的功能

- 存取、刪除排在 deque 最前端的資料
- 存取、刪除排在 deque 最後端的資料
- 新增資料到 deque 的最前端、最後端

用陣列實作 Deque

- front(), back() 詢問
- pop_front(), pop_back() 刪除
- push_front(x), push_back(x) 加入
- size()

```cpp
template<typename T>
struct Deque {
    int head, tail;
    T arr [MAXN];
    Deque() : head (0), tail (0) {}
    // 回傳 deque 最前端的值
    T& front() {
        return arr [head];
    }
    // 回傳 deque 最後端的值
    T& back() {
        if (tail == 0) tail = MAXN;
        return arr[tail-1];
    }
    // 刪除 deque 最前端的資料
    void pop_front() {
        head++;
        if (head == MAXN) head = 0;
    }
    // 刪除 deque 最後端的資料
    void pop_back() {
        if (tail == 0) tail = MAXN;
        tail--;
    }
    // 將一個新的值加入 deque 的最前端
    void push_front(T val) {
        if (head == 0) head = MAXN-1;
        arr [head--] = val;
    }
    // 將一個新的值加入 deque 的最後端
    void push_back(T val) {
        arr[tail++] = val;
        if (tail == MAXN) tail = 0;
    }
    // 回傳 deque 的大小
    int size() {
        return (tail + MAXN - head) % MAXN;
    }
};
```

Linked-list

Linked-list 的概念

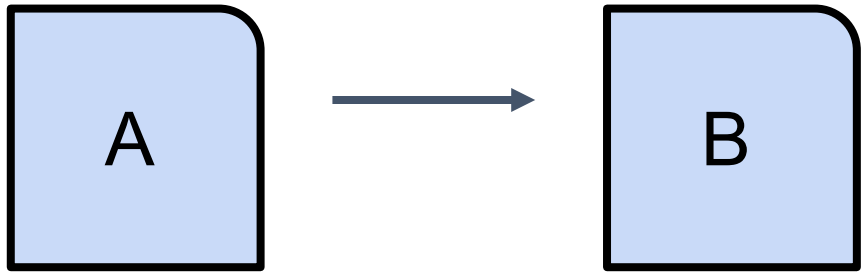
- 對於每個資料紀錄前後資料的位置
- 可以 O(1) 加入、刪除特定資料
- 不支援 random-access
  - 不能 O(1) 存取指定 index 的資料

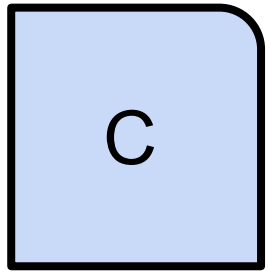
Linked-list 的概念

- 對於每個資料紀錄前後資料的位置
- 可以 O(1) 加入、刪除特定資料
- 不支援 random-access
  - 不能 O(1) 存取指定 index 的資料
- 這跟陣列不一樣的地方在哪？

加入資料

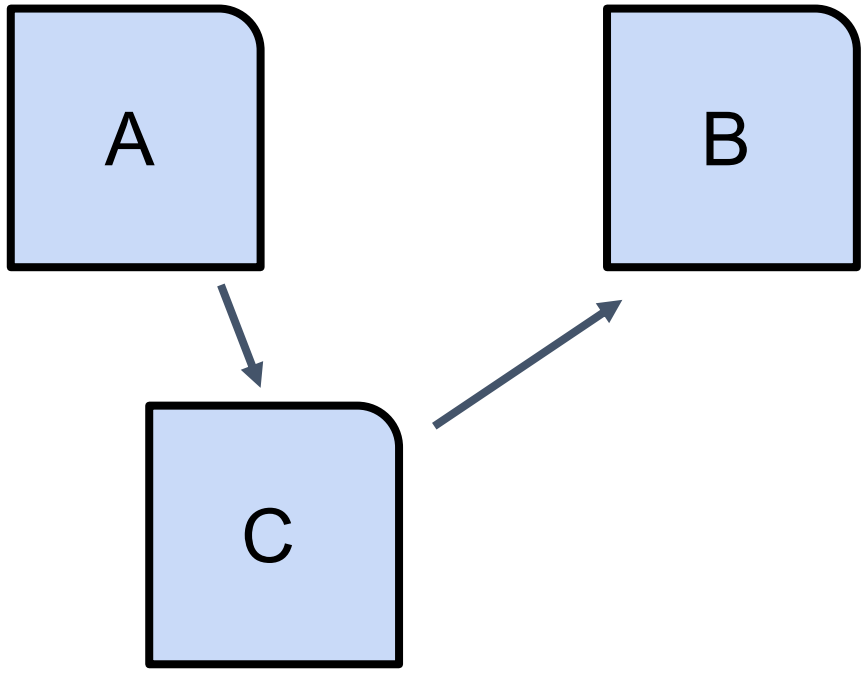
- 假設我們想將資料 C 插入在資料 A、B 之間





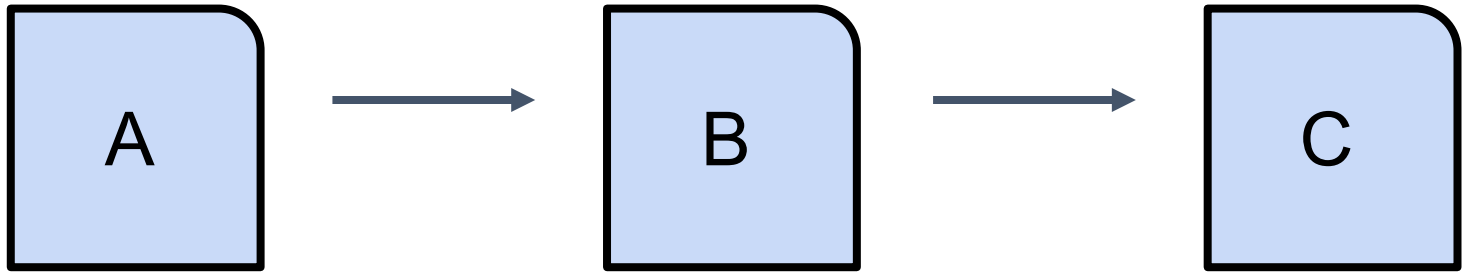
加入資料

- 改變他們指向前後的那些箭頭！



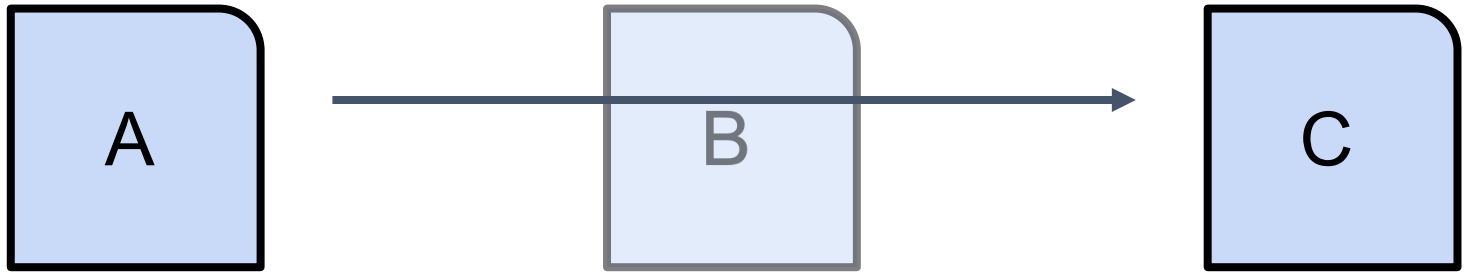
刪除資料

- 假設我們想將資料 B 從資料 A、C 之間刪除



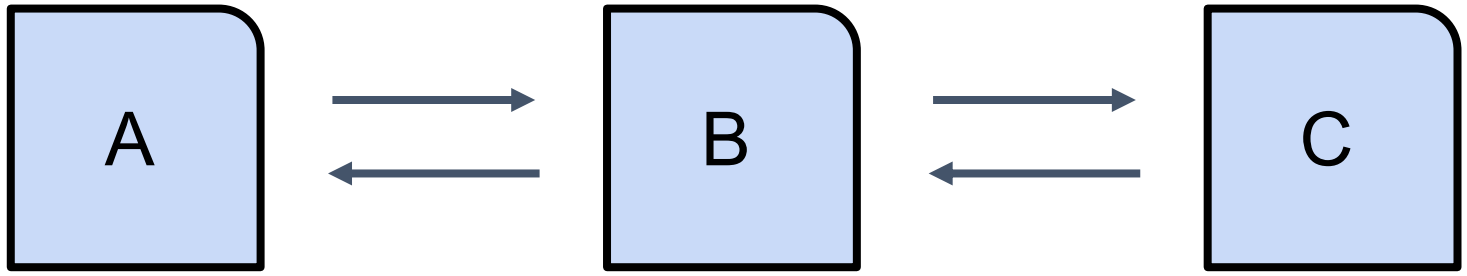
刪除資料

- 假設我們想將資料 B 從資料 A、C 之間刪除



雙向 Linked List

- 單向不夠好用



練習時間

- Try NEOJ #21, #25
- #21 Hint: https://pastebin.com/sWRRxrpz

STL

Sprout

stack

- C++ 的 STL 中，也有提供 stack 用

- #include <stack>

- std::stack<int> sta;
- sta.push(1);
- std::cout << sta.top() << '\n';
  // 只會回傳最頂端的元素，不會 pop
- sta.pop();
  // 只會 pop ，不會回傳最頂端的元素

queue

- C++ 的 STL 中，也有提供 queue 用

- #include <queue>

- std::queue<int> que;
- que.push(1);
- std::cout << que.front() << '\n';
  // 只會回傳最前端的元素，不會 pop
- que.pop();
  // 只會 pop ，不會回傳最前端的元素

deque

- C++ 的 STL 中，也有提供 deque 用

- #include <deque>

- std::deque<int> deq;
- deq.push_front(1);
- std::cout << deq.front() << ‘\n’;
- que.pop_front();

- std::cout << deq[k]<< ‘\n’;

Linked list

- C++ 的 STL 中，也有提供 linked list 用

- #include <list>

- std::list<int> lis;
- lis.push_front(1);
- std::cout << lis.front() << ‘\n’;
- lis.pop_front();

例題討論

例題一、括弧匹配

括弧匹配

給定一個僅包含 '('、')' 的字串，問其是否為合法括弧字串。

範例：

"()(()())" 是一個合法括弧字串

"()((()()" 不是一個合法括弧字串

括弧匹配

- 什麼樣的字串是合法括弧字串？

括弧匹配

- 什麼樣的字串是合法括弧字串？
  - 「每個左括弧都能夠找到右括弧與其互相配對，且不會有多餘的右括弧沒有配對到」

括弧匹配

- 由左到右把字元加到 stack 看看
  - 遇到 '(' 就 push
  - 遇到 ')' 就 pop
- 什麼樣的情況是非法字串？

括弧匹配

- 由左到右把字元加到 stack 看看
  - 遇到 '(' 就 push
  - 遇到 ')' 就 pop
- 什麼樣的情況是非法字串？
  - 如果 pop 的時候發現 stack 空了
    - => 非法字串
  - 如果 stack 最後不是空的
    - => 非法字串

例題二、加減運算

題目敘述

給定一個包含 '('、')'、'+'、'-' 的運算式，計算該運算式的答案。（保證該運算式合法）

範例：

"(5+4)-3" 的答案是 6

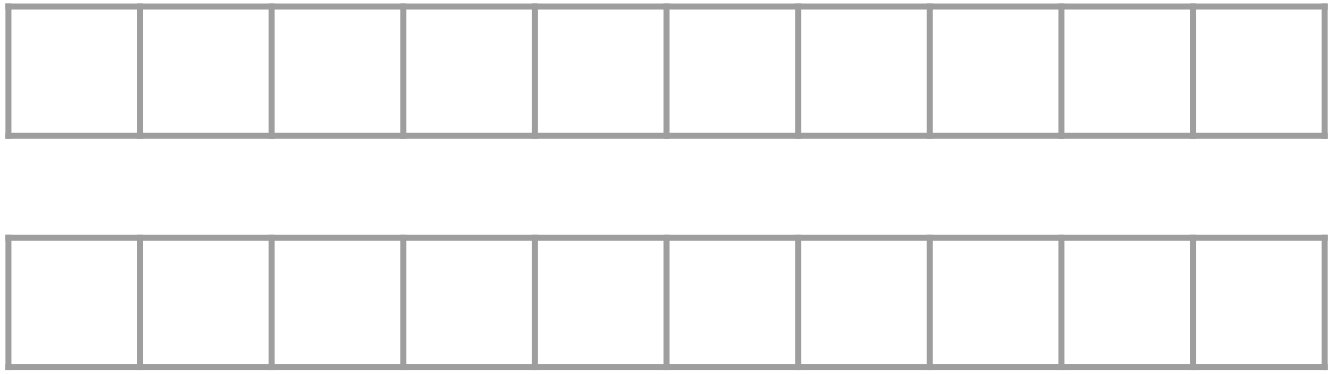
"(1+2)-(7+3)" 的答案是 -7

Hint

- 可以從後面做回來
- 兩個 Stack 比較好實作

作法

- 兩個 stack，一個紀錄符號、一個紀錄數值
- 碰到 '(' 或結尾再做事！



11 - (2 - (3 + 2)) + 5

還有一個小問題

- 數字不是個位數怎麼辦？

實作細節

- 開一個變數紀錄目前的 digit 是 10 的幾次方，先處理好數字再丟進 stack 中
- 遇到 '(' 往前計算的時候不是只算一次，是一路到 ')' 並將其 pop 出來為止！

延伸問題

給定一個包含加減乘除和括弧的四則運算式，計算其答案。

例題三、長條圖最大矩形

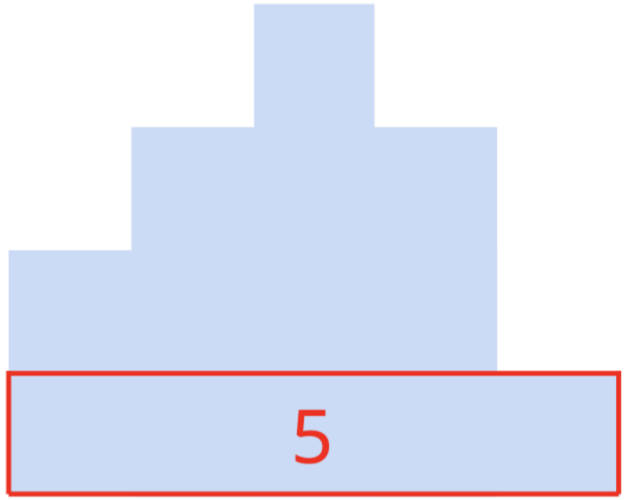
長條圖最大矩形

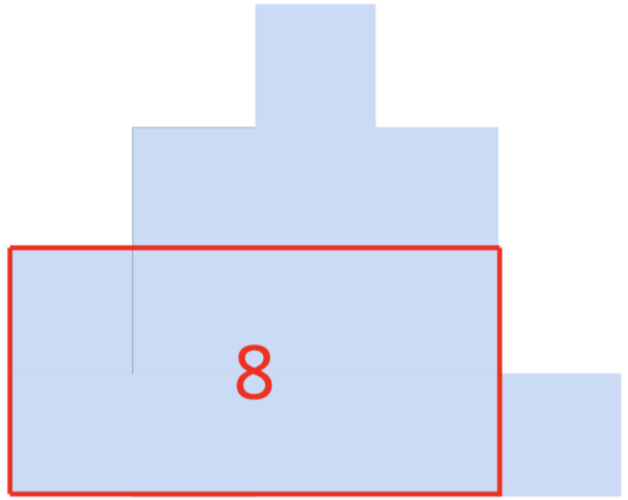
給你一張長條圖每個位置的高度，問你能畫出的最大矩形面積。

（N <= 10^5、高度 <= 10^9）

範例：

2 3 4 3 1





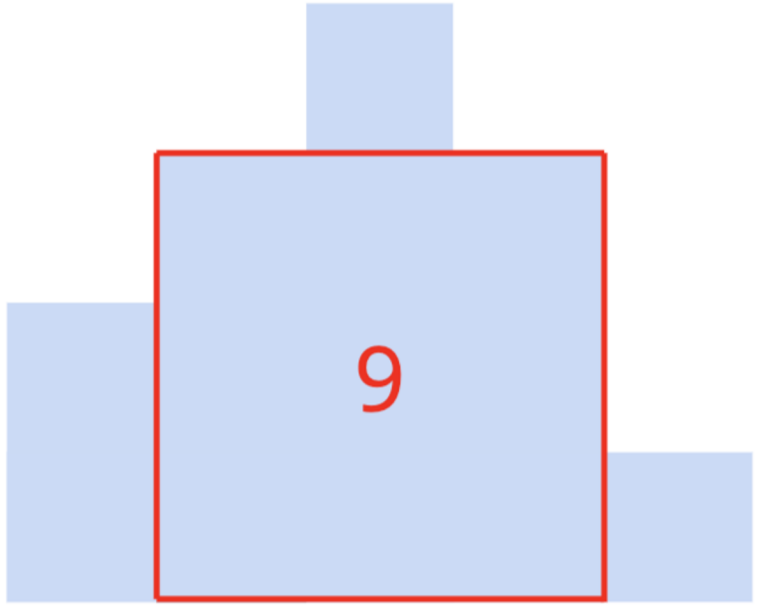
長條圖最大矩形

給你一張長條圖每個位置的高度，問你能畫出的最大矩形面積。

（N <= 10^5、高度 <= 10^9 ）

範例：

2 3 4 3 1



長條圖最大矩形

- 不知道從何下手的時候，可以先從複雜度較差的解開始想！

直覺的做法

- 枚舉每段區間，然後看高度最高可以是多少
- 正確性？
- 複雜度？$O(N^3)$
  - 區間總共有 $N(N+1)/2$ 個
  - 高度至多只能到最矮的那個
    - => 掃一遍區間找最小值

再想多一點...

- 「一塊區間的高度至多只能到最矮的那個」

再想多一點...

- 「一塊區間的高度至多只能到最矮的那個」
- 重點不是區間，是「最矮的那個」
  - 從枚舉區間，變成枚舉每個 bar 的高度

再想多一點...

- 「一塊區間的高度至多只能到最矮的那個」
- 重點不是區間，是「最矮的那個」
  - 從枚舉區間，變成枚舉每個 bar 的高度
- 如果我是最低的，那往左往右至多可以延伸多少？
  - 只要分別找到左右兩邊第一個比我小的！

問題轉換

- 給定序列，對每一項分別找到左右離他最近且比他小的值。
  （N <= 10^5、高度 <= 10^9 ）
  - 其實等價於對每一項找到左邊離他最近且比他小的值，
    然後再把序列反轉過來做一次
  - 複雜度？
    - 但有沒有可能做得更好呢？

作法

- 考慮每一項在什麼時間點以後注定不可能成為答案

作法

- 考慮每一項在什麼時間點以後注定不可能成為答案
- 「如果右邊有東西不比我大，那我就不可能是答案」
  - 我們要用 stack 維護這樣的「單調性」
  - stack 裡頭的每一項一定比前一項大

作法

- 考慮每一項在什麼時間點以後注定不可能成為答案
- 「如果右邊有東西不比我大，那我就不可能是答案」
  - 我們要用 stack 維護這樣的「單調性」
  - stack 裡頭的每一項一定比前一項大

```
1 // top 比我還大
2 while (size() > 0 && top() >= value[idx])
3     pop(); // 把 top 丟掉
4 if (size() > 0) ans[idx] = top();
5 // 目前的 top 會是比我小且離我最近的那個
6 push(value[idx]);
7 // 記得把值丟進 stack
```

思路、步驟整理

1. 要找最大矩形，可以「枚舉每個值作為最小值」向外延伸
2. 將向左、向右拆開成兩個問題
3. 題目轉化為「找到左邊第一個比我小的值」
4. 一個值不可能成為最小值的條件（右邊出現比它小的值）
5. 利用 stack 維護這樣的「單調遞增」

延伸問題

給定長度為 N 的序列，問每個長度 K 連續區間的區間最大值。（N, K <= 10^6）

謝謝大家！

Sprout