鍊結串列 Linked List

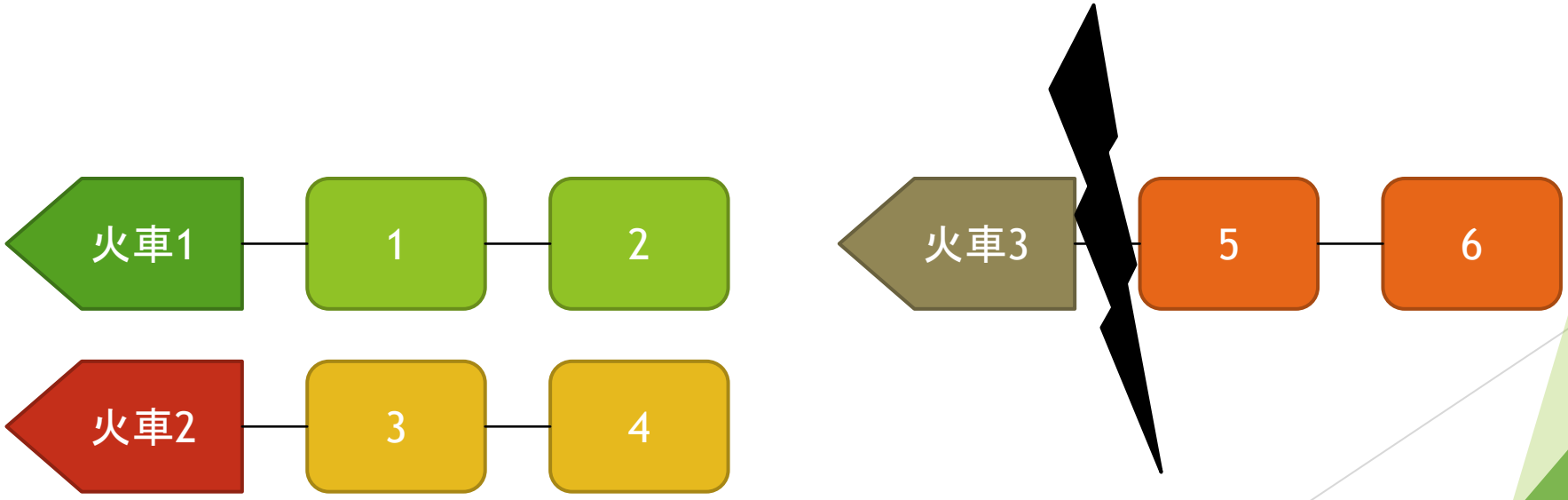
redleaf23477
(2022 Summer TA)
chchiang
(2023 Spring TA)
葉宥辰
(2023 Summer TA)
黃頂軒
(2024 Spring TA)

生活情境題...

- 有N台火車，每節車廂有個字的編號，總共有M節車廂
- 每次把第a台火車條插到第b台火車的後面
  - 火車頭不可能接到b的後面，所以就留在原地吧
- 最後詢問每台火車長怎樣

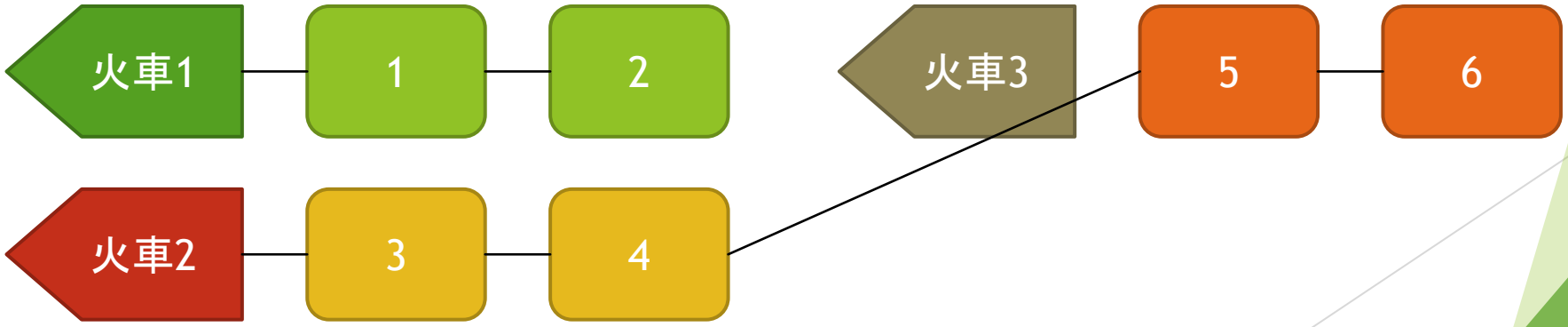
把第3台火車條插到第2台火車的後面

- 正常做法:
- 1. 先把火車3的火車頭和車廂斷開



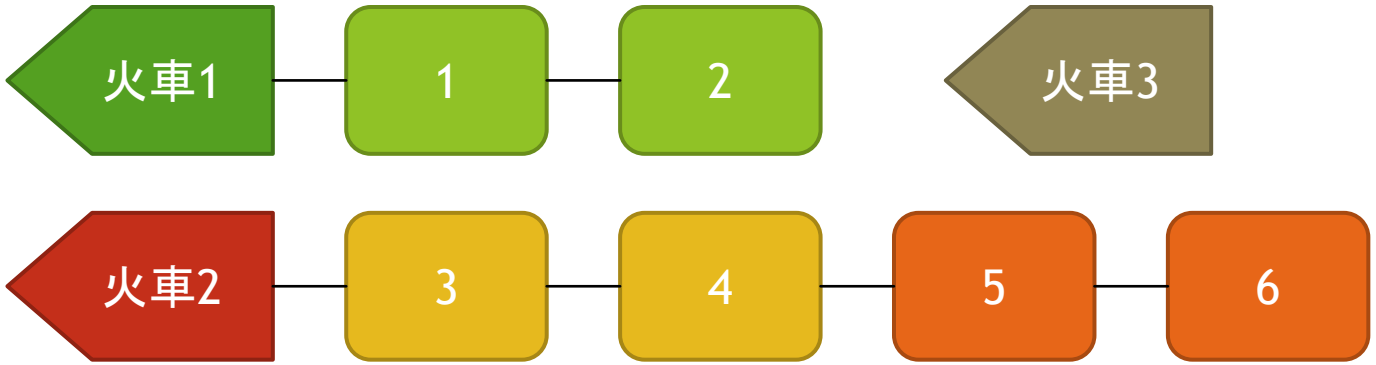
把第3台火車條插到第2台火車的後面

- 2. 把車廂接到火車2的後方



把第3台火車條插到第2台火車的後面

- 然後就完成了~~



用程式寫寫看？

- 怎麼表示一台火車？用我們很熟悉的陣列試試看？
- 一台火車建一個1D陣列，很多台火車建2D陣列？
  - 尷尬了，陣列建多大？
  - 一台火車最長M節，總共N台，建N*M好了…
  - 對於每台火車，紀錄他的長度

	0	1	2	3	4	5	長度
火車1	1	2					2
火車2	3	4					2
火車3	5	6					2

把第3台火車條插到第2台火車的後面

```
for( int i = 0; i < 火車3長度; i++ )
    火車2[火車2長度++] = 火車3[i];
火車3長度 = 0;
```

	0	1	2	3	4	5
火車1	1	2				
火車2	3	4	5	6		
火車3						

長度
2
4
0

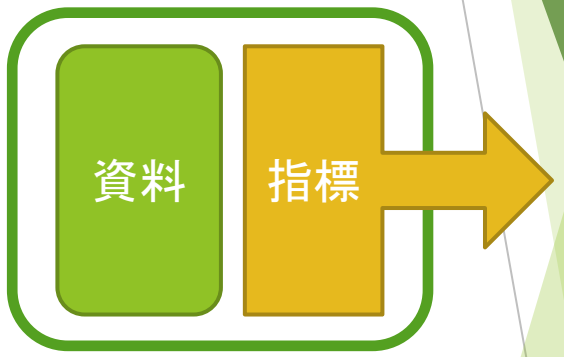
感覺這個寫法很虧

- 浪費很多空間
  - 開了N*M, 實際只用M
- 好浪費時間
  - 每次操作O(M)
- 陣列看起來不是個模擬火車的好東西RRRRR

- 能不能像正常作法一樣？

鍊結串列 Linked List

- Linked List 是個像火車一樣的資料結構
- 由好多節點(Node)組成，每個Node包含
  - 資料
  - 指標，指向下一個節點，或是NULL



鍊結串列 Linked List

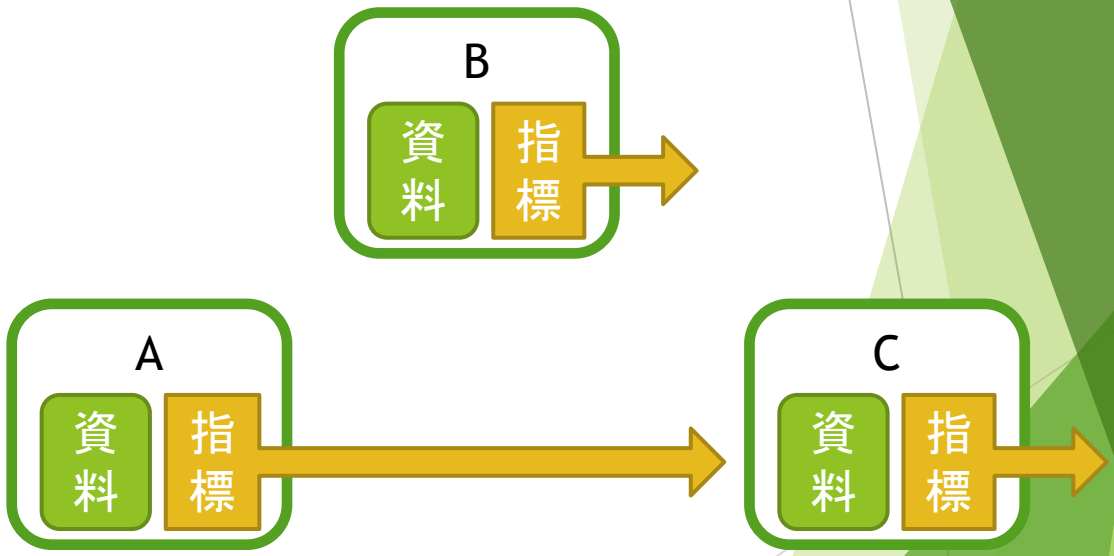
- 一條Linked List長這樣
  - head指標: 指向這條list的起點Node
  - 最後一個Node指向NULL

Linked List 的基本操作

- 插入節點：在某個Node後面插入新的Node
- 刪除節點：移除某個Node

插入節點

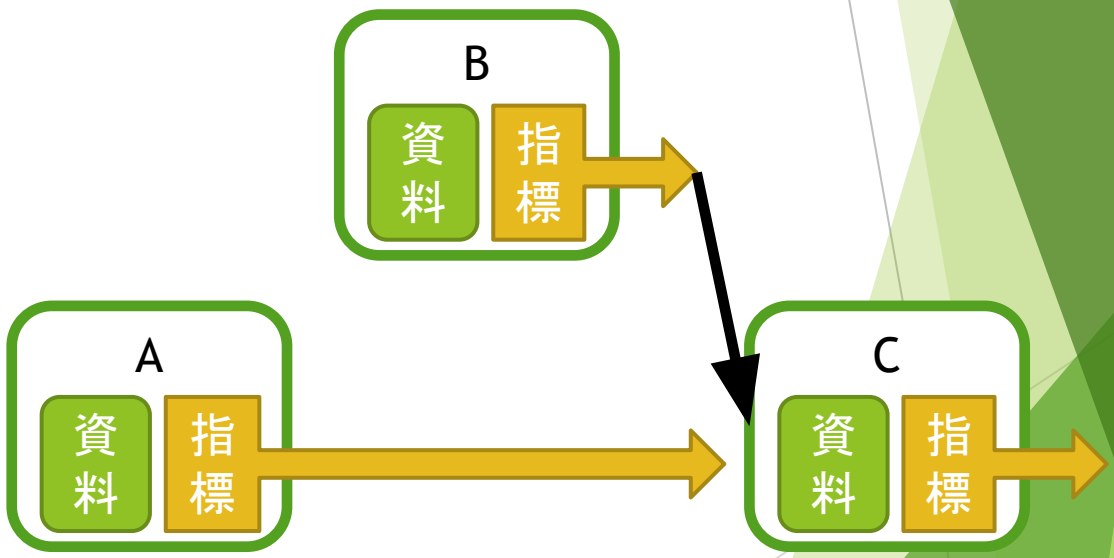
- 在A節點後面插入B節點



插入節點

- 在A節點後面插入B節點

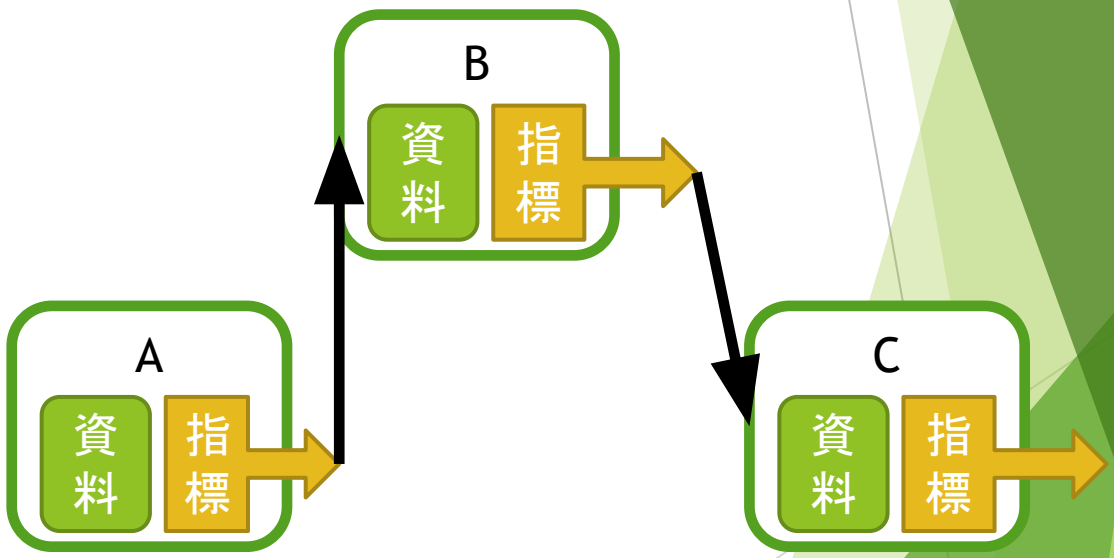
1. B指向A指到的節點



插入節點

- 在A節點後面插入B節點

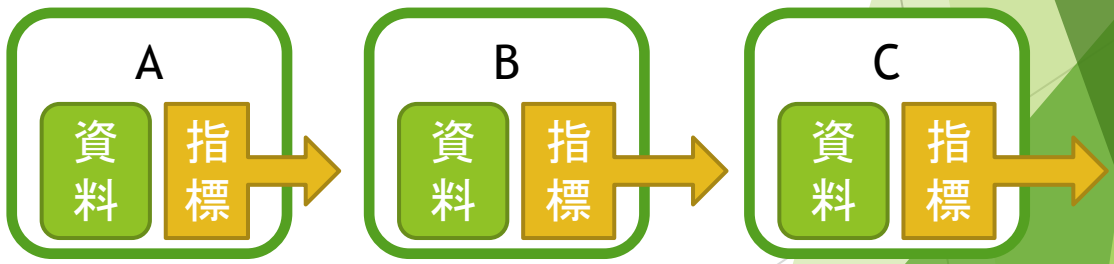
1. B指向A指到的節點
2. A指向B



插入節點

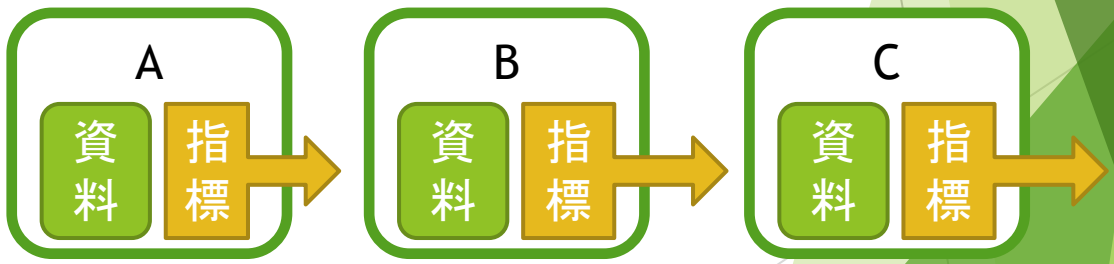
- 在A節點後面插入B節點

1. B指向A指到的節點
2. A指向B
3. 完成~~



刪除節點

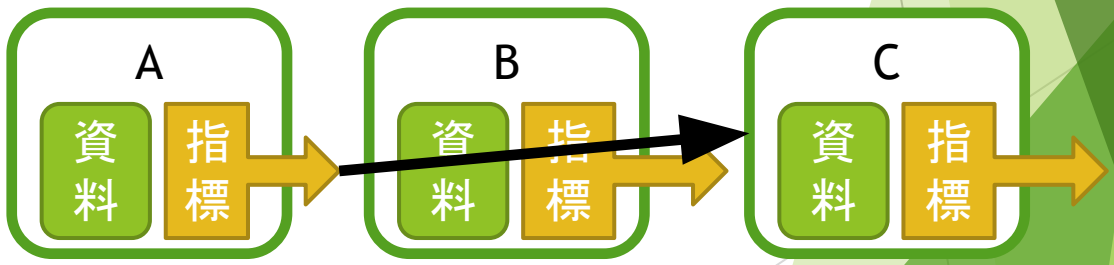
- 刪除B節點



刪除節點

- 刪除B節點

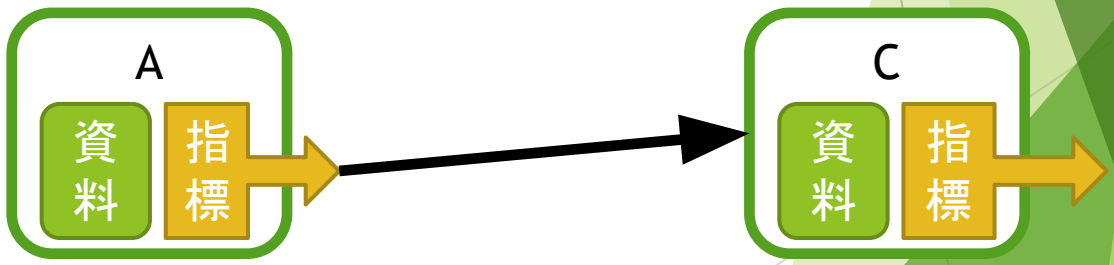
1. A指到B指到的節點



刪除節點

- 刪除B節點

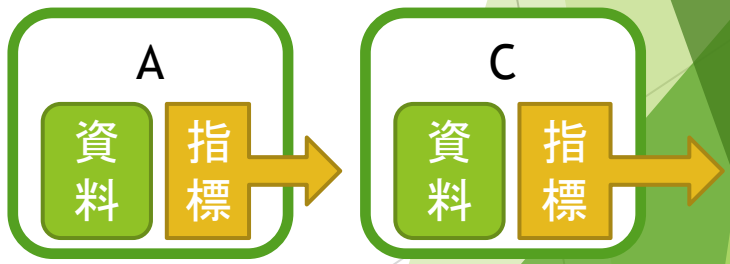
1. A指到B指到的節點
2. 消滅B



刪除節點

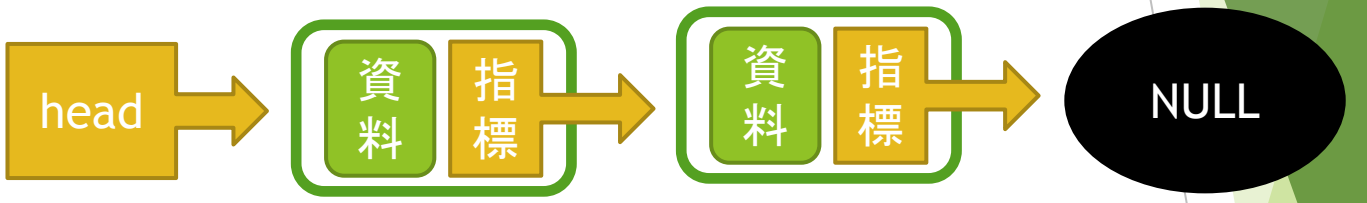
- 刪除B節點

1. A指到B指到的節點
2. 消滅B
3. 完成~~

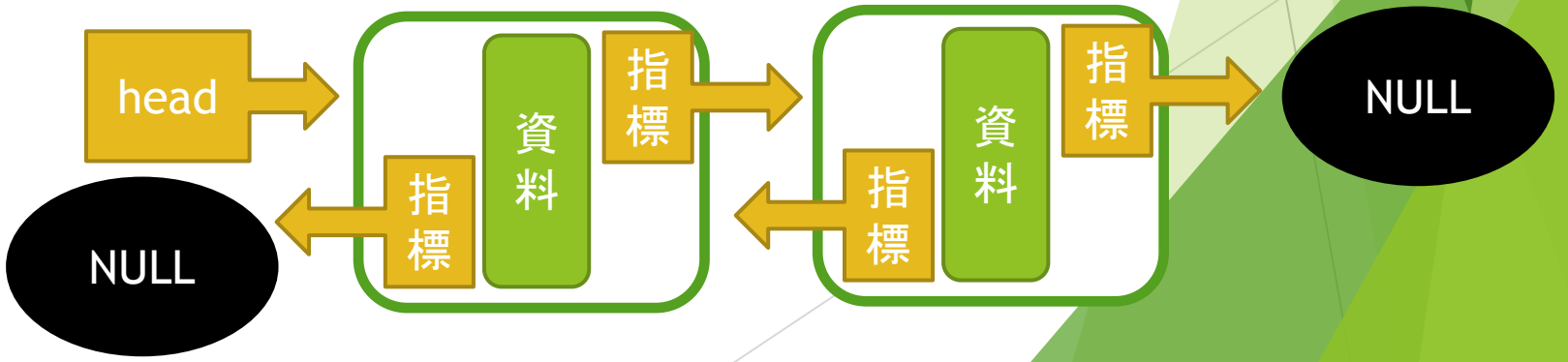


Singly Linked List / Double Linked List

- Singly Linked List
  - 指向下一個Node

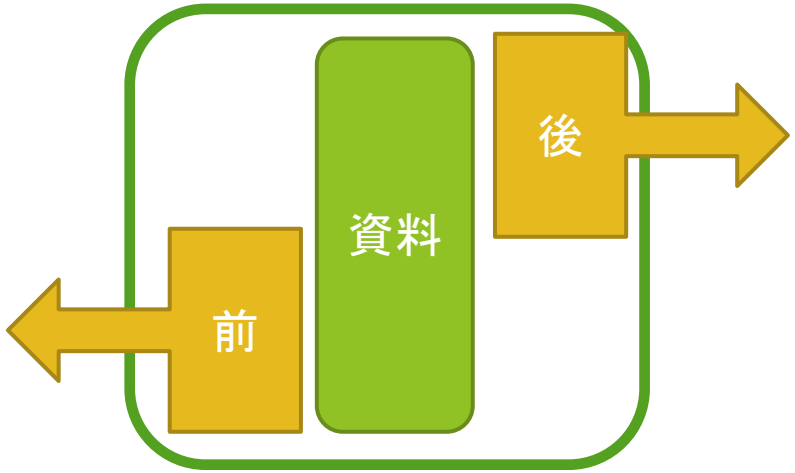


- Double Linked List
  - 指向下一個和前一個Node
  - 支援一樣的操作

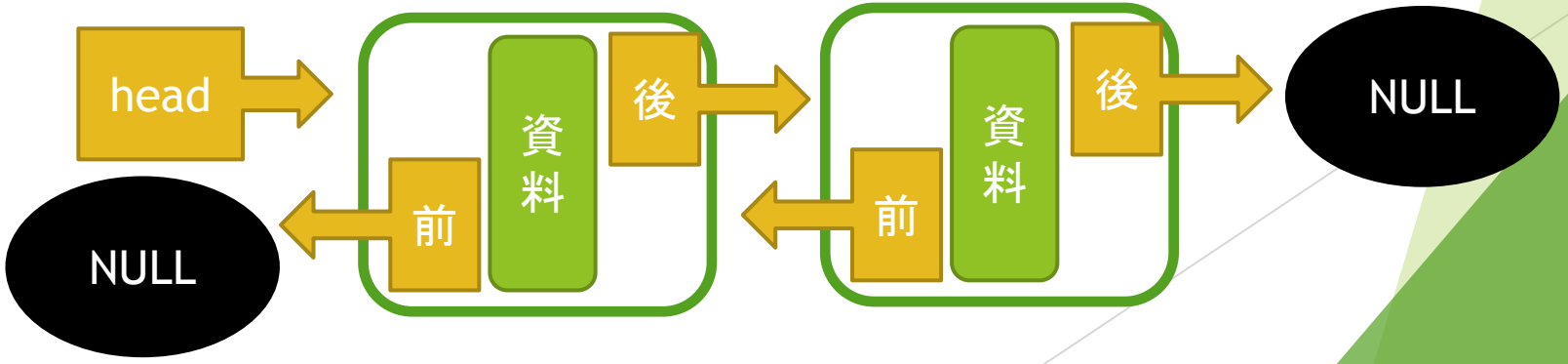


Double Linked List

- Double Linked List 的節點
  - 資料
  - next：指向後一個節點
  - prev：是向前一個節點

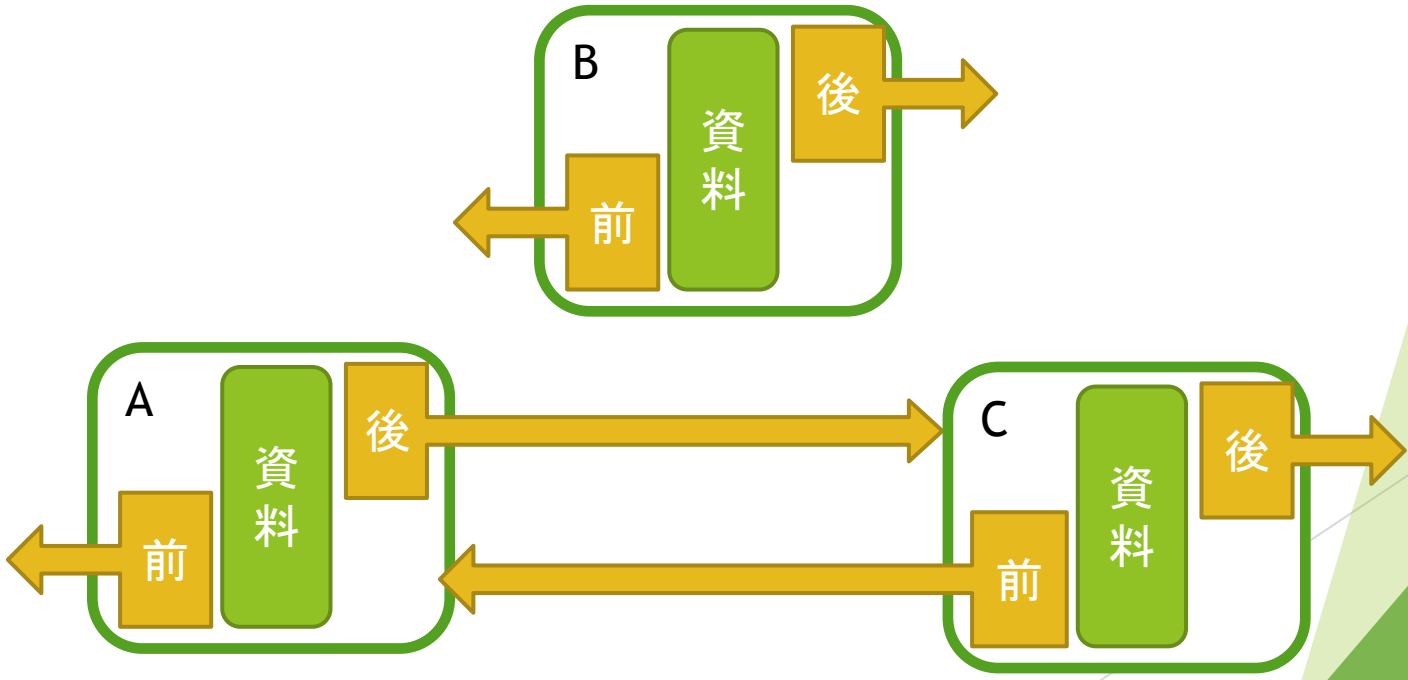


- 一個Double Linked List長這樣



插入節點

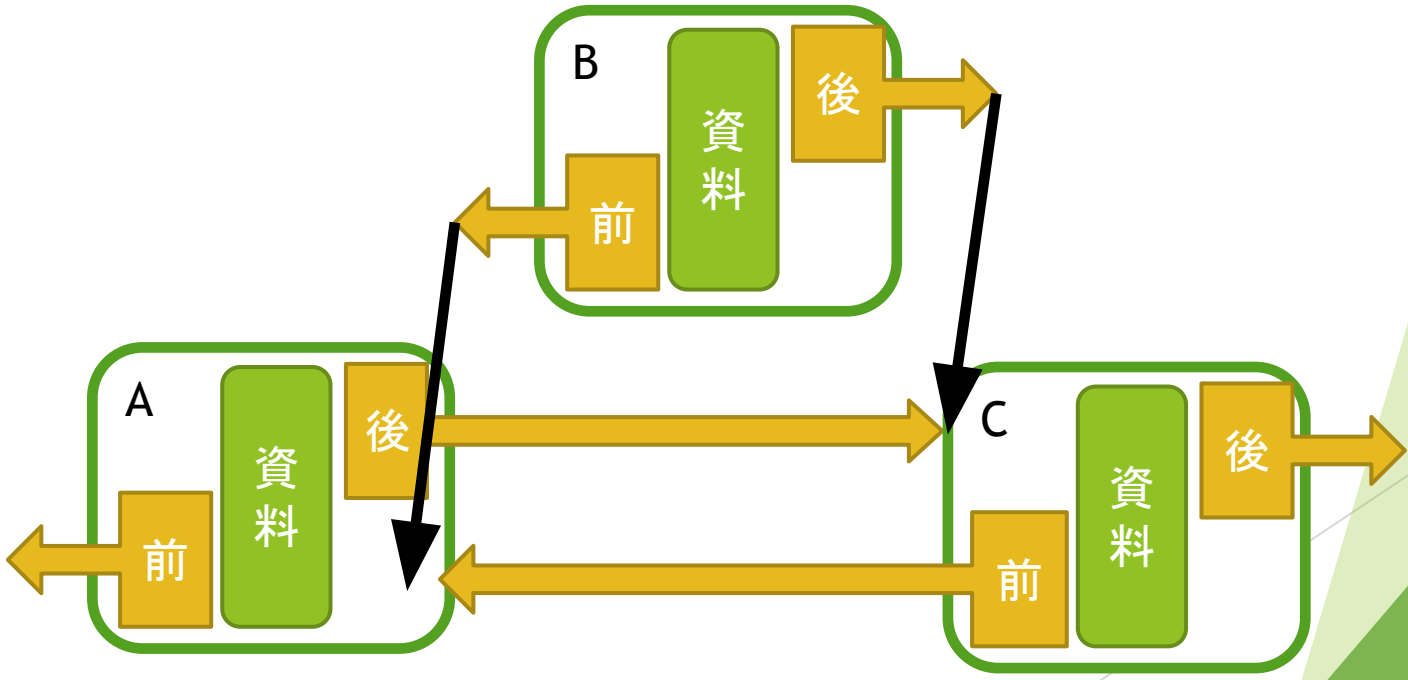
- 在A節點後面插入B節點



插入節點

- 在A節點後面插入B節點

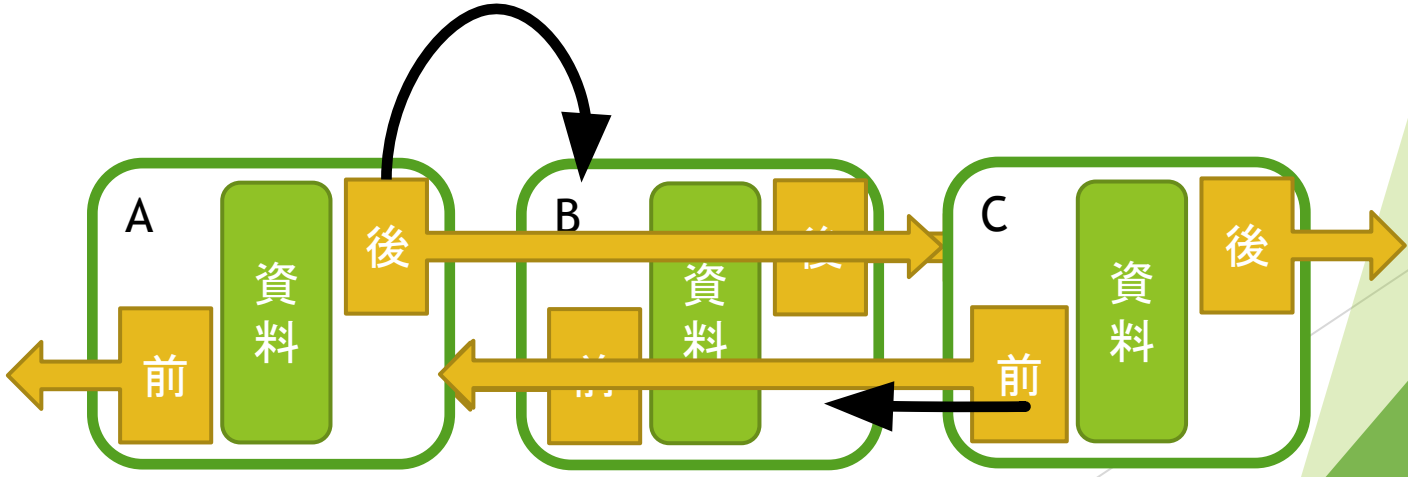
1. 把B接好



插入節點

- 在A節點後面插入B節點

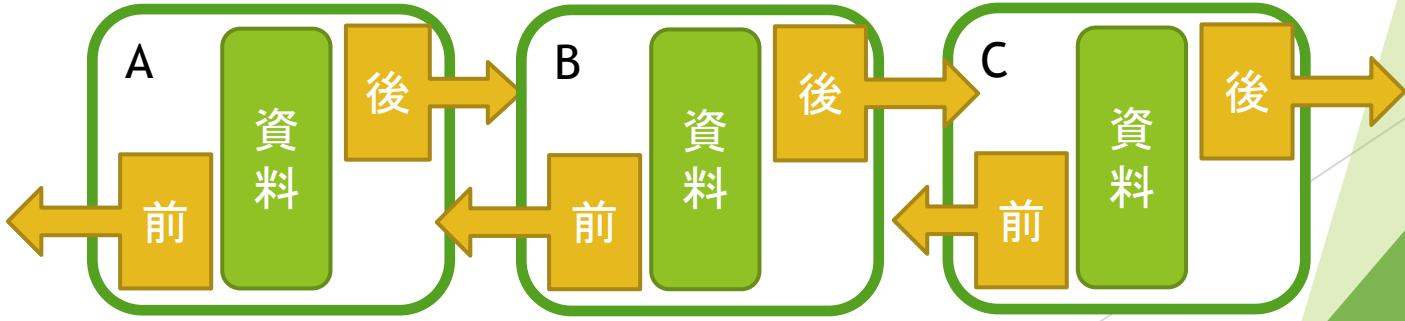
1. 把B接好
2. 把A、C接好



插入節點

- 在A節點後面插入B節點

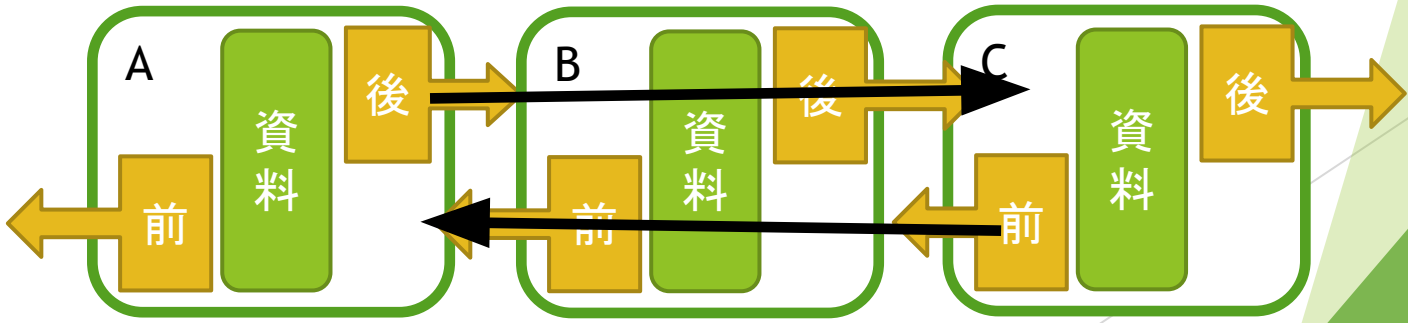
1. 把B接好
2. 把A、C接好
3. 完成~~



刪除節點

- 刪除B節點

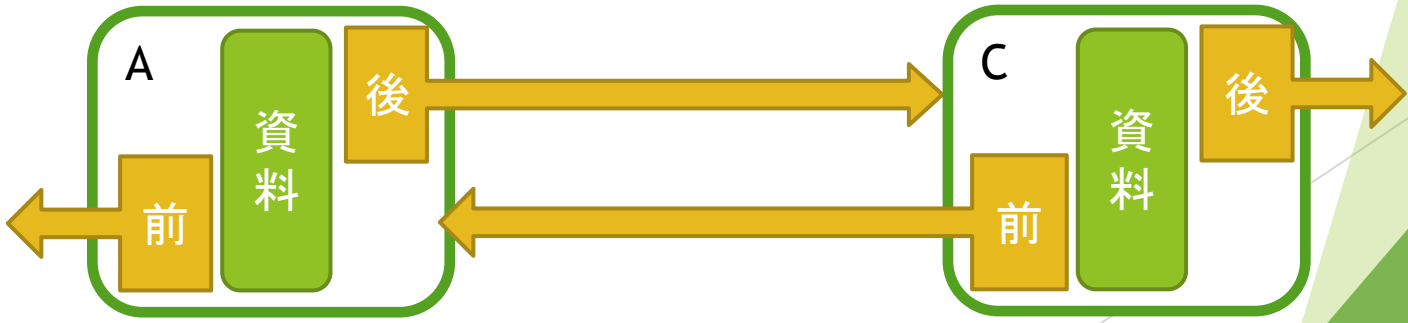
1. 把A、C接好



刪除節點

- 刪除B節點

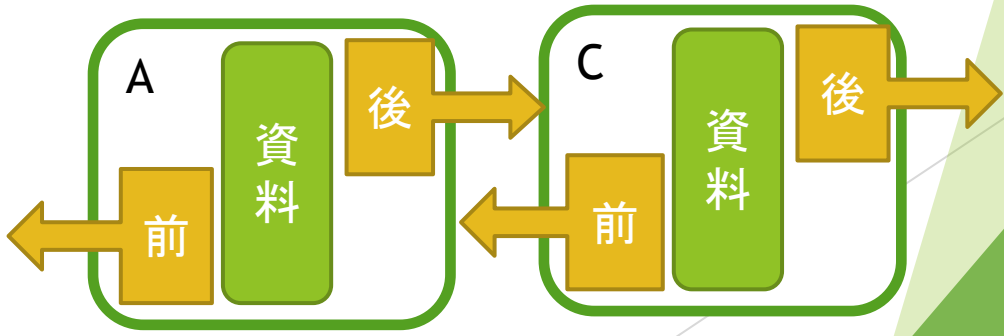
1. 把A、C接好
2. 幹掉B



刪除節點

- 刪除B節點

1. 把A、C接好
2. 幹掉B
3. 完成~~



來看看時間複雜度

- Single Linked List
- 插入一個Node
  1. B指向A指到的節點
  2. A指向B
- 刪除一個Node
  1. A指到B指到的節點
  2. 消滅B

- Double Linked List
- 插入一個Node
  1. 把B接好
  2. 把A、C接好
- 刪除一個Node
  1. 把A、C接好
  2. 幹掉B

來看看時間複雜度

- Single Linked List
- 插入一個Node
  1. B指向A指到的節點
  2. A指向B
- 刪除一個Node
  1. A指到B指到的節點
  2. 消滅B

- Double Linked List
- 插入一個Node
  1. 把B接好
  2. 把A、C接好
- 刪除一個Node
  1. 把A、C接好
  2. 幹掉B

都是O(1)阿

來看看時間複雜度

- Single Linked List
- 尋找從前面數來第 i 個元素
  1. 從第一個元素開始往後找

- Double Linked List
- 尋找從前面數來第 i 個元素
  1. 從第一個元素開始往後找

來看看時間複雜度

- Single Linked List
- 尋找從前面數來第 i 個元素
  1. 從第一個元素開始往後找

- Double Linked List
- 尋找從前面數來第 i 個元素
  1. 從第一個元素開始往後找

竟然要 O(N)

怎麼實做？

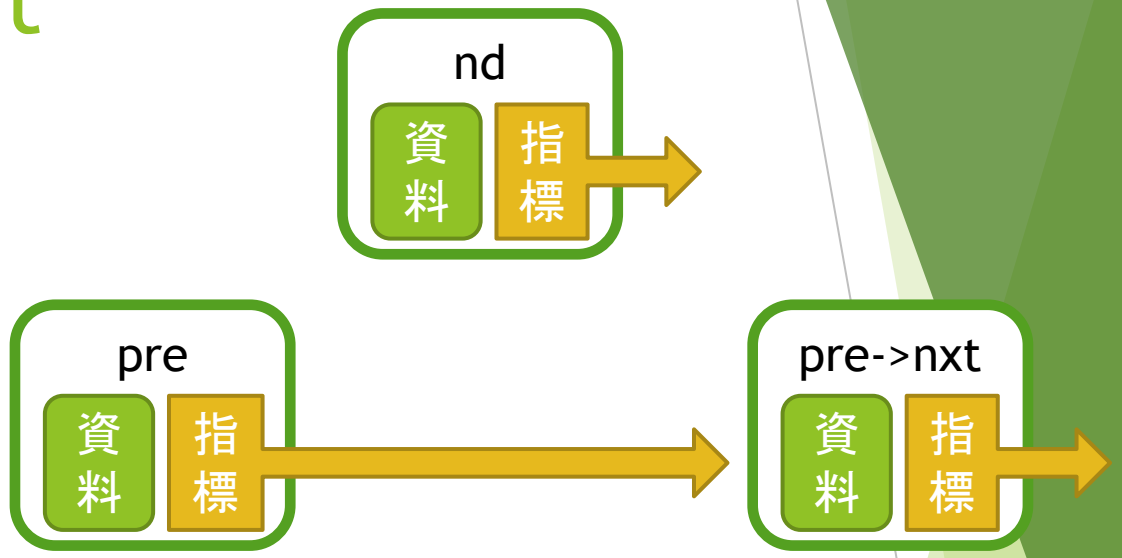
- 一個 Node 裡面不只存一個資料？
  - 建立一個struct
- Linked List 的長度不固定？
  - 動態記憶體配置：new / delete

```
// single linked-list
struct Node {
  int data;
  Node *nxt;
};

// double linked-list
struct Node {
  int data;
  Node *nxt, *prv;
};
```

Single Linked List - insert

- 建立一個值為data的節點
- 插到pre節點的後面
- 注意順序

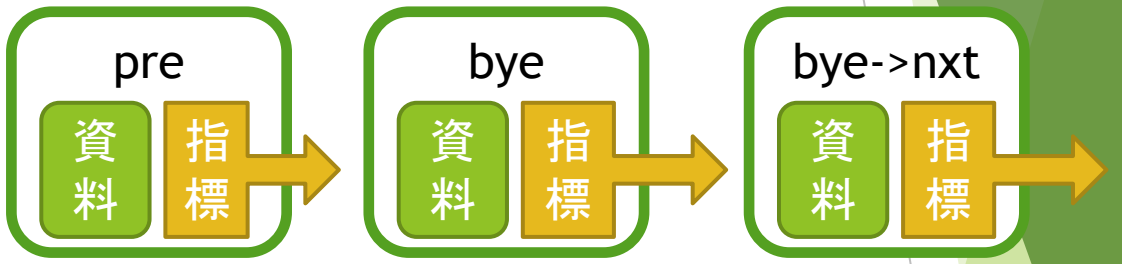


1. 建立新節點nd，賦值
2. 接好nd
3. 接好pre

```
void ins(Node *pre, int data)
{
    Node *nd = new Node;
    nd->data = data;
    nd->nxt = pre->nxt;
    pre->nxt = nd;
}
```

Single Linked List - delete

- 幹掉pre的下一個節點
- 注意順序



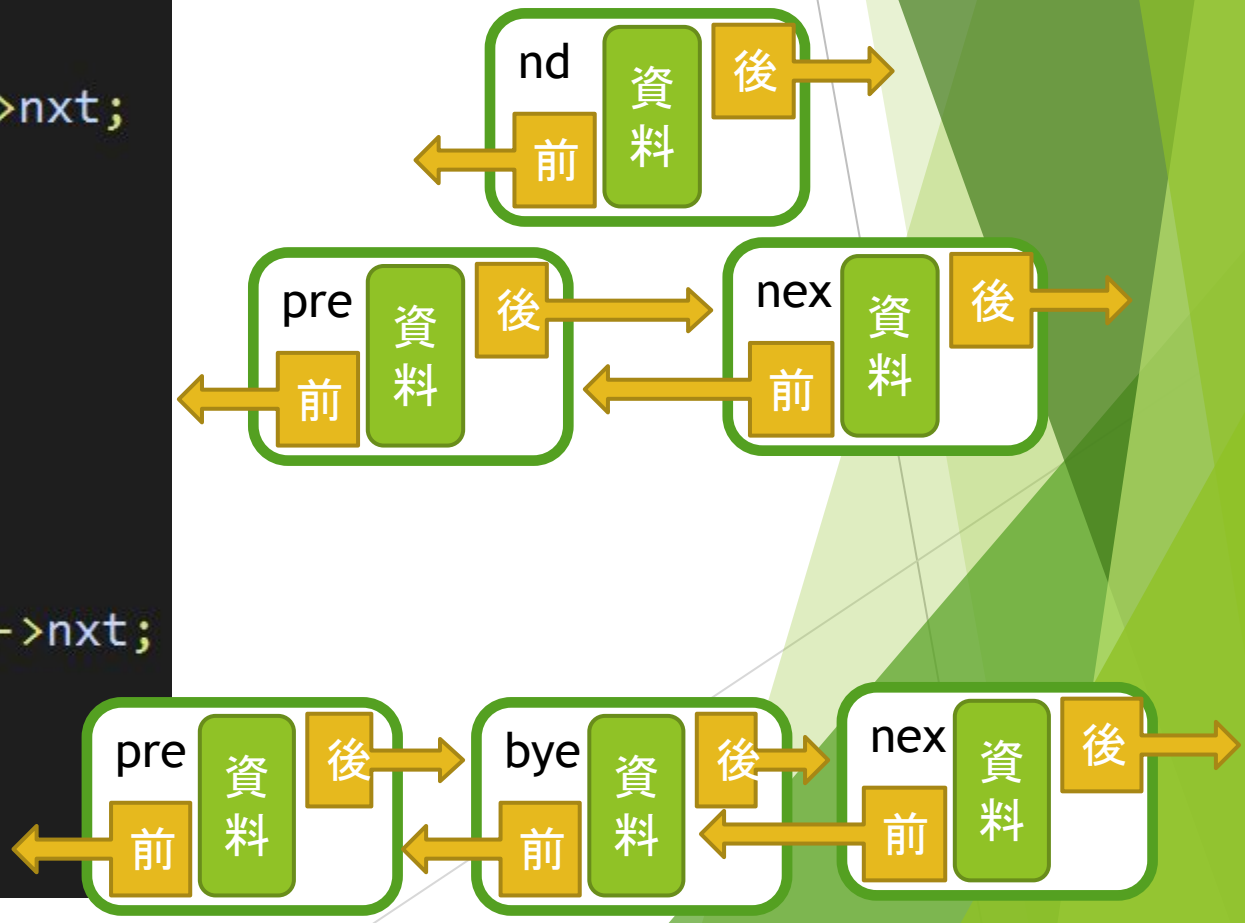
1. bye指向要刪的節點
2. 接好pre
3. 刪掉bye

```
void del(Node *pre)
{
    Node *bye = pre->nxt;
    pre->nxt = bye->nxt;
    delete bye;
}
```

Double Linked List

```
void ins(Node *pre, int x)
{
    Node *nd = new Node, *nex = pre->nxt;
    nd->data = x;
    nd->prv = pre, nd->nxt = nex;
    pre->nxt = nd;
    if(nex) nex->prv = nd;
}

void del(Node *bye)
{
    Node *pre = bye->prv, *nex = bye->nxt;
    if(pre) pre->nxt = nex;
    if(nex) nex->prv = pre;
    delete bye;
}
```

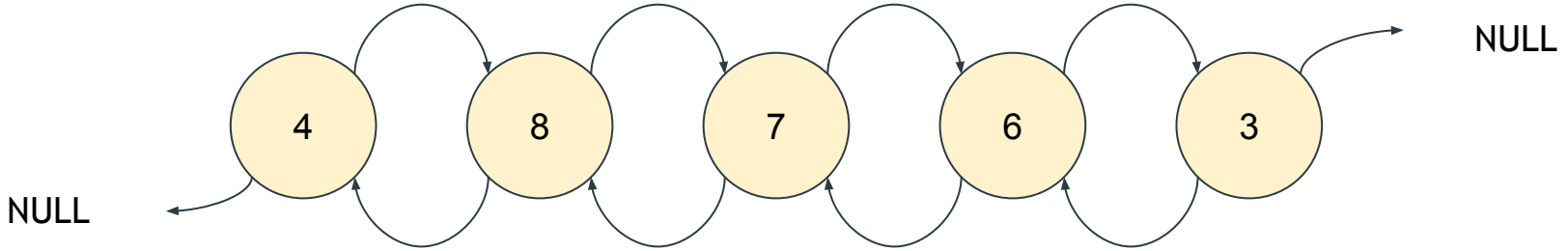


想一想

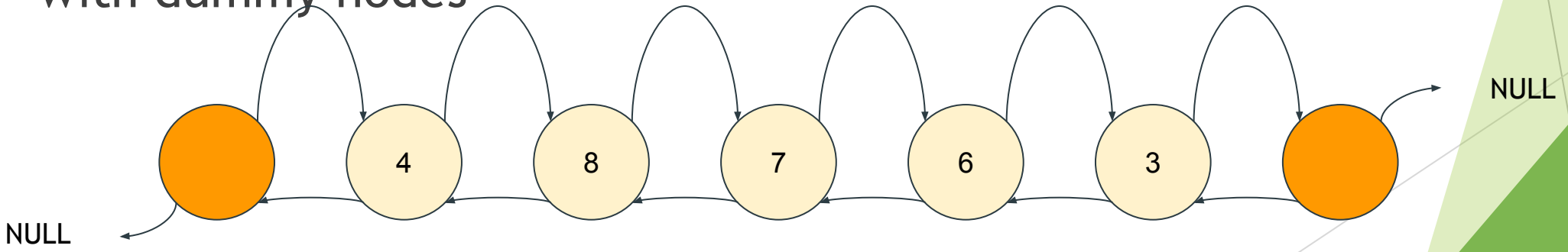
- 給你一個陣列 { 9, 4, 8, 7, 7, 1, 2, 2 }，把他建成Link List
- 在某節點pre後面插入一個link list？
- 給你一個List，問說裡面有沒有值為 x 的節點？
  - 如果他排序了，可以二分搜嗎

Dummy Nodes

without



with dummy nodes



Constructors and Destructors

```
struct Node {
  int data;
  Node *prv, *nxt;

  Node(int d) : data(d), prv(nullptr), nxt(nullptr) {
    // This is constructor
    // will be called when `Node` is initialized
  }

  ~Node() {
    // This is distructor
    // will be called when `Node` is destroyed
  }
};

Node *ptr = new Node(87);
```

Encapsulation (封裝)

```
struct LinkedList {
  Node *head, *tail;

  LinkedList() : head(new Node(-1)), tail(new Node(-1)) {
    head->nxt = tail;
    tail->prv = head;
  }
  void ins_front(int data);
  void ins_back(int data);
  void rm_front();
  void rm_back();
  void print();
};

LinkedList mylist;
mylist.print();
```

偷懶時間 std::list

Modifiers

`clear`	clears the contents (public member function)
`insert`	inserts elements (public member function)
`insert_range` (C++23)	inserts a range of elements (public member function)
`emplace` (C++11)	constructs element in-place (public member function)
`erase`	erases elements (public member function)
`push_back`	adds an element to the end (public member function)
`emplace_back` (C++11)	constructs an element in-place at the end (public member function)
`append_range` (C++23)	adds a range of elements to the end (public member function)
`pop_back`	removes the last element (public member function)
`push_front`	inserts an element to the beginning (public member function)
`emplace_front` (C++11)	constructs an element in-place at the beginning (public member function)
`prepend_range` (C++23)	adds a range of elements to the beginning (public member function)
`pop_front`	removes the first element (public member function)
`resize`	changes the number of elements stored (public member function)
`swap`	swaps the contents (public member function)

都是O(1)

偷懶時間 std::list

Example

```cpp
// inserting into a list
#include <iostream>
#include <list>
#include <vector>

int main ()
{
  std::list<int> mylist;
  std::list<int>::iterator it;

  // set some initial values:
  for (int i=1; i<=5; ++i) mylist.push_back(i); // 1 2 3 4 5

  it = mylist.begin();
  ++it;       // it points now to number 2           ^

  mylist.insert (it,10);                        // 1 10 2 3 4 5

  // "it" still points to number 2                     ^
  mylist.insert (it,2,20);                      // 1 10 20 20 2 3 4 5

  --it;       // it points now to the second 20            ^

  std::vector<int> myvector (2,30);
  mylist.insert (it,myvector.begin(),myvector.end());
                                                // 1 10 20 30 30 20 2 3 4 5
                                                //               ^
  std::cout << "mylist contains:";
  for (it=mylist.begin(); it!=mylist.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

Edit & run on cpp.sh

https://cplusplus.com/reference/list/list/insert/

偷懶時間 std::list

```
list<int> mylist = {1, 2, 3};
mylist.push_front(10); // mylist = {10, 1, 2, 3}
mylist.pop_back();     // mylist = {10, 1, 2}
mylist.push_front(1);  // mylist = {1, 10, 1, 2}
list<int>::iterator it;

for (it = mylist.begin(); it != mylist.end(); it++) {
  if (*it == 1) { // delete 1 in the list
    it = mylist.erase(it);
  }
}
// mylist = {10, 2}

std::cout << "mylist = { ";
for (int i : mylist)
  std::cout << i << ", ";
std::cout << "};\n";
```

Description

Maintain a linked list, which supports the following operations:

- IH i : Insert head. Insert a new node with integer i to the head of the linked list.
- IT i : Insert tail. Insert a new node with integer i to the tail of the linked list.
- RH : Remove head. Remove the node at the head of the linked list. (If the linked list is empty, don't do anything.)
- RT : Remove tail. Remove the node at the tail of the linked list. (If the linked list is empty, don't do anything.)
- S i : Search. Traverse the linked list and find if there exists a node with integer i. If yes, print a line "Y". Otherwise, print a line "N". (If the linked list is empty, print a line "E".)
- O : Output. Traverse the linked list from head to tail. Print the integers saved in the nodes sequentially. (If the linked list is empty, print a line "E".)

```cpp
int n; cin >> n;
list<int> li;
while (n--) {
    string opt; cin >> opt;
    int i;
    if (opt[0] == 'I') {
        cin >> i;
        if (opt[1] == 'H') {   // "IH"
            li.push_front(i);
        } else {               // "IT"
            li.push_back(i);
        }
    } else if (opt[0] == 'R') {
        if (li.empty()) // do nothing
            continue;
        if (opt[1] == 'H') {   // "RH"
            li.pop_front();
        } else {               // "RT"
            li.pop_back();
        }
    } else if (opt[0] == 'S') {// "S"
        cin >> i;
        if (li.empty()) { cout << "E\n"; continue;}
        // traverse the linked list
        int ok = 0;
        for (auto it : li)
            if (it == i) ok = 1;

        cout << (ok ? "Y" : "N") << '\n';
    } else {                   // "O"
        if (li.empty()) { cout << "E\n"; continue;}
        for (auto it : li) cout << it << ' ';
        cout << '\n';
    }
}
```

in class sample codes

doubly linked list without dummy nodes:
http://codepad.org/k0O29XyO

doubly linked list with dummy nodes:
http://codepad.org/4dOldgAa

std::list code:
http://codepad.org/wD081pB9

Josephus Problem

Description

> Problem copied from cses.fi

Consider a game where there are n children (numbered $1, 2, \cdots, n$) in a circle. During the game, every second child is removed from the circle, until there are no children left. Counting begins at child 1 in the circle and proceeds around the circle in clockwise direction (where the number of the children is increasing except child n and child 1). In which order will the children be removed?

Input

The only input line has an integer n.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$

Josephus Problem sample codes

O(N log N) naive solution:

https://pastebin.com/t8CSj7md

O(N) lined-list solution:

https://pastebin.com/6fcvTRP1

Class Implementation

Basic Linked List Problem: https://ideone.com/0UqPGi

Josephus Problem:

- O(N log N) Naive Solution: https://ideone.com/C6GMaW
- O(N) Linked List Solution: https://ideone.com/rdSqw0
- O(N) std::list Solution: https://ideone.com/2Juj1I