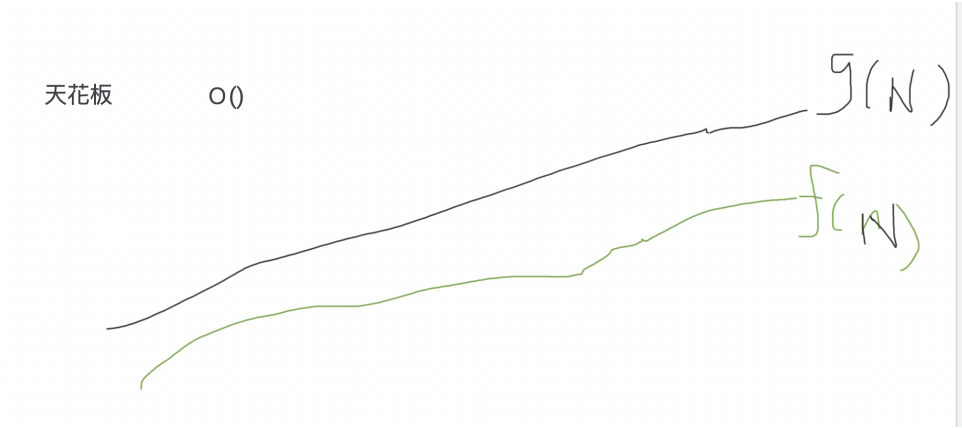
Phill-CPP-0823,0828

algorithm

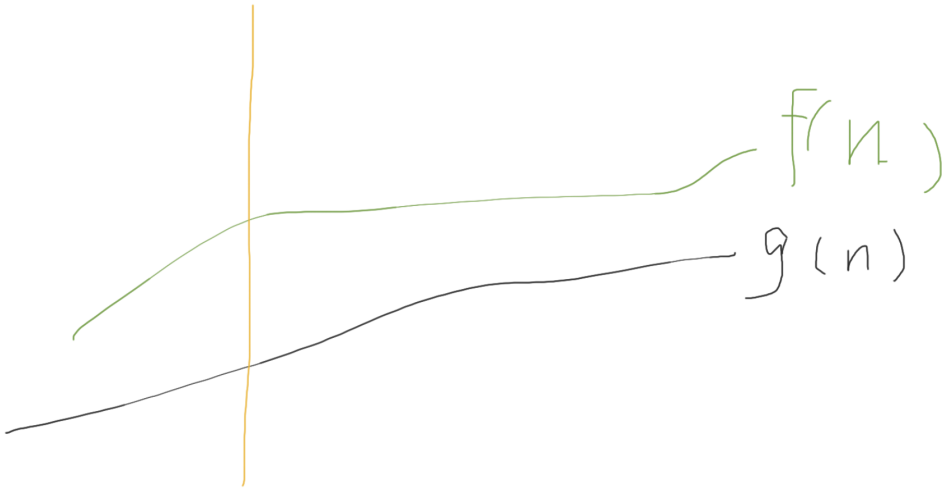
- able to be evaluation → performance evaluation → 1. time 2. space
- finite steps(有限指令)
- sequence (instruction sequence) 正確而有序的步驟指令解決問題
- A → P
  - P ‘s instance → A → solved correctly
  - deterministic (確定性)
  - halting problem → 有限步驟停機 → finite loop constraint
  - for each instance → A → solved
- performance
  - time complexity (時間複雜度)
  - 指令 → 分類基本指令
  - 運算次數統計法 → 1. 困難統計 2. non-deterministic
  - 運算次數統計 ←→ 輸入資料的量級
    - 不看運算指令數 → analysis function → f(x) → X:input size → y = time
    - f(100), f(100000)
  - time complexity → 輸入資料量級的函數
- 複雜度原則,符號
  - 複雜度上限(upper bound) → O → big O 天花板



f(n) = O(g(n))

→ C, N_0 → $\|f(n)\| \leq c\|g(n)\|, n \geq N_0$

- 複雜度下限(lower bound) → Ω → omega 地板



$f(n) = \Omega(g(n))$

→ C, N_0 → $\|f(n)\| \geq c\|g(n)\|, n \geq N_0$

- 複雜度等級
  - O(1) → 常數複雜度 → $g(n) = c \cdot 1$ → 神級演算法
  - $O(log_2 n)$ → 對數複雜度 → 蠻好的
  - O(n) → 線性複雜度 → reasonable
  - O(n log n) → 次線性複雜度
  - $O(n^2)$ → 平方複雜度 → 100 (10000 sec) → 100萬 (100萬*100 萬秒)

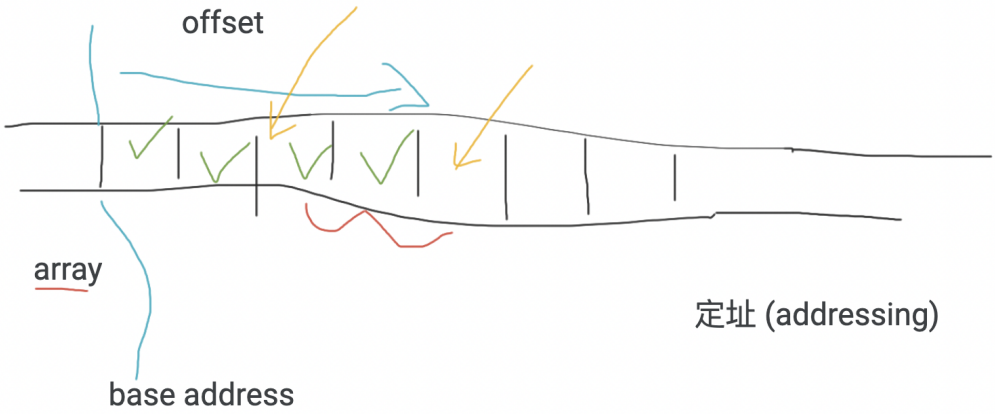
```
for (int i=0; i<=n; i++)
  for (int j=0; j<=m; j++){
      cout << i<< j<<endl;
  }
```

- $O(n^3)$
- $O(2^n)$ → 指數複雜度
- O(n!) → 階乘複雜度

Data Structure

Linear List 線性序列

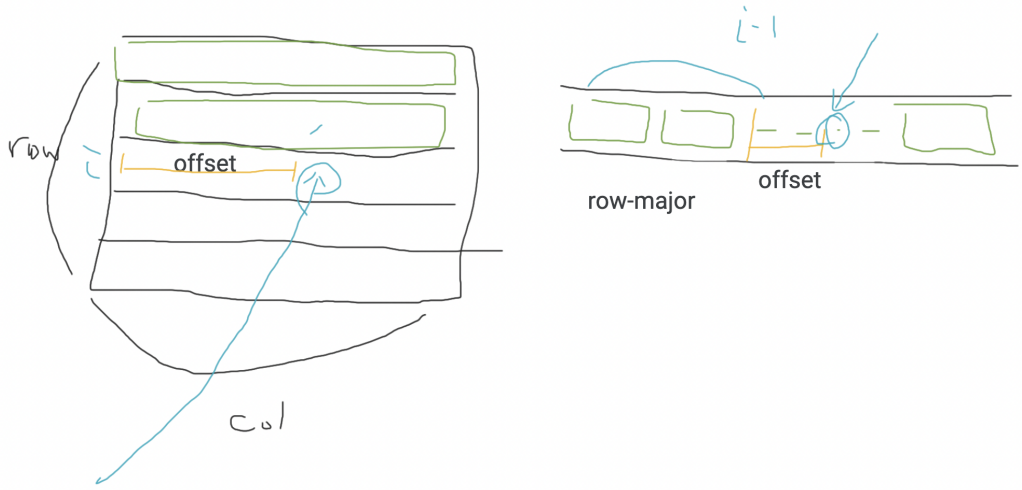
array



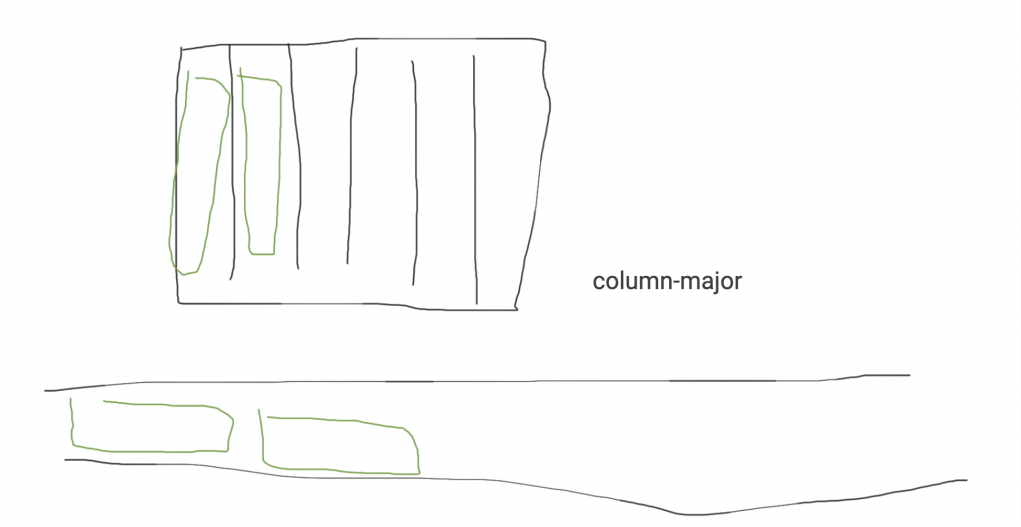
- 取值 O(1)
- 新增 插入 -> O(n)
- 刪除 O(n)
- 搜尋 O(n)

del a[2]
a[2] <- a[3]

- 二維線性 list 的配置方式(array)
  - row-major



- column-major



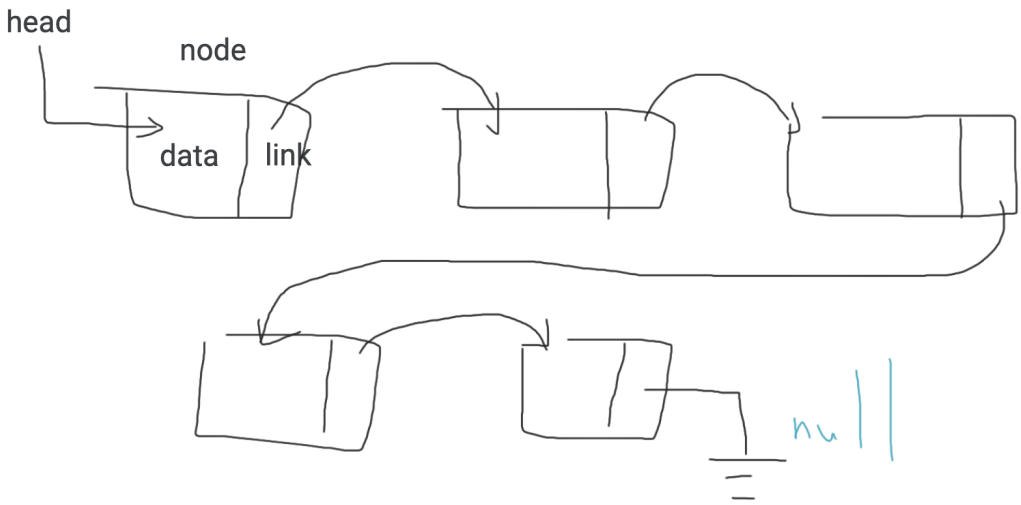
- 陣列

```
int arr[5] ={'a','b','c','d','e'};
cout << arr[0] << endl;
```

  - 線性關係的資料
  - 有序的資料 index 0→ ‘a’, 1 →’b’ …. 前後關係
    - 星期 → Mon Tue…
    - 撲克牌 → 2,3,…..,K,A
  - 取值 arr[index] → 0 index, 1 index → 循序對應

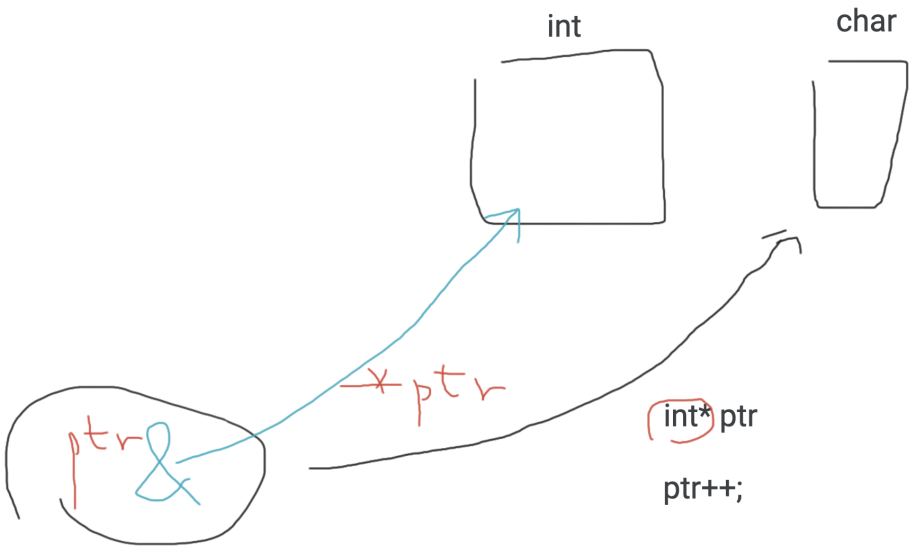
- 優缺點
  - 優點 → random access → 抽屜
  - 缺點 → 插入 刪除 → 循序對應 線性關係 → 沒有效率
- 應用
  - $f(x) = C_3 \cdot x^3 + C_2 \cdot x^2 + C_1 \cdot x^1 + C_0 \cdot x^0$
  - int poly[4]={c_3, c_2,c_1, c_0}

Linked List 鏈結串列



被 link 起來的元素節點 linked list

- node 節點 → data存資料 , link連結另一個 node
- link → pointer



比較一下 performance

access 元素 → Array O(1) , Linked List O(n)

尾巴加元素 → Array O(1), Linked List O(1)

開始或中間加元素 → Array O(n), Linked List O(1)

尾端刪元素 → O(1), O(1)

開始中間刪元素 → O(n), O(1)

尋找某個元素 → O(n), O(n)

→ O(n+1)=O(n)

實作

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

struct Node{
  int data;
  Node* next;
};

class LinkedList{
  private:
    Node* head;
```

```
  public:
    LinkedList(){
      head=NULL;
    }

    void add(int a){
      Node* newNode = new Node(data);
      if(head==NULL){
        head=newNode;
      }
      else{
        Node* current=head;
        while(current->next!=NULL){
          current = current -> next;
        }
        current -> next = newNode;
      }
    }
};

int main()
{

    return 0;
}
```