



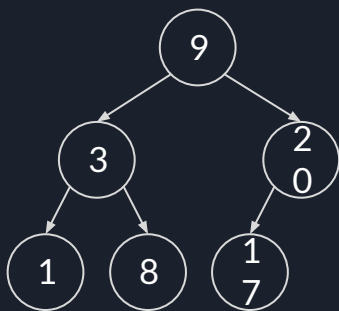
# 16. set

2024資訊研究社語法班  
Made with ❤️ by jheanlee

# set

set 是一種有序的關聯型容器，常被用在自動排序、查找及消除重複項目

set的本質是 平衡二元搜尋樹(balanced binary search tree)



平衡二元搜尋樹



固定大小  
fixed-size



動態  
dynamically allocated



## set 的特性

有序：set中所有項目都會被排序

集合：set的任何項目的key都是該項目本身

key的單一性：set中的key不會重複（因此項目也不會重複）

```
set<int> s;  
s.insert(1);  
s.insert(10);  
s.insert(2);  
s.insert(2);
```

```
cout << "s: ";  
for (int i : s) {  
    cout << i << " ";  
}  
cout << '\n';
```

```
s: 1 2 10
```



## set 的子函式

`set<T> variable_name;` -> 建構子

`set.empty()` -> 空true 或 有內容false

`set.size()` -> 回傳內容物的數量

`set.insert(val)` -> 插入一個值

`set.erase(val)` -> 移除一個值

`set.find(val)` -> 尋找值

找到的話回傳第一個找到位置iterator;找不到回傳`set.end()`

`set.count(val)` -> 回傳set中val的數量 (因為單一性所以只可能回傳0和1)

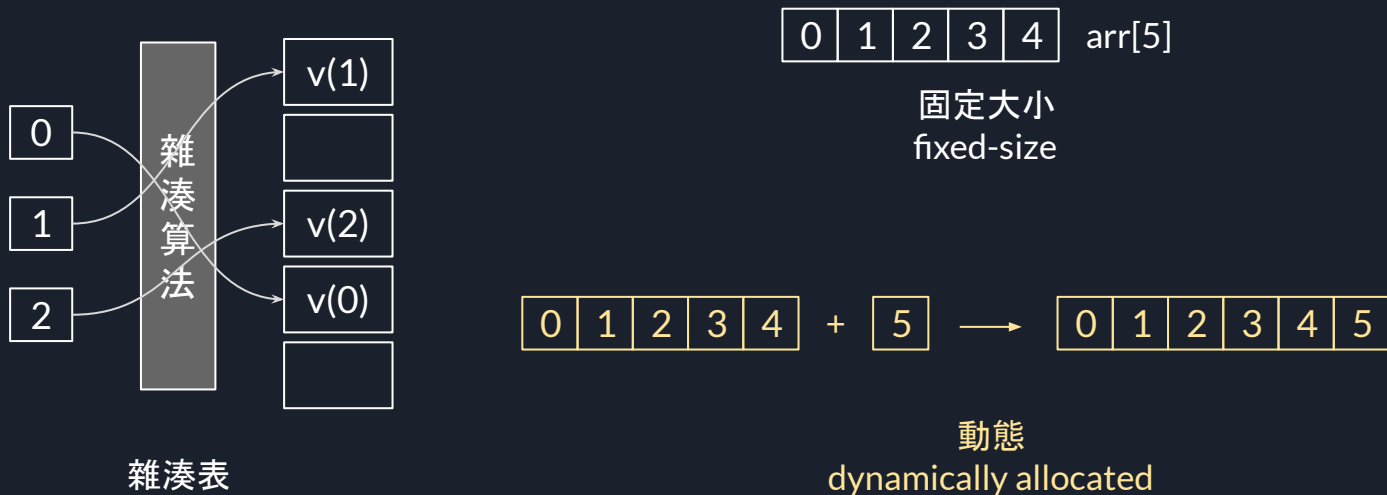
`set.begin()` -> 回傳第一個值的iterator

`set.end()` -> 回傳最後一個值的下一個位置的iterator

# unordered\_set

unordered\_set 是一種無序的關聯型容器，常被用於高效率的查找及移除重複項目

unordered\_set的本質是雜湊表(hash table)





## unordered\_set 的特性

有序：unordered\_set中所有項目會以雜湊算法的值分類而失去順序性

集合：unordered\_set的任何項目的key都是該項目本身

key的單一性：unordered\_set中的key不會重複（因此項目也不會重複）

```
unordered_set<int> us;  
us.insert(1);  
us.insert(10);  
us.insert(2);  
us.insert(2);
```

```
cout << "us: ";  
for (int i : us) {  
    cout << i << " ";  
}  
cout << '\n';
```

```
us: 2 10 1
```



## set 的子函式

`unordered_set<T> variable_name;` -> 建構子

`unordered_set.empty()` -> 空`true` 或 有內容`false`

`unordered_set.size()` -> 回傳內容物的數量

`unordered_set.insert(val)` -> 插入一個值

`unordered_set.erase(val)` -> 移除一個值

`unordered_set.find(val)` -> 尋找值

找到的話回傳第一個找到位置`iterator`;找不到回傳`unordered_set.end()`

`unordered_set.count(val)` -> 回傳`unordered_set`中`val`的數量

(因為單一性所以只可能回傳0和1)

`unordered_set.begin()` -> 回傳第一個值的`iterator`

`unordered_set.end()` -> 回傳最後一個值的下一個位置的`iterator`



# LeetCode 217

## Contains Duplicate

回傳nums中是否有重複的項目

(練習使用set 或 unordered\_set解)

**Input:** nums = [1,2,3,1]

**Output:** true




## 解法一 (count/find - 效率不佳)

```
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        unordered_set<int> s;

        for (int i:nums) {
            if (s.count(i)) {
                return true;
            } else {
                s.insert(i);
            }
        }

        return false;
    }
};
```



## 解法二 set.size() vs vector.size()

運用set的值不重複的特性, 只要set的大小與vector的大小不同就是有重複

```
class Solution {  
public:  
    bool containsDuplicate(vector<int>& nums) {  
        unordered_set<int> s;  
  
        for (int i : nums) {  
            s.insert(i);  
        }  
  
        return s.size() != nums.size();  
    }  
};
```



## 解法三 - (非set) sort()後逐項比大小

這個方法效率會比set好, 因為set/unordered\_set是關聯型容器, 在新增新的值的時候效率較差

set:  $O(\log n)$

unordered\_set:  $O(1)$  (但有hash的步驟)

vector:  $O(1)$

```
class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        sort(nums.begin(), nums.end());

        for (int i = 1; i < nums.size(); i++) {
            if (nums[i] == nums[i - 1]) return true;
        }

        return false;
    }
};
```