STL & pb_ds

NCTU_Cobra 呂爾軒 鄧佰理

STL (Standard Template Library)

- 各式容器
- 利用疊代器造訪

容器 Containers

- vector
- queue deque(double-ended-queue)
- priority_queue
- pair
- stack
- map multimap unordered_map
- set multiset unordered_set
- list
- bitset

pair

- pair<t1,t2>,兩個不同類型(也可以同樣)的變數和在一起的一個class

- 有內建的比較(如果t1,t2都有比較operator的話),先比前再比後

vector

- 跟array非常相似(vector<int>num vs int num[size])
- 可動態控制容量不須於宣告決定
- 可多層 (vector<vector<int>>)也可包其他容器或自定義的struct(vector<pair<int,int>>)
- 因為不像陣列是指標，傳vector用pass by value會炸時間、記憶體

vector 基本操作

- 宣告 vector<type>ID (v)
- v.push_back(num) 將num加入vector尾
- v.pop_back() 移除最尾項
- v[index] reference index項之值 和array用法相同

vector

- insert O(1) (v.push_back(num))
- lookup O(1) (v[index])
- deletion O(1) (v.pop_back())
- lower_bound O(lgN) (lower_bound(v.begin(),v.end(),value))

vector用途

- 許多情況可替代array
- 存graph (adjacency lists)
- 使用內建function ex: lower_bound　upper_bound　　reverse

疊代方式

- for(int i=0 ;i < v.size() ; i++){
  //do something of v[i]
  }

- for(auto i : v){
  //do something of i
  }

lower_bound upper_bound 功能

- 僅能用於sort過後的container(sort(v.begin(),v.end()))
- 為內建的二分搜功能
- 用法 lower(upper)_bound(v.begin(),v.end(),value)
- 回傳值為 lower(第一個大於等於) upper(第一個大於) value之pointer
- 若不存在回傳end()
- 和.begin()相減即可找到index為第幾個

vector二分搜 例題

- https://codeforces.com/problemset/problem/975/C
- https://codeforces.com/problemset/problem/978/F
- https://codeforces.com/contest/1077/problem/E

deque

- 和vector很像，但是頭尾都可以進行push和pop
- 內部記憶體區段不連續

deque

- insert O(1) (push_back() ,push_front())
- lookup O(1) (v[index])
- deletion O(1) (pop_back(),pop_front())
- lower_bound O(lgN) (lower_bound(v.begin(),v.end(),value))

vector, deque其他內建功能

- size():回傳目前vector或deque裡面有幾個element
- empty():看vector或deque size是否=0
- clear():清空vector或deque
- insert(), erase():在vector或deque中間插值或刪值*

*注意：時間O(N)

map

- index和value 一對一對應
- 功能相近於index可自行定義的array 且index不須連續
- 實作上為二元樹狀結構
- index為需要才創建 -> map大小動態 查找時間也隨規模增加

map 基本操作

- 宣告 map<Type1,Type2>ID (mp)
- mp[num1]=num2 將 index num1的value 設為num2
- mp[num1] reference 值

map

- insert O(lgN) (mp[index] = value)
- lookup O(lgN) (num = mp[index])
- deletion O(lgN) (mp.erase(index))

map例題

https://oj.nctu.me/problems/128/ (找有幾段連續項和為0)

map lookup問題

- 當lookup 某index本來在map之中無該項 即會將該項value設成0 並回傳 (某些情況會大量提高 map tree的size)
- lgN 查找時間有明顯提高 測資時限緊
- https://codeforces.com/contest/1029/problem/D

解決方法 好習慣

```
if(map.find(num1)!=map.end()){
        //do reference
}
```

當查到空項不會增加map.size()

unordered_map

- 功能大致和map相似
- 實作方式不同 非樹狀結構 而是hash
- 在規模較大時平均查找速度較map快
- 注意hash function的速度

multimap, unordered_multimap

- 功能分別和map, unordered_map相似，但同一個index值可以存多項element

set

- 集合 存值 一般功用為查找某值是否在此容器中
- 類似於index=value的map
- 樹狀結構
- 自動排序功能 (預設為 less than 也可自定義 若為struct 則需自定義 operator <)

set 基本操作

- 宣告 set<Type> ID(st)
- 加入 st.insert(num)
- 查找 st.find(num)==st.end()? “Not Exist”:”Exist”;
- *st.begin() *(--st.end()) 造訪自動排序後的最前最後項

Complexity

- insert O(lgN)
- deletion O(lgN)
- lookup O(lgN)

set 之lower_bound upper_bound

- 用法 st.lower_bound(key)
- 若使用lower_bound(st.begin(),st.end(),key)則會掃過整個set 不是lgN查找

multiset

- 相同的值可以存在多個的set
- 數據規模大時效率較set好一些

unordered_set, unordered_multiset

- 功能一樣分別類似set和multiset，但使用hash而非tree來儲存和查找

iterator

- 用來指向container中的元素
- begin()指向第一個元素，end()指向最後一個元素後面一個空element
- 每個iterator都可用 ++來指向下一個元素
- 可以用*iterator來呼叫元素的值
- 注意iterator種類!操作有不同
- vector, deque 的iterator可以+ n, -n,比較等, stack和map的iterator不行

queue

- 為用來做一系列 取出 插入 操作的容器
- 先進先出

queue 基本操作

- 宣告 queue<Type>ID(q)
- q.front() reference 首項(最先進入的)
- q.pop() 首項自queue中移除
- q.push(num) 將num丟到queue最尾
- q.empty() 查詢是否為空

Complexity

- insert O(1)
- lookup O(N)
- delete O(1)

queue用途

- BFS (graph為例)

```
queue<int>q;
q.push(start);
while(!q.empty()){
        int now=q.front();q.pop();
        //do something
        for(auto i:g[now]){
                q.push(i);
        }
}
```

stack

- 為用來做一系列 取出 插入 操作的容器
- 後進先出

queue 基本操作

- 宣告 stack<Type>ID(q)
- q.top() reference 頂端(最後進入的)
- q.pop() 頂端項自stack中移除
- q.push(num) 將num丟到stack頂
- q.empty() 查詢是否為空

Complexity

- insert O(1)
- lookup O(N)
- delete O(1)

stack例題

- https://e-tutor.itsa.org.tw/e-Tutor/mod/programming/view.php?id=40905 (後序運算法)
- https://codeforces.com/contest/1092/problem/D1 (判奇偶匹配)

bitset

- 相當於只能存0,1的array
- 省記憶體 單元僅需1bit (int為32 bit,long long為64bit)
- 可進行bitwise操作 | & ^
- 可進行shift操作 <<, >>

bitset基本操作

- bitset<size>ID(b)
- b<<=10 向左shift
- b |= (b<<10) 向左shift再進行bitwise or操作

bitset用途

- 背包問題
- subsetsum問題
- https://oj.nctu.me/problems/484/

subset sum問題 (partition knapsack)

```
bitset <10005>b;
    b[0]=1;
    for(int i=0;i<n;i++)   {
        int add;
        cin>>add;
        b|=(b<<add);
    }
```

priority_queue

- priority_queue, 取出最大的元素
- 宣告:priority_queue<type,std::vector<type>, comp<type>>()可以選類型，裝的container type,還有比較用的function,constructor在後面括號可以選定要用的container
- push()放入，top()看目前最大的元素，pop()取出
- empty()看是否為空的
- push(),pop()時間複雜度O(logn)

std::priority_queue實踐

- stl的priority_queue本身不支援元素更新

- 解決方法之一(像用在dijkstra's的時候)：每次要更新的時候，都放一項新的元素進去，每次處理的時候檢查，只處理有更新到最新的那一項。

其他c++有用function

- sort(), stable_sort()：從小到大排序，可以寫自定義的比較function，stable_sort()有保證stable
- auto:可以自動根據設的值選擇變數類型
- next_permutation(),prev_permutation()：照字典序產生下一個排序或前一個排序

string

- stl的string
- 有size(), begin(), end(), push_back(), pop_back(), empty()等function
- 內建有+(兩個字串相接)還有比較operator(用字典序比較)
- 其他function：c_str():產生一個對應這個string的cstring
- substr(pos,len):產生一個stl string的substring

pb_ds

- pb_ds:policy based data structure
- implement在gnu c++中

pb_ds使用

- #include <ext/pb_ds/assoc_container.h>
- using namespace __gnu_pbds;
- 基本container: tree, trie, cc_hash_table, gp_hash_table, list_update, priority_queue
- 每個container有不同的設定
- 範例
- Documentation