# 2015年厦门大学ACM寒假集训五（0212）

考虑到大家觉得进度太快，这次我们先把数学期望和概率、矩阵放到下次学，这次学学STL好了。

这次我们学学C++ 中的STL吧，STL类型很多，我附带一份STL的pdf，供大家参考。

在这里，我们只讲常用的可以带来极大便利的类型。关于栈、队列这些我们开一个数组就可以实现的数据结构，没必要再去记录那些繁杂的STL函数，所以这类的不讲。

头文件如下：

#include<vector> //vector<T> a;

#include<queue> //我们只讲priority\_queue<T>，优先队列（可认为是堆）

#include<set> //set<T>，multiset<T>集合

#include<map> //使用上和set诸多相似，也很有用，但不讲。

using namespace std; //命名空间，注意加上去。

1. **Vector容器**

功能：相当于一个**大小会自动调节**的数组，可以用a[i]直接访问。

解决的问题：我们会遇到这样的图论题，一开始都是：给定一个n个点(n<=100000)，m条边(m<=200000)的图.....

那么首先我们要存储这个图，并且能快速访问某个点连出去的边。

现在问题来了：有些点可能有n-1条边，有些点可能只有1条边，数组都是定长的，开一个类似a[100000][100000]的数组明显会超内存。

我们要做的就是：一个点有多少边，我们就给它多少内存，怎么实现呢？

1. 链表来实现的。
2. **vector<int> a[100010]**这样实现。

常用的函数：（从来没用过的我就不写了，有兴趣看附带的那个pdf文件）

以vector<int> a为例

push\_back()：在数组末端插入1，下标从0开始。

size() :返回a中元素个数。

clear() :清空a中元素。

vector<>数组大小自动调整的原理：当前大小为k时，此时要插入第k+1个数，那么它会自动申请一个2\*k大小的空间，然后将这k+1个数已过去，释放原来的空间。大小一般为2的幂1,2,4,8,16,32...

下面是一段小程序:

a.clear();//清空a中的元素。

a.push\_back(3);

a.push\_back(1);

a.push\_back(2); //插入3,1,2，按顺序输出

for (int i=0;i<a.size();i++)printf(“%d “,a[i]);

1. **priority\_queue 容器**

功能：优先队列，**实现堆的功能**

解决的问题：我们知道，一个堆实现起来挺麻烦的，这个应该可以完全替代手写堆。

常用的函数：

push() : 插入一个元素

top() : 返回堆顶元素

pop() : 删除堆顶元素

empty():返回堆是否为空

size() : 返回堆中元素个数

给段小程序说明一下：

//自写结构体的优先级设定还要复杂一点，考虑你们接受能力，就用int演示一下

#include<iostream>

#include<queue>

using namespace std;

int i,j,k,m,n;

priority\_queue<int> a;

int main(){

a.push(3);

a.push(1);

a.push(2); //插入3，1，2这3个元素

printf("size=%d\n",a.size()); //输出当前堆中的元素

while (!a.empty()){ //输出每一个元素

printf("%d ",a.top());

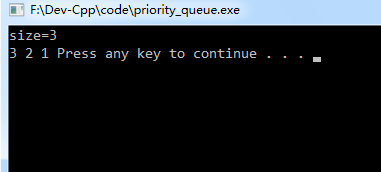
a.pop(); //a.top()并不删除堆顶元素，需要调用a.pop();删除

}

return 0;

}

运行结果：int默认情况下，数值高的优先级大。故输出3，2，1



**3、set 容器**

功能：可以直接理解为一个集合（set），可以在log(n)内，往集合中插入、删除、查找元素

说明一下：

1. set 中，2次插入同一个元素时，只记录一个。也就是集合元素不重合。

要想记录重合的元素，要用multiset 。

1. set无法实现“名词树”的功能，也就是说对这类询问无能为力(无法log(n)实现)：
2. 求第k个元素
3. 求值为k的名词。（比<k的结点个数+1）

遇到了，一般只能手写平衡树。

常用函数：

begin(): 返回指向第一个元素的迭代器。迭代器的概念我们后面例子说。

end()：返回指向最后一个元素**再往后一个**的迭代器。注意它不指向任何一个我们插入的元素!

empty():判断是否为空

erase() :删除元素

count():返回元素的个数（在set中，返回值非0即1，在mutiset中，才可能>1）;

lower\_bound():返回指向大于（或等于）某值的第一个元素的迭代器

upper\_bound(): 返回大于某个值元素的迭代器

size() :返回元素个数。

迭代器概念：可以理解为一个指针。

迭代器的声明：<set>::iterator x,y;

#include<iostream>

#include<set>

using namespace std;

set <int>a;

set <int>::iterator it; //声明一个迭代器

int main() {

a.insert(1);a.insert(1); //1插入两次，看看结果怎么样。

a.insert(3);

a.insert(2);

cout<<"count(1) = "<<a.count(1)<<endl;

cout<<"size = "<<a.size()<<endl; //此时会输出3，两个1只记录一个。

for (it=a.begin(); it!=a.end(); it++) //begin()和end()是好基友。

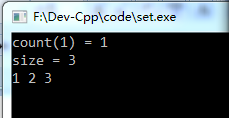
//当it==a.end()时，不处理,it++表示指向下一个元素，同理有it--;

cout<<\*it<<" "; //和指针一样，前面加\*取值。

return 0;

}

运行结果：



另外

1、关于map,multiset,multimap 大家自己看。大同小异。

2、当我们自定义结构体时，需要自己写大小判断的函数。

类似这样：

struct point{

Int x,y;};

bool operator < (point a,point b) { //定义< 操作。

Return a.x<b.x||(a.x==b.x&&a.y<b.y);

}

**Problem 1 poj 2631**

简要描述：求树的直径。即求树上2个点，使它们距离最远。

**Description**

Building and maintaining roads among communities in the far North is an expensive business. With this in mind, the roads are build such that there is only one route from a village to a village that does not pass through some other village twice.   
Given is an area in the far North comprising a number of villages and roads among them such that any village can be reached by road from any other village. Your job is to find the road distance between the two most remote villages in the area.

The area has up to 10,000 villages connected by road segments. The villages are numbered from 1.

**Input**

Input to the problem is a sequence of lines, each containing three positive integers: the number of a village, the number of a different village, and the length of the road segment connecting the villages in kilometers. All road segments are two-way.

**Output**

You are to output a single integer: the road distance between the two most remote villages in the area.

**Sample Input**

5 1 6

1 4 5

6 3 9

2 6 8

6 1 7

**Sample Output**

22

**Problem 2 xmu 1514**

关于xmu无法提交的话，因为这题代码量也就30行，重在思想。

取数游戏2

**Description**

给定一个由N个数字构成的序列，并且定义一个数字序列中的M值为这个序列中任意两个数字的差值的最大值，一个数字序列中的m值为任意两个数字的差值的绝对值的最小值。现在的问题是，让你移除K个数字，使得剩余的N-K个数字构成的序列的M+m值最小。

**Input**

第一行两个整数N（3<=N<=1,000,000）、K（1<=K<=N-2），为初始给定的序列长度以及要移除的数字个数。

接下来一行N个整数Vi（-5,000,000<=Vi<=5,000,000），代表初始序列的每个数的值。

**Output**

输出为一个整数，代表移除K个数字之后剩余序列M+m的最小值。

**Sample Input**

6 2

-5 8 10 1 13 -1

**Sample Output**

13

**Source**

Xmu

*提示：需要用到一点贪心技巧，加上set优化一下。*

**Problem 3 uva 1203**

<http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=497&page=show_problem&problem=3644>

试试使用优先队列解决这题。

Argus

问题描述：你的任务是编写一个称为Argus的系统。该系统支持一个register命令：

Register q\_num period

该命令注册一个触发器，它每period就会产生一次编号为q\_num的事件。

你的任务：模拟前k次事件，多个事件如果同时发生，先处理q\_num小的事件。

Input

The first part of the input are the register instructions to Argus, one instruction per line. You can assume the number of the instructions will not exceed 1000, and all these instructions are executed at the same time. This part is ended with a line of #.

The second part is your task. This part contains only one line, which is one positive integer K (≤ 10000).

Output

You should output the Q\_num of the first K queries to return the results, one number per line.

Sample Input

Register 2004 200

Register 2005 300

#

5

Sample Output

2004

2005

2004

2004

2005

以上3题把vector ,priority\_queue，set都用过一遍了。后面就来点简单题吧。

另外：博弈论中双方走的都是最优策略。

**Problem 4 hdu 2147**

问题描述：

就是有一个游戏，在一个n\*m的矩阵中起始位置是（1，m），走到终止位置（n,1）；游戏规则是只能向左，向下，左下方向走，想走到终点的为获胜者。

Input

Input contains multiple test cases. Each line contains two integer n, m (0<n,m<=2000). The input is terminated when n=0 and m=0.

Output

If kiki wins the game printf "Wonderful!", else "What a pity!".

Sample Input

5 3

5 4

6 6

0 0

Sample Output

What a pity!

Wonderful!

Wonderful!

*提示：这道题可以用DP实现。需要开2000\*2000的数组，用int会超空间（只提供10MB空间）。怎么办呢？*

*另外还有简洁的结论，不过最好无视它，用DP写。*

**Problem 5 poj 1088 经典的滑雪问题。**

滑雪

问题描述：Michael喜欢滑雪百这并不奇怪， 因为滑雪的确很刺激。可是为了获得速度，滑的区域必须向下倾斜，而且当你滑到坡底，你不得不再次走上坡或者等待升降机来载你。Michael想知道载一个区域中最长底滑坡。区域由一个二维数组给出。数组的每个数字代表点的高度。下面是一个例子

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

一个人可以从某个点滑向上下左右相邻四个点之一，当且仅当高度减小。在上面的例子中，一条可滑行的滑坡为24-17-16-1。当然25-24-23-...-3-2-1更长。事实上，这是最长的一条。

Input

输入的第一行表示区域的行数R和列数C(1 <= R,C <= 100)。下面是R行，每行有C个整数，代表高度h，0<=h<=10000。

Output

输出最长区域的长度。

Sample Input

5 5

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

Sample Output

25