/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nth\_element

O(logN)

附STL中nth\_element方法说明

(测试代码 ： <https://paste.ubuntu.com/p/KCMHJ7Pq2b/> )

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <algorithm>

using namespace std;

/\* 使[l, r] 中下标为k的元素有序，且[l, k)中元素均不大于k处， (k, r]中元素均不小于k处 \*/

void mynth\_element(int l, int r, int k, int v[])

{

/\* 此处快排 \*/

int i = l, j = r + 1;

while (1)

{

while (v[l] > v[++i]);

while (v[l] < v[--j]);

if (i >= j)

break;

swap(v[i], v[j]);

}

swap(v[l], v[j]);

/\* 判断分治区间 \*/

if (j > k)

mynth\_element(l, j - 1, k, v);

else if (j < k)

mynth\_element(j + 1, r, k, v);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nth\_element函数原型:

使 nth 指向的元素有序， 且first到nth间元素均不大于\*nth， nth到last之间元素均不小于\*nth

template <class Iter>

void nth\_element (Iter first, Iter nth,

Iter last);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

comp函数定义： bool comp(const Iter a, const Iter b);

返回true表示\*a应排在\*b前面

template <class Iter, class Compare>

void nth\_element (Iter first, Iter nth,

Iter last, Compare comp);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/