8.STL

Vector

1. 初始化
   - 数组初始化 int a[5] = {1,2,3,4,5}
   - vector初始化 vector<int> A[10] = {1,2,3,4,5}
   - vector<int> A(10, 1);
1. 二维数组初始化赋值
   - vector<vector<int>> f1(n, vector<int>(m,0));
1. 常用函数
   - 排序：sort(nums.begin(), nums.end());
   - 排序：sort(nums.begin(), nums.end(), cmp);
   - 逆排：sort(nums.begin(), nums.end(), greater<int>());
   - cmp是一种比较方法，static bool cmp()
   - sort(courses.begin(), courses.end(), [](const auto& a, const auto& b) { return a[1] < b [1];});
   - 插入：nums.insert(nums.begin(), x);
   - 插入：nums.insert(nums.begin() + n , x);
   - 删除：nums.erase(nums.begin());
   - 求和：accumulate(v.begin(), v.end(), 0);
   - 反转：reverse(s.begin(), s.begin() + i - 1); //左闭右开区间
   - 返回元素个数：size()
   - 返回是否为空：empty()
   - 清空：clear()
   - 末尾添加/删除：push_back()/pop_back()
   - front()/back()
   - 返回开始和结束的迭代器：begin()/end()
   - []
   - 支持比较运算（>,<），按字典序
   - erase() (1) 输入是一个数x，删除所有x O(k + logn) k是x的个数
   - 去重 alls.erase(unique(alls.begin(), alls.end()), alls.end());
   - 查找 -- vector中的find()

vector<int>::iterator result = find(arr2.begin( ), arr2.end( ), arr1[i]);

if (result == arr2.end( )) //如果没找见

1. 遍历
   - for(vector<int>::iterator i = a.begin(); i != a.end(); i++) cout << *i << endl;
   - for(auto i = a.begin(); i != a.end(); i++) cout << *i << endl;
   - for(auto x : a) cout << x << endl;

1. 参数
   - int rows=triangle.size();//求得行数
   - int col=triangle[0].size();//求的列数

String

1. 常用函数
   - 返回字符串长度：size()/length()
   - 插入：str.insert(str.begin(), 'a')
   - 返回子串：str.substr(起始坐标，(长度))
   - 字符串添加元素：str.push_back('a')
   - 字符串删除末尾元素：str.pop_back('a')
   - 删除元素 str.substr(0, str.length() - 1);
   - 删除元素 str.erase(str.end() - 1);
   - 返回字符串所在字符数组的起始地址： c_str();
   - 为空/清空 empty()/clear();
   - 改变大小 resize();
1. 整型和字符串的相互转换
   - 整型转字符串
   - to_string(i)
   - 字符串转整型
   - stoi头文件：<string>,c++函数
   - atoi头文件:<cstdlib>,c函数
   - int a=atoi(s.c_str());
   - int b=stoi(s);

set（没有重复元素）

- set遍历

1. set<int>::iterator it;
2. for(it=notAppearSet.begin ();it!=notAppearSet.end ();it++) cout << *it;

- 查找

s.find() 查找一个元素，如果容器中不存在该元素，返回值等于s.end()

if(numSet.find(findNum)!=numSet.end()
代表找到了

multiset（可以有重复元素）

map

- map<int, int> loc;
- map a["yxc"] = 1
- 获取键、值 for(auto x : f1) x.first,x.second
- map.find()查找的是key

set, map, multiset, multimap（基于平衡二叉树（红黑树），动态维护有序序列）

- size()
- empty()
- clear()
- begin()/end()
- ++, -- 返回前驱（前一个数）和后继（后一个数），时间复杂度 O(logn)
- set/multiset
- insert() 插入一个数
- find() 查找一个数
- count() 返回某一个数的个数
- erase()

(1) 输入是一个数x，删除所有x O(k + logn)

(2) 输入一个迭代器，删除这个迭代器

- lower_bound()/upper_bound()

lower_bound(x) 返回大于等于x的最小的数的迭代器

upper_bound(x) 返回大于x的最小的数的迭代器

- map/multimap

insert() 插入的数是一个pair

erase() 输入的参数是pair或者迭代器

find()

[] 注意multimap不支持此操作。 时间复杂度是 O(logn)

lower_bound()/upper_bound()

unordered_set,unordered_multiset,unordered_map,unordered_multimap

- 和上面类似，增删改查的时间复杂度是 O(1)
- 不支持 lower_bound()/upper_bound()， 迭代器的++，--
- key不能是vector<int>等复合类型，因为没法计算hash，map可以，map<vector<int>, int>m

queue

- 没有clear(), 清空：q = queue<int>()
- size()
- empty()
- push() 向队尾插入一个元素
- front() 返回队头元素
- back() 返回队尾元素
- pop() 弹出队头元素

priority_queue（优先队列）

- 优先队列具有队列的所有特性，包括队列的基本操作，只是在这基础上添加了内部的一个排序，它本质是一个堆实现的。
- 定义priorith_queue<类型, vector<类型>, cmp>;

//升序队列，小顶堆

- priority_queue <int,vector<int>,greater<int> > q;

//降序队列，大顶堆

- priority_queue <int,vector<int>,less<int> >q;

//自定义排序

```
auto cmp = [&](const pair<int, int>& x, const pair<int, int>& y) {
    return arr[x.first] * arr[y.second] > arr[x.second] * arr[y.first];
};

priority_queue<PII, vector<PII>, decltype(cmp)> heap(cmp);
```

```
struct cmp{
    operator()(const int& a, const int&b){
        return ...;
    }
}
priority_queue<PII, vector<PII>, cmp> heap;
```

//greater和less是std实现的两个仿函数（就是使一个类的使用看上去像一个函数。其实现就是类中实现一个operator()，这个类就有了类似函数的行为，就是一个仿函数类了）

- 没有clear()
- size()
- empty()
- push() 插入一个元素
- top() 返回堆顶元素
- pop() 弹出堆顶元素
- emplace() 插入元素
- 大根堆插入-x就是小根堆
- 定义成小根堆的方式：priority_queue<int, vector<int>, greater<int>> q;

deque（效率慢）

- size()
- empty()
- push() 向队尾插入一个元素
- front() 返回队头元素
- back() 返回队尾元素
- pop() 弹出队头元素

pair<int, int>

1. 定义
   - pair底层是结构体
   - pair<int, string> 存储一个二元组
   - pair<int, pair<int, int>> 三元组
1. 赋值/初始化
   - p = make_pair(10, "yxc")
   - p = {10, "yxc"}
1. 取到元素
   - p.first
   - p.second

stack

- size()
- empty()
- push() 向栈顶插入一个元素
- top() 返回栈顶元素
- pop() 弹出栈顶元素

bitset

- bitset<10000> s;
- ~, &, |, ^
  - , <<
- ==, !=
- []
- count() 返回有多少个1
- any() 判断是否至少有一个1
- none() 判断是否全为0
- set() 把所有位置成1
- set(k, v) 将第k位变成v
- reset() 把所有位变成0
- flip() 等价于~
- flip(k) 把第k位取反

auto 自动推断类型

- auto x = max_element(a.begin() + i, a.end());用的时候*x

小知识点

typedef pair<int,int> PII;

vector<PII> x;

x. push_back({l,r});

x.first