**C++；**

**一、string转int的方式**

1. 采用最原始的string, 然后按照十进制的特点进行算术运算得到int，但是这种方式太麻烦，这里不介绍了。
2. 采用标准库中atoi函数。

string s = "12";   
int a = atoi(s.c\_str());   
对于其他类型也都有相应的标准库函数，比如浮点型atof(),long型atol()等等。

1. 采用sstream头文件中定义的字符串流对象来实现转换。

istringstream is("12"); //构造输入字符串流，流的内容初始化为“12”的字符串   
int i;   
is >> i; //从is流中读入一个int整数存入i中

**二、int转string的方式**

1. 采用标准库中的to\_string函数。

int i = 12;   
cout << std::to\_string(i) << endl;   
不需要包含任何头文件，应该是在utility中，但无需包含，直接使用，还定义任何其他内置类型转为string的重载函数，很方便。

1. 采用sstream中定义的字符串流对象来实现。

ostringstream os; //构造一个输出字符串流，流内容为空   
int i = 12;   
os << i; //向输出字符串流中输出int整数i的内容   
cout << os.str() << endl; //利用字符串流的str函数获取流中的内容   
字符串流对象的str函数对于istringstream和ostringstream都适用，都可以获取流中的内容。

字典树（TriTree）

https://www.cnblogs.com/TheRoadToTheGold/p/6290732.html

关于字典树的用处：

查找单词前缀、数字运算中XOR求最大值（将所有的数字变成32位字符串，求得XOR最大值即为看看哪些位上不同）、

二叉树的先序 中序 后序遍历 非递归

preorder:

vector<**int**> preorderTraversal(TreeNode\* root) {

vector<**int**> res;

std::stack<TreeNode\*> temp;

**while** (root || !temp.empty()) {

**while** (root) {

temp.push(root);

res.push\_back(root->val);

root = root->left;

}

root = temp.top();

temp.pop();

root = root->right;

}

**return** res;

}

inorder:

vector<**int**> inorderTraversal(TreeNode\* root) {

vector<**int**> res;

std::stack<TreeNode\*> temp;

**while** (root || !temp.empty()) {

**while** (root) {

temp.push(root);

root = root->left;

}

root = temp.top();

temp.pop();

res.push\_back(root->val);

root = root->right;

}

**return** res;

}

postorder:（利用先序（root right left）遍历 然后翻转结果）

vector<**int**> postorderTraversal(TreeNode\* root) {

vector<**int**> res;

std::stack<TreeNode\*> temp;

**while** (root || !temp.empty()) {

**while** (root) {

temp.push(root);

res.insert(res.begin(),root->val);

root = root->right;

}

root = temp.top();

temp.pop();

root = root->left;

}

**return** res;

}

# 蓄水池抽样（Reservoir Sampling）

在未知长度的情况下从中等概率抽取k个数

在一个给定长度的数组中随机等概率抽取一个数据很容易，但如果面对的是长度未知的海量数据流呢？蓄水池采样(Reservoir Sampling)算法就是来解决这个问题的, 它在分析一些大数据集的时候非常有用。

**1. 算法描述**

1. 先选取数据流中的前k个元素，保存在集合A中；
2. 从第j（k + 1 <= j <= n）个元素开始，每次先以概率p = k/j选择是否让第j个元素留下。若j被选中，则从A中随机选择一个元素并用该元素j替换它；否则直接淘汰该元素；
3. 重复步骤2直到结束，最后集合A中剩下的就是保证随机抽取的k个元素。

Init : a reservoir with the size： k

for i= k+1 to N

    M=random(1, i);

    if( M < k)

     SWAP the Mth value and ith value

end for

创造一个 插入删除以及随机选取的数据结构，要求每次操作o(1)

随机选取必须要vector,下面问题就在于 如何让vector的插入删除保证在o(1),考虑到在在数组的末尾插入删除满足时间要求，所以当我们删除一个数时将其与末尾元素置换。

删除元素必须要在o(1)时间内找到 则需要一个map<int,int>,mp[num]=index;

index对应着vector中的下标。

随机数的生成【m,n】

Rand()%(m-n)+m//不是真正的均匀分布

使用c++11新特性

std::default\_random\_engine engine(time(nullptr));　　//设置了种子，每次种子都不一样

std::uniform\_int\_distribution<> dis(1, 10);for (int n = 0; n < 10; n++)

std::cout << dis(engine) << " " << std::endl;

std::mt19937 gen;

std::uniform\_int\_distribution<> dis;

std::random\_device rd;

gen = std::mt19937(rd());

dis = std::uniform\_int\_distribution<>(0, N - v.size());

扫描线算法：

关于扫描线算法用来求一些线段的overlap数量，需要将线段上所有的点进行顺序排序，然后从头到尾遍历一遍，遇到起点就+1，遇到终点就-1。

想法一：

Line(start,end):

Vector<pair<int,int>>lines

Pair<int,int> first:点的坐标，second:0表示起点，1表示终点。

这个方法有个陷阱：遍历的时候要注意把与当前点位置相同的所有点都算完再更新overlap的结果。

想法二：

Map<int,int>mp;//不可以使用unordered\_map，因为要保证我们遍历是按照position的顺序遍历。

对于Line(start,end)，mp[start]++,mp[end]--;

最后直接遍历mp

for(auto t:mp)

Overlap+=t.second;