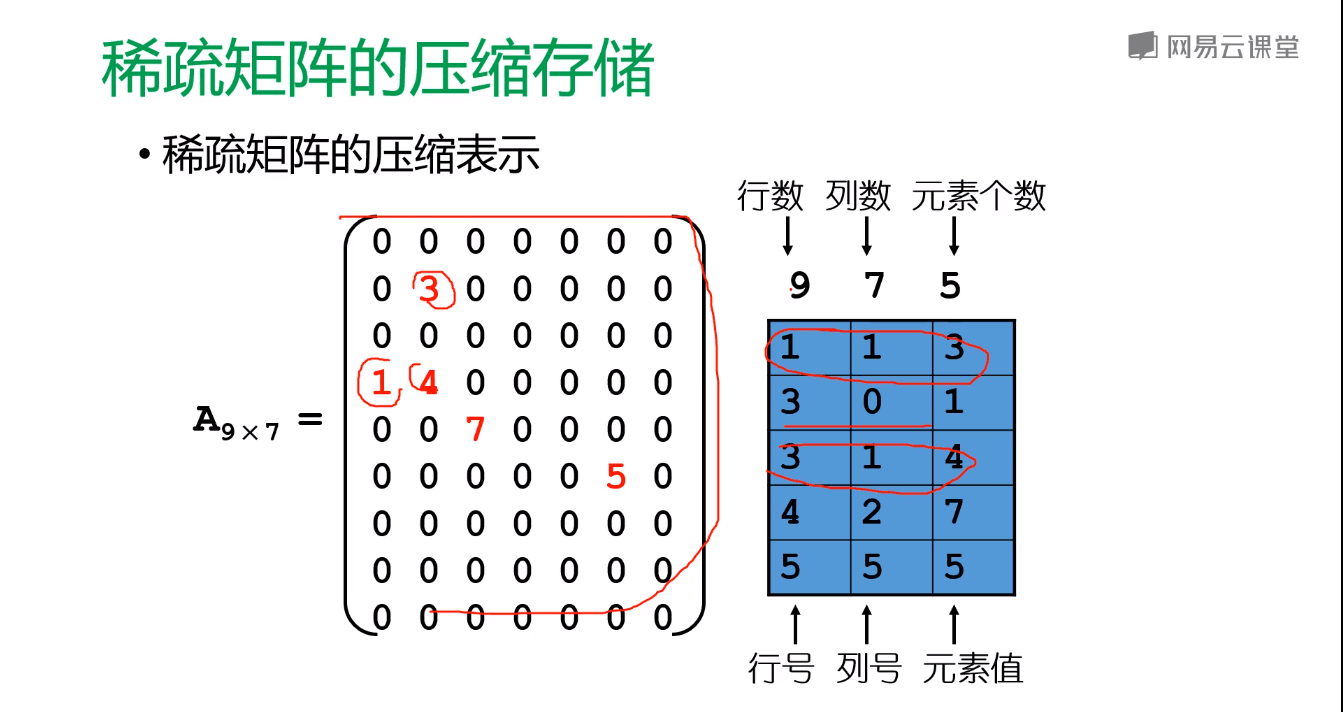
**稀疏矩阵的压缩**

* 1. **稀疏**矩阵与压缩
     1. 用三元组的方式来进行压缩



* + 1. （稀疏考我们自己的喜好来定义稀疏）我们可以想一下，如果用整个数组来放置，就会浪费挺多空间。
    2. 那我们怎么进行实现

1. 我们定义一个结构体
2. 我们用vector来进行接收
3. 我们构造三元组，传进去坐标和非零的个数
4. 代码
   1. 头文件

#pragma once

#include <vector>

#include <iostream>

#include <string>

using namespace std;

template <typename T>

class TSMatrix {

struct Triple {

//两个坐标

int i, j;

T data;

};

public:

vector<Triple> data;

int x = 0;

int y = 0;

int t = 0; //非零个数

//构造函数

TSMatrix(int x, int y, int t = 0) {

if (t > 0) {

data.resize(t);

}

}

//下标取值函数,返回的是三元组的类型的

Triple& operator[](int i)const {

return data[i];

}

Triple& operator[](int i) {

return data[i];

}

int size() {

return data.size();

}

//待会在进行实现

friend void invTSMatrix(const TSMatrix<T>& m, TSMatrix<T>& T);

};

* 1. 主函数

#include <iostream>

#include "TSMatrix.h"

int main()

{

//实例化三行四列，4个非零数组，其实我么也应该知道，我们非零的个数应该与我们的列数是相等的呀

// 既然是三元组，所以我们知道肯定是3列

TSMatrix<int> m(4, 3, 4);

//我们自己输入一些值来进行测试一下

m[0].i = 3; m[0].j = 5; m[0].data = 4;

m[1].i = 4; m[1].j = 6; m[1].data = 5;

m[2].i = 5; m[2].j = 7; m[2].data = 6;

m[3].i = 6; m[3].j = 8; m[3].data = 7;

std::cout << "x" << " " << "y" << " " << "e" << std::endl;

//现在我们进行打印输出一下

for (int i = 0; i < 4; i++) {

std::cout << m[i].i << " " << m[i].j << " " << m[i].data << std::endl;

}

}