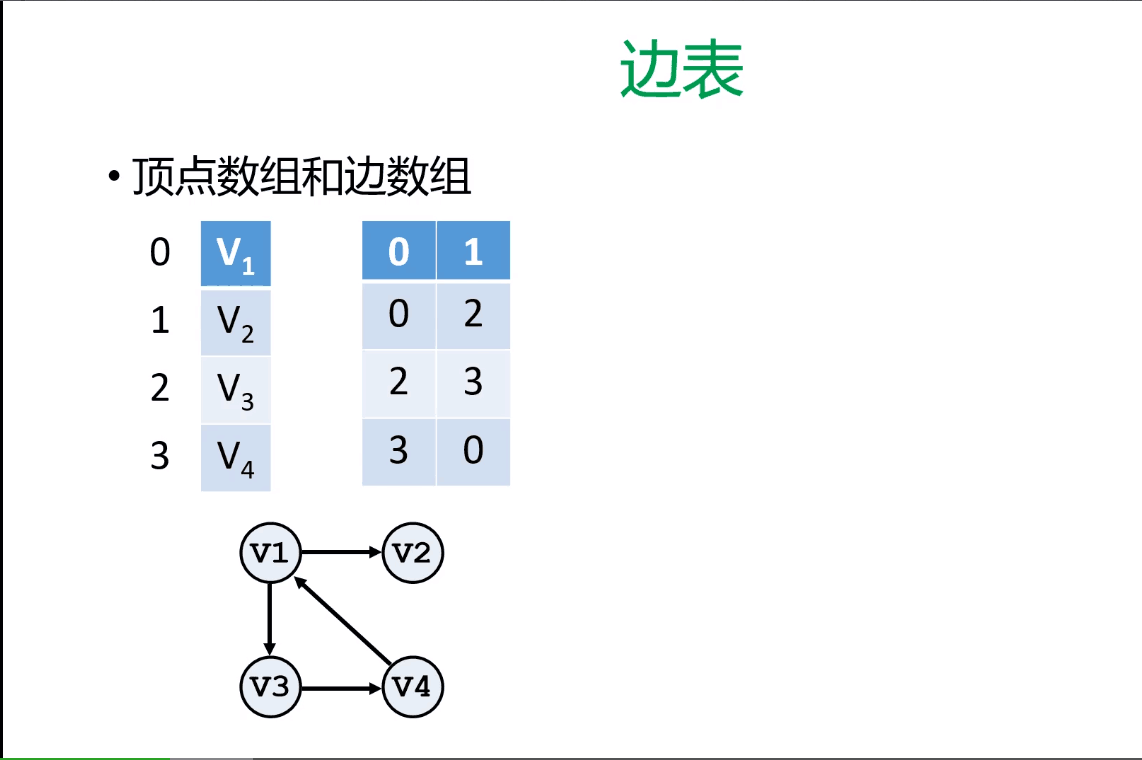
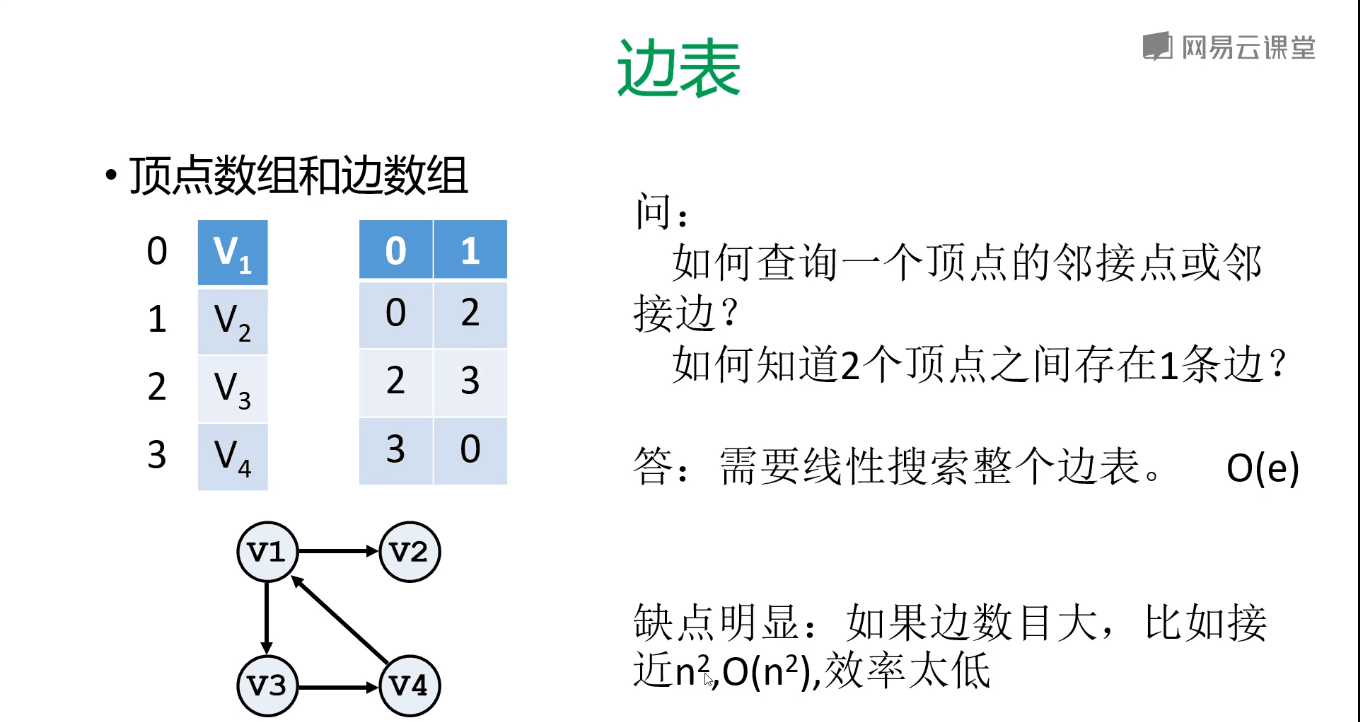
**图的边表表示**



1. 我们可以表示有向图和无向图，无向图无非就是1-2 加上2-1即可
2. 边表的情况



A·如果要求V1到V2是否右边

B·那么我们就需要遍历整个边表，得到0-1是个否有

1. 代码实现

// 无向图的边表表示.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

//

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class eListVertex {

//首先，我们们需要一个二元组来存储边的情况

//再次，我们需要一个数组来存储元素

struct Edge {

int i;

int j;

Edge(int x, int y):i(x),j(y){} //列表初始化

};

bool undirected{ true }; // 无向图

vector<T> vertexs; //存放元素的数组

vector<Edge> edges; //存放边

public:

eListVertex(bool undirected = true):undirected(undirected){} //构造函数

void add\_vertex(T e) {

vertexs.push\_back(e); //把元素push\_back进入数组里面去

}

void add\_edge(int x, int y) {

edges.push\_back(Edge(x, y)); //把边的关系放进数组里面去

}

void print() {

cout << "顶点信息 : " << endl;

for (auto& v : vertexs) {

cout << v <<"-";

}

cout << endl;

cout << "边的信息 : " << endl;

for (auto& e : edges) {

cout << e.i << "-" << e.j << endl;

}

}

};

int main()

{

eListVertex<char> G;

G.add\_vertex('A');

G.add\_vertex('B');

G.add\_vertex('C');

G.add\_vertex('D');

G.add\_edge(0, 1);

G.add\_edge(1, 2);

G.add\_edge(2, 0);

G.print();

}