## 1.比较const graph\_size=10;和# define graph\_size=0;

## 2.对于模板类的书写

我还没有完全掌握：

template <int graph\_size,class Weight>

void Digraph<graph\_size,Weight>::read()//注意在类后面应该这样写Digraph<graph\_size,Weight>

## 3.fatal error C1075:

end of file found before the left brace '{' at 'G:\算法程序\图论\graph\graph.cpp(14)' was matched

出现这种错误的原因使的{没有匹配好

## 4.注意在vs2013中，当对程序进行修改后，

需要选择：“项目”-〉“生成解决方案”或“重新生成项目解决方案”这样才能使修改生效。否则如果只是单单点击“调试”-〉“执行而不调试”是无法使修改生效的

## 5.查如何来定义类的静态成员

## 6．模板类中，模板参数的作用范围

template<int graph\_size,class Weight>

class Digraph

{

}

const Weight infinity = numeric\_limits<int>::max();

如果把infinity的定义放在类定义的外面，则系统是会报错的，因为模板参数Weight的作用范围是class Digraph{}的一对花括号之间

## 7．最短路径问题和最小生成树问题的对比

1.研究对象

最短路径问题是用来解决有向图，它研究的是从源点到图中每个点的最短距离。求解的结果是一个数组，数组中保存的是从源点到该点的距离。源点到自己的距离是0。和从源点到该点的最短路径。

最小生成树是用来解决无向图的。首先需要明白生成树的含义：生成树是树，这棵树包含原图的所有结点。最小生成树就是在所有生成树中总权值最小的生成树。

## 8.二维数组作为参数

Prim(Weight weight\_adjacency[count][count])这种声明方式是否合法

## 9.模板类的继承的书写

template<int graph\_size, class Weight>

class Network public: Digraph<graph\_size, Weight>

## 10.函数作为参数的写法

比如遍历算法：

Traverse（void（\*visite）（Vertex &））

记住要点：

1. 函数的返回类型
2. 函数名
3. 函数的参数列表

下面介绍怎样使用这个函数参数：

void Digraph<graph\_size, Weight>::depth\_first\_traverse(void(\*visite)(Vertex &))const

{

bool visited[graph\_size];

for (Vertex v = 0; v < graph\_size; v++){

visited[v] = false;

}

for (v = 0; v < count; v++){

if (!visited[v])

traverse(v, visisted, visit);

}

}

11.多维数组作为函数参数的写法。