软件使用说明书

目录

《软件使用说明书》以下简称《说明书》,是对”图论算法包”的使用说明.实际输出结果可能与《说明书》中不同，请以实际为准.

一、 适应性简述

“图论算法包”包含 7 个算法:

1. 拓扑排序算法
2. 单源最短路径算法
3. 最小生成树算法
4. 最大流增广路径算法
5. 割点计算算法
6. 强连通分支算法
7. 欧拉回路算法

“图论算法包”能对以上各种算法涉及到的图的类型,边的属性,点的属性有较为完整的支持,能支持无向图、有向图、无权图、有权图以及最大流增广路径算法中的残余图、流图等.

二、 图类型使用条件

图模板类型参数需要输入一个 vector<vetor<int> >类型的二维数组 vt、int 类型的 nodeNum、int 类型的 arcNum,二维数组 vt 中存放邻接表,其中每一个元素记录起点、终点以及边的权值;nodeNum 表示输入顶点数量;arcNum 表示输入边的数量. 注意:由于这种输入所创建的图默认为有向赋权图,若需要计算的算法中涉及到无向图的相关计算,则算法内会自动添加一条反向边.

图类型内包含一个点集和一个边集.其中顶点集最大容量为 101,边集最大容量为 401,即图论算法包最大支持输入 100 个顶点、400 条边的有向图或 100 个顶点、200 条边的无向图.

三、 算法使用条件

1. 拓扑排序算法 - Topological sort algorithm

目的：对于给定标号为 1, 2, ..., N 的 N 个顶点的有向无环图 DAG，计算其顶点的拓扑全序序列。

时间复杂度: $O(|E|+|V|)$, $|E|$为边集大小, $|V|$为顶点集大小.

数据规格：

1. 输入：输入数据由两部分组成:

第一部分包含两个非负整数 N、M，其中 N 为给定图顶点个数，M 为其边数

第二部分有 M 对不超过 N 的正整数 x、y，表示顶点 x 到顶点 y 之间有边
 2. 输出：对于给定的输入数据，输出一行以一个空格分隔的正整数，其恰好是输入数据所确定的有向无环图顶点集的拓扑全序序列。

数据样例：
 1. 输入:

```
17 22
1 4
2 4 2 5 2 6
4 3 4 8 4 9 4 10 4 11 4 12
5 11
6 7 6 10 6 11
9 14
10 13 10 15 10 16
11 12 11 16
15 14
16 17
```

 2. 输出：(注意，输出可能不唯一)

```
1 2 4 5 6 7 8 9 10 11 12 13 15 16 17 3 14
```

2. 单源最短路算法 - Dijkstra algorithm

目的：对于标号为 1, 2, ..., N 的 N 个顶点的边赋权有向图以及源顶点 s，计算由 s 到其余所有顶点的最短路径.

时间复杂度:$O(|V|^2)$, $|V|$为顶点集大小.

数据规格：
 1. 输入：输入数据由三部分组成:
第一部分包含两个非负整数 N、M，其中 N 为给定有向图顶点个数，M 为其边数
第二部分有 M 个三元组 x、y、w，表示顶点 x 到顶点 y 有带实数权 w 的边
第三部分为源顶点标号 s
 2. 输出：对于给定的输入数据，输出 N 行，每行的第一部分为一个顶点的标号 K，紧接一个空格后的第二部分是一个表示由源 s 到顶点 K 的最短路径长的实数，在一个空格后的第三部分是若干正整数，表示由源 s 到顶点 K 的一条最短路径的顶点序列

数据样例：
 1. 输入:

7 12
1 2 2
1 4 1
2 4 3
2 5 10
3 1 4
3 6 5
4 3 2
4 5 2
4 6 8
4 7 4
5 7 6
7 6 1
1

2. 输出：(注意，输出可能不唯一)

1 0
2 2 1 2
3 3 1 4 3
4 1 1 4
5 3 1 4 5
6 6 1 4 7 6
7 5 1 4 7

3. 最小生成树算法 - Prim algorithm

目的：对于给定标号为 1, 2, ..., N 的 N 个顶点的边赋权无向连通图，计算其最小生成树.

时间复杂度: $O(|V|^2)$, $|V|$为顶点集大小.

数据规格：

1. 输入：输入数据由两部分组成:

第一部分包含两个非负整数 N、M，其中 N 为给定无向图顶点个数，M 为无向图的边数

第二部分有 M 个三元组 x、y、w，表示顶点 x 与顶点 y 之间有带实数权 w 的边

2. 输出：对于给定的输入数据，输出 N 行，第一行为一个实数，表示最小生成树边权的和，第二到第 N 行为两个正整数，表示最小生成树的所有边

数据样例：

1. 输入:

7 12
1 2 2
1 4 1
1 3 4
2 4 3
2 5 10
3 4 2
3 6 5
4 5 7
4 6 8
4 7 4
5 7 6
6 7 1

2. 输出：(注意，输出不唯一)

16
1 4
6 7
1 2
3 4
4 7
5 7

4. 最大流增广路径算法 - Maximium flow (Augmented path) algorithm

目的：对于给定标号为 1(源), 2, 3, ... , N-1, N(汇) 的 N 个顶点的网络，计算其最大流.

时间复杂度:$O(|E|^2 \log|V|)$, $|E|$为边集大小, $|V|$为顶点集大小.

数据规格：
1. 输入：输入数据由两部分组成:
第一部分包含两个非负整数 N、M，其中 N 为给定图顶点个数，M 为其边数
第二部分有 M 对不超过 N 的正整数 x、y，表示顶点 x 到顶点 y 之间有边
2. 输出：对于给定的输入数据，在第一行上输出最大流值，并从下一行开始每行输出一对正整数 X，Y 和一个实数 W，表示所确定的最大流在边 (X,Y) 上分配的流量值为 W.

数据样例：
1. 输入:

7 12
1 2 9
1 3 8
1 4 3
2 4 4
3 4 3
2 5 3
3 6 9
4 5 2
4 6 2
4 7 6
5 7 3
6 7 4

2. 输出：(注意，输出可能不唯一)

13
1 2 6
1 3 4
1 4 3
2 4 3
3 4 0
2 5 3
3 6 4
4 5 0
4 6 0
4 7 6
5 7 3
6 7 4

5. 割点计算算法 - Articulation finding algorithm

目的：对于给定标号为 1, 2, 3, ... , N-1, N 的 N 个顶点的无向图，判别其连通性和双连通性，并给出割点集(可能是空集)。

时间复杂度:$O(|E|+|V|)$，$|E|$为边集大小,$|V|$为顶点集大小.

数据规格：
1. 输入：输入数据由两部分组成:
第一部分包含两个非负整数 N、M，其中 N 为给定图顶点个数，M 为其边数
第二部分有 M 对不超过 N 的正整数 x、y，表示顶点 x 到顶点 y 之间有边

2. 输出：对于给定的输入数据，按以下要求输出若干行：

(1) 每行对应一个连通分支

(2) 每行输出格式为：

K: A v[1] v[2] ... v[A],

其中 K 表示第 K 个连通分支，接着的 A 表示这个连通分支的割点个数,后面的 v[i] 表示这个连通分支的割点标号。如果 A 为 0，则之后为空。

数据样例一：

1. 输入:

```
5 7
1 2 1 4 1 5
2 3 2 4
3 4 3 5
```

2. 输出：(注意，输出可能不唯一)

```
1: 0
```

数据样例二：

1. 输入:

```
7 8
1 2 1 4
2 3
3 4 3 7
4 5 4 6
5 6
```

2. 输出：(注意，输出可能不唯一)

```
1: 2 3 4
```

6. 强连通分支算法 - Strong Connected Component algorithm

目的：对于给定标号为 1, 2, 3, ... , N-1, N 的 N 个顶点的有向图，计算其所有强连通性分支.

时间复杂度: $O(|E| + |V|)$, $|E|$为边集大小, $|V|$为顶点集大小.

数据规格：

1. 输入：输入数据由两部分组成:

第一部分包含两个非负整数 N、M，其中 N 为给定图顶点个数，M 为其边数
第二部分有 M 对不超过 N 的正整数 x、y，表示顶点 x 到顶点 y 之间有边
 2. 输出：对于给定的输入数据输出若干行：每行对应一个强连通分支所包含的顶点

数据样例：
 1. 输入:

```
10 15
1 2 1 4
2 3 2 6
3 1 3 4 3 5
4 5
6 3
7 6 7 8
8 6 8 10
9 8
10 9
```

 2. 输出：(注意，输出可能不唯一)

```
1 2 3 6
4
5
7
8 9 10
```

7. 欧拉回路算法 - Euler circuit algorithm

目的：对于给定标号为 1, 2, ... , N 的 N 个顶点的无向图，计算其欧拉回路。

时间复杂度:$O(|E|+|V|)$, $|E|$为边集大小,$|V|$为顶点集大小.

数据规格：
 1. 输入：输入数据由两部分组成:
第一部分包含两个非负整数 N、M，其中 N 为给定图顶点个数，M 为其边数
第二部分有 M 对不超过 N 的正整数 x、y，表示顶点 x 到顶点 y 之间有边
 2. 输出：对于给定的输入数据，输出一行以一个空格分隔的正整数，其恰好是输入数据所确定的无向图的欧拉回路,若不存在欧拉回路,则输出不存在欧拉回路。

数据样例一：

1. 输入:

12 21
1 3
1 4
2 3
2 8
3 4
3 6
3 7
3 9
4 5
4 7
4 10
4 11
5 10
6 9
7 9
7 10
8 9
9 10
9 12
10 11
10 12

2. 输出：(注意，输出可能不唯一)

1 3 2 8 9 3 4 5 10 4 7 3 6 9 7 10 9 12 10 11 4 1

数据样例二：

1. 输入:

4 3
1 2
2 3
2 4

2. 输出：

不存在欧拉回路

四、 实验环境

操作系统: Windows 11
IDE: Visual Studio Community 2019
C++标准: C++ 14