图的深度优先遍历

深度优先算法

例如，第一行两个整数 n m。n 表示顶点个数（顶点编号为 1~n），m 表示边的条数。接下来 m 行表示，每行有 3 个数 u v，表示顶点 u 和顶点 v 直接相连。

输入样式如下：

5 5

1 2

1 3

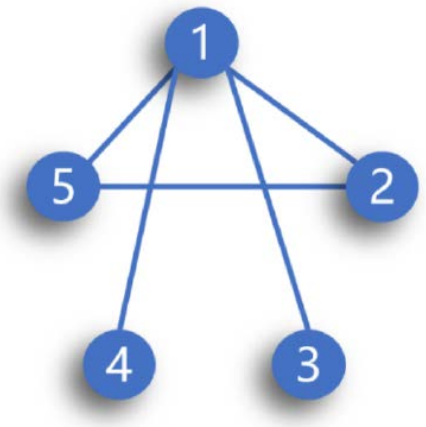
1 4

1 5

2 5

输出样式如下：

1 2 5 3 4



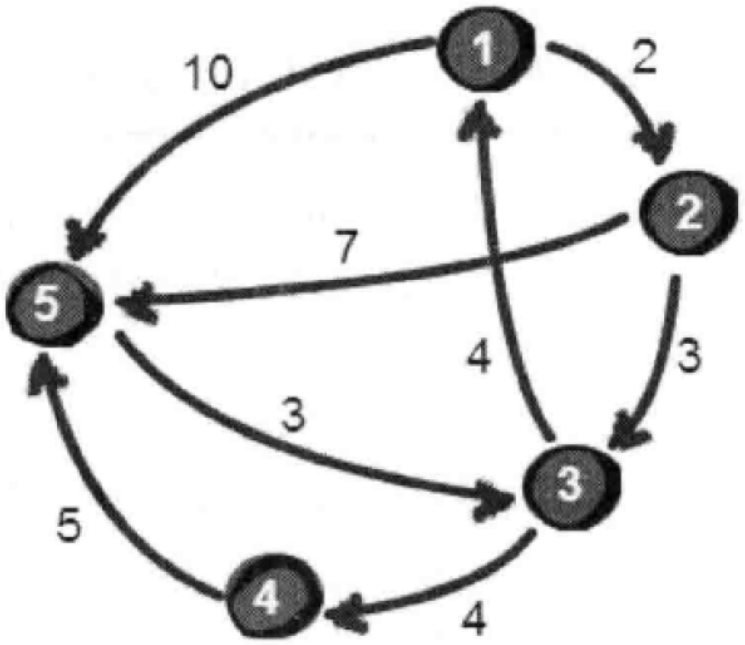
```cpp
#include <iostream>
using namespace std;

int g[101][101], vis[101], n, m, a, b;
void dfs(int i);
int main() {
   cin >> n >> m;
   for (int k = 1; k <= m; k++) {
      cin >> a >> b;
      g[a][b] = 1;
      g[b][a] = 1;
   }

   // 每个顶点为起点，访问图，因为不一定每个顶点都可以遍历整个图
   for (int i = 1; i <= n; i++) {
      if (!vis[i]) {
         dfs(i);
      }
   }
   return 0;
}

void dfs(int i) {
   cout << i << " ";
   vis[i] = 1;
   for (int j = 1; j <= n; j++) {
      if (g[i][j] && !vis[j]) {
         dfs(j);
      }
   }
}
```

最短路径



	1	2	3	4	5
1	0	2	∞	∞	10
2	∞	0	3	∞	7
3	4	∞	0	4	∞
4	∞	∞	∞	0	5
5	∞	∞	3	∞	0

5 个顶点 8 条边

5 8

1 2 2

1 5 10

2 3 3

2 5 7

3 1 4

3 4 4

4 5 5

5 3 3

```
#include <iostream>
using namespace std;

int mindistance = 0x7fffffff, vis[101], n, m, g[101][101], x, y, z;
int istart = 1, iend = 5;
// cur 代表当前所在的城市编号，dis 代表当前已经走过的路程
void dfs(int cur, int dis);
int main() {

   cin >> n >> m;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         if (i == j)
            g[i][j] = 0;
         else
            g[i][j] = 0x7fffffff;
      }
   }

   // 读入城市之间道路
   for (int i = 1; i <= m; i++) {
      cin >> x >> y >> z;
      g[x][y] = z;
   }

   // 从 1 号城市出发
   vis[istart] = 1;   // 标记 1 号城市已经在路径中
   dfs(istart, 0);    // 0 代表当前已走过路程

   cout << mindistance;   // 打印 1-5 城市最短路径
```

```
    return 0;
}

void dfs(int cur, int dis) {

    // 如果当前走过路程已经大于之前找到的最短路，返回
    if (dis > mindistance)
        return;
    // 判断是否到达目标城市
    if (cur == iend) {
        if (dis < mindistance) {
            mindistance = dis; // 更新最小值
        }
        return;
    }

    for (int j = 1; j <= n; j++) {
        // 判断当前城市到城市 j 是否有路
        if (g[cur][j] != 0x7fffffff && !vis[j]) {
            vis[j] = 1;     // 标记城市 j 已经在走过的路径中
            dfs(j, dis + g[cur][j]);  // 从城市 j 继续出发，寻找目标城市
            vis[j] = 0;     // 探索完毕后，取消对城市 j 的标记
        }
    }

    return;
}
```