中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

# 离散数学（2-2）图论上机实验


学生姓名：张世琛

学　　号：1804030401

专业班级：计科 1802

学　　院：计算机科学与技术学院


2019 年 12 月 25 日

# 1 实验内容

1. 实现深度优先搜索方式的递归汉密尔顿图的判定和汉密尔顿回路求解
2. 计算时间复杂度，给出节点数为 5,10,20,100 情况下的运行时间，并模拟出算法效率曲线图，
3. 实现使用部分计算简便的充分条件来实现汉密尔顿图判定的算法。

# 2 背景知识

考虑汉密尔顿的判定和求汉密尔顿回路的问题汉密尔顿问题对应于在一个连通图上求通过所有结点一次且仅一次的回路问题，这个问题是一个 NPC 难题，因此一直没有特别有效的算法。判断一个连通图是不是汉密尔顿图最基本的方法就是使用递归穷举的方法，下面就是一个利用深度搜索来找出图中的汉密尔顿回路的算法。

# 3 算法解释

这里面用到了两个辅助的数组：整型数组 ans[] 用来记录找到的汉密尔顿回路，而布尔型数组 vis[] 用来记录结点的被访问情况，u 表示获取与指定结点相邻的第一个结点，v 表示获取与指定结点相邻的下一个结点，deep 表示获取结点总数。用 vector 储存边

利用这种递归方式，若我们从某一个连通图 G 的第 1 个结点开始搜素，则直接调用 dfs(1,1)

递归穷举的方式对于结点数较少的连通图是比较有效的，但是这种方法毕竟有限，而且速度较慢。

当仅需要判定一个连通图是不是汉密尔顿图的时候，我们也可以使用上面的递归算法，若返回值为 true，则该图是一个汉密尔顿图，不过由于递归算法在结点数比较多的时候效率不高，因此，对于汉密尔顿图的判定问题，可以首先采用一些非常简便的判定方法来确定一个连通路是不是汉密尔顿图，例如：汉密尔顿图的判定并没有充分必要条件，只有几个充分条件和几个必要条件可以使用，我们可以首先利用一些计算非常方便的充分条件来判定一个连通图是汉密尔顿图，比如利用下面提到的一些定理：

1. G 是连通的,

$$\forall u,v \in V \quad u \neq v \quad deg(u) + deg(v) >= n$$

则此连通图一定是汉密尔顿图。

2. 若一连通图的闭包是完全图，则此连通图一定是汉密尔顿图。

尝试将这些定理加入到汉密尔顿图的判定当中，之后这些定理无法判定的时候，才需要采用递归的方式来解决。

# 4 算法核心代码

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=100010;
vector<int>E[maxn];
int n,flag,vis[maxn],m,ff,ans[maxn],deg[maxn];
```

```
void dfs(int u,int deep) {
   if (flag) return;
   if (deep == n + 1 && u == 1) {
      flag = 1;
      return;
   }
   for (auto v:E[u]) {
      if (!vis[v] || (deep == n && v == 1)) {
         vis[v] = 1;
         ans[deep] = u;
         dfs(v, deep + 1);
         vis[v] = 0;
      }
   }
}
int main() {
   //freopen("555.txt", "r", stdin);
   scanf("%d%d", &n, &m);
   for (int i = 1, u, v; i <= m; i++) {
      scanf("%d%d", &u, &v);
      deg[u]++;
      deg[v]++;
      E[u].emplace_back(v);
      E[v].emplace_back(u);
   }
   if (m == n * (n - 1) / 2) {
      printf("Yes\n");
      return 0;
   }
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         if (i != j) {
            if (deg[i] + deg[j] >= n) {

            } else {
               ff = 1;
            }
```

```
43            }
44         }
45      }
46      if (ff == 0) {
47         printf("Yes\n");
48         return 0;
49      }
50      vis[1] = 1;
51      dfs(1, 1);
52      if (flag) {
53         printf("Yes\n");
54         for (int i = 1; i <= n; i++) {
55            printf("%d ", ans[i]);
56         }
57         printf("\n");
58      } else printf("No\n");
59      return 0;
60  }
```