1. 实验

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。

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

//matrixs图的邻接矩阵，v查找起始节点，k当前通路上的结点数量。

Boolean dfs\_hamilton( int [] [] matrixs, int v, int k)

{

int w,n =k;

static Boolean finish;

n++; //n+1 用于下一次递归参数

loop[k]=v;

visit[v]=true;

w=firstAdj(matrixs,v);

while( w!=-1)

{

If(k==getVertex()-1) //已找到一条通路，判定能否构成汉密尔顿回路

{

While( w!=loop[0])

{

//不能构成回路

If(w==-1) break;

//继续查找与v相邻的结点

W=nextAdj(matrixs, v, w)

}

If(w==-1)

{

//不能构成回路，清除相关记录

Visit[loop[k]]=false;

loop[k] = -1;

finish = false;

continue;

}

Else

{

//已经找到汉密尔顿回路，记录并返回

Finish = true;

Loop[super.getVertex()]=loop[0];

Return finish;

}

//节点数量不够，继续扩增通路

If(!visit[w])

{

//找到最近访问的结点

While(loop[k]==-1) k--;

If (matrixs[w][loop[k]]==1)

Dfs(matrixs,w,n);//扩增通路，递归调用

Else

{

//两者不直接连通，清除访问记录直到其父节点

While(loop[k]!=v)

{

Visit[loop[k]]=false;

Loop[k]=-1;

k--;

}

Visit[v]=false;

Loop[k]=-1;

}

}

//获取下一个结点

W=nextAdj(matrixs,v,w);

If(w==-1)

{

//无下一个结点，退回去

Visit[v]=false;

Loop[k]=-1;

}

}

Return finish;

}

利用这种递归方式，若我们从某一个连通图G的第0个结点开始搜素，则直接调用dfs\_hamilton(adj\_matrix,0,0)即可。

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. 若连通图各节点度数满足，并且对任一，满足和，此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。

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。