




数据结构与算法 (Python版)

图的应用：骑士周游问题

陈斌 北京大学 gischen@pku.edu.cn

骑士周游问题

- ❖ 在一个国际象棋棋盘上，一个棋子“马”（骑士），按照“马走日”的规则，从一个格子出发，要走遍所有棋盘格恰好一次。把一个这样的走棋序列称为一次“周游”

35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21		03
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01

骑士周游问题

- ❖ 在 8×8 的国际象棋棋盘上，合格的“周游”数量有 1.305×10^{35} 这么多，走棋过程中失败的周游就更多了
- ❖ 采用图搜索算法，是解决骑士周游问题最容易理解和编程的方案之一
- ❖ 解决方案还是分为两步：
首先将合法走棋次序表示为一个图
采用图搜索算法搜寻一个长度为（行 \times 列-1）的路径，路径上包含每个顶点恰一次

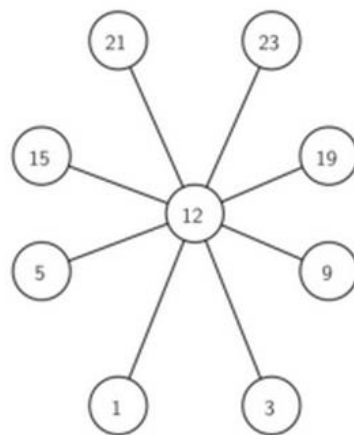
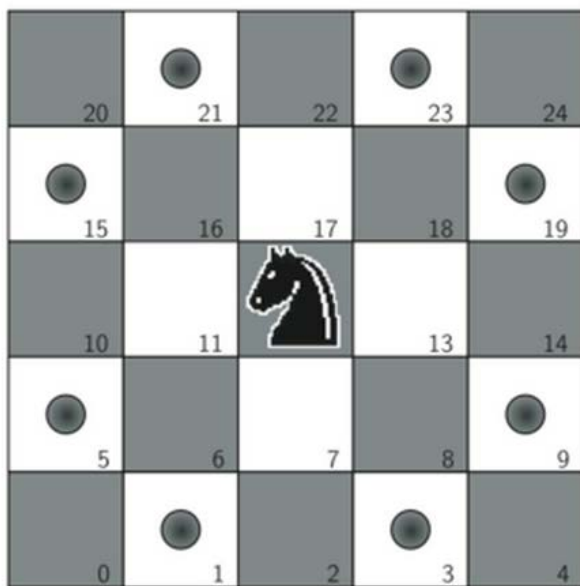
构建骑士周游图

❖ 将棋盘和走棋步骤构建为图的思路

将棋盘格作为顶点

按照“马走日”规则的走棋步骤作为连接边

建立每一个棋盘格的所有合法走棋步骤能够到达的棋盘格关系图



合法走棋位置函数

```
def genLegalMoves(x,y,bdSize):  
    newMoves = []  
    moveOffsets = [(-1,-2),(-1,2),(-2,-1),(-2,1),  
                   ( 1,-2),( 1,2),( 2,-1),( 2,1)]  
    for i in moveOffsets:  
        newX = x + i[0]  
        newY = y + i[1]  
        if legalCoord(newX,bdSize) and legalCoord(newY,bdSize):  
            newMoves.append((newX,newY))  
    return newMoves  
  
def legalCoord(x,bdSize):  
    if x >= 0 and x < bdSize:  
        return True  
    else:  
        return False
```

马走日8个格子

确认不会走出棋盘

构建走棋关系图

```
def knightGraph(bdSize):  
    ktGraph = Graph()  
    for row in range(bdSize):  
        for col in range(bdSize):  
            nodeId = posToNodeId(row,col,bdSize)  
            newPositions = genLegalMoves(row,col,bdSize)  
            for e in newPositions:  
                nid = posToNodeId(e[0],e[1],bdSize)  
                ktGraph.addEdge(nodeId,nid)  
    return ktGraph  
  
def posToNodeId(row,col,bdSize):  
    return row*bdSize+col
```

遍历每个格子

单步合法走棋

添加边及顶点

骑士周游图：8×8棋盘生成的图

❖ 具有336条边，相比起全连接的4096条边，仅8.2%，还是**稀疏图**

