数据结构与算法（Python）-11/第12周

北京大学 陈斌

2021.05.25

线下课堂

- 本周内容小结：图及算法（上）
- 【期末大作业】星际群落

W11：图及算法（上）

图Graph的概念

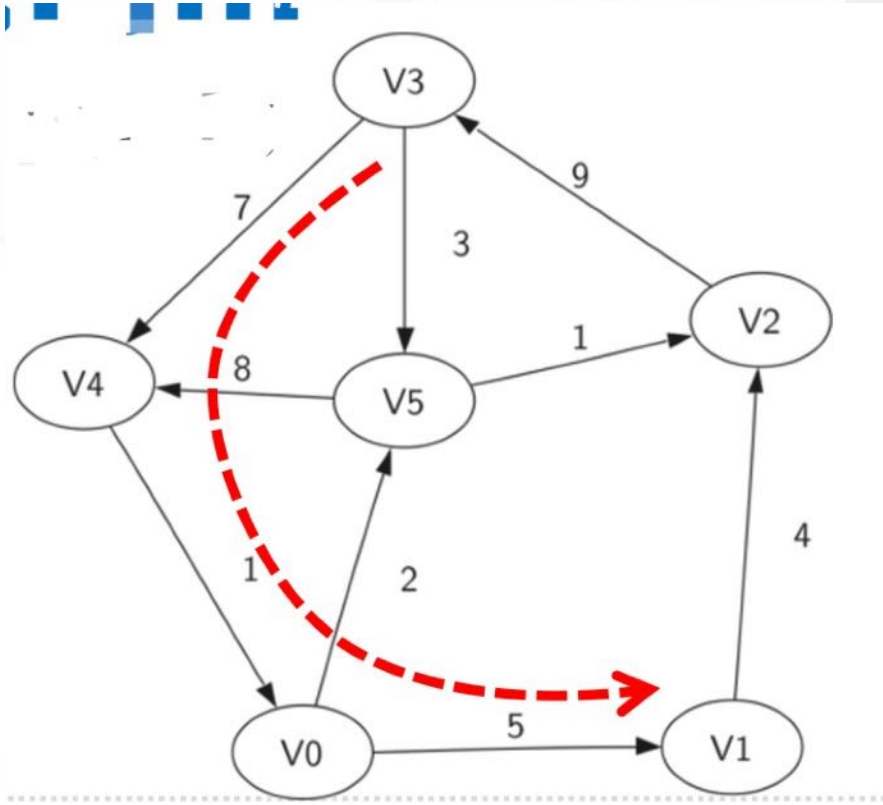
- **图Graph是比树更为一般的结构，也是由节点和边构成**

  实际上树是一种具有特殊性质的图

- **图可以用来表示现实世界中很多事物**

  道路交通系统、航班线路、互联网连接、或者是大学中课程的先修次序

- 节点Node
- 边Edge
- 边权重Weight
- 路径Path
- 带权路径Weighted Path
- 圈Cycle



抽象数据类型：ADT Graph

- 抽象数据类型ADT Graph定义如下：

  Graph()：创建一个空的图；

  addVertex(vert)：将顶点vert加入图中

  addEdge(fromVert, toVert)：添加有向边

  addEdge(fromVert, toVert, weight)：添加带权的有向边

  getVertex(vKey)：查找名称为vKey的顶点

  getVertices()：返回图中所有顶点列表

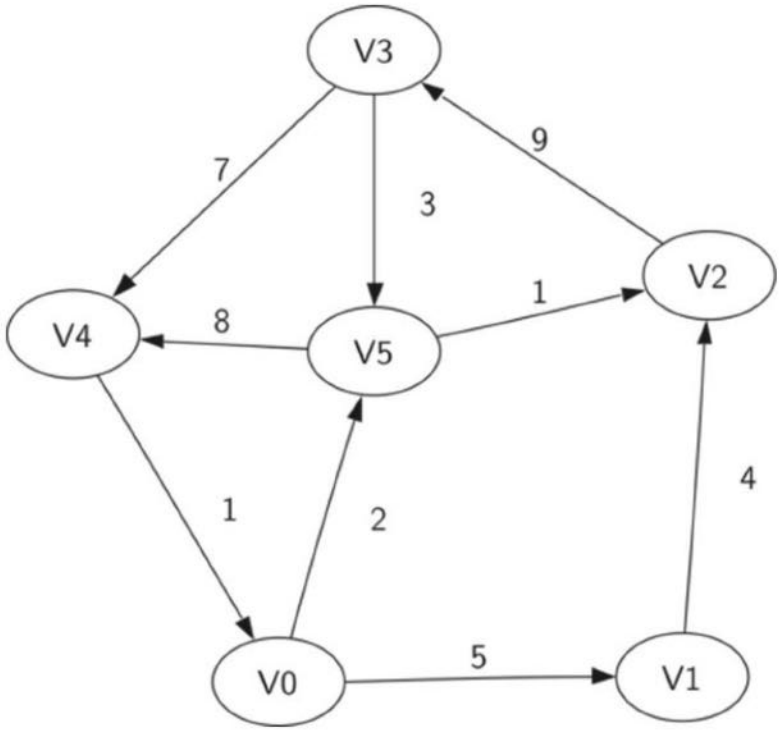
  in：按照vert in graph的语句形式，返回顶点是否存在图中True/False

邻接矩阵Adjacency Matrix

无权边则将矩阵分量标注为1，或者0

带权边则将权重保存为矩阵分量值

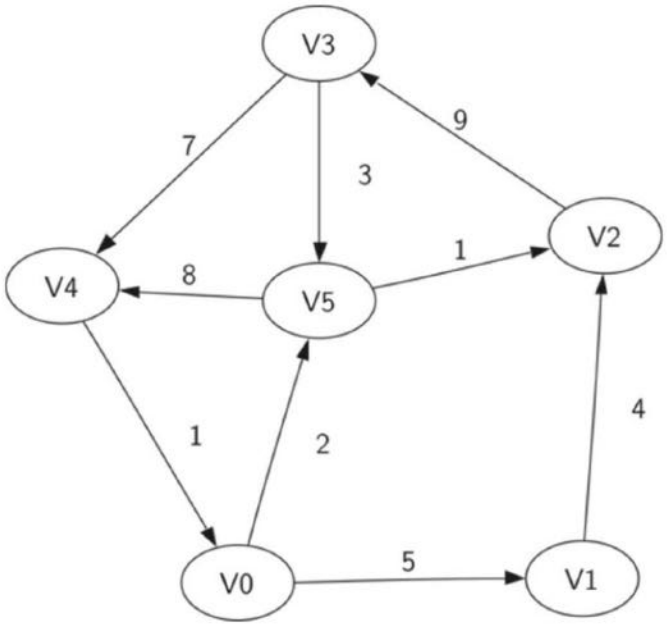
	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

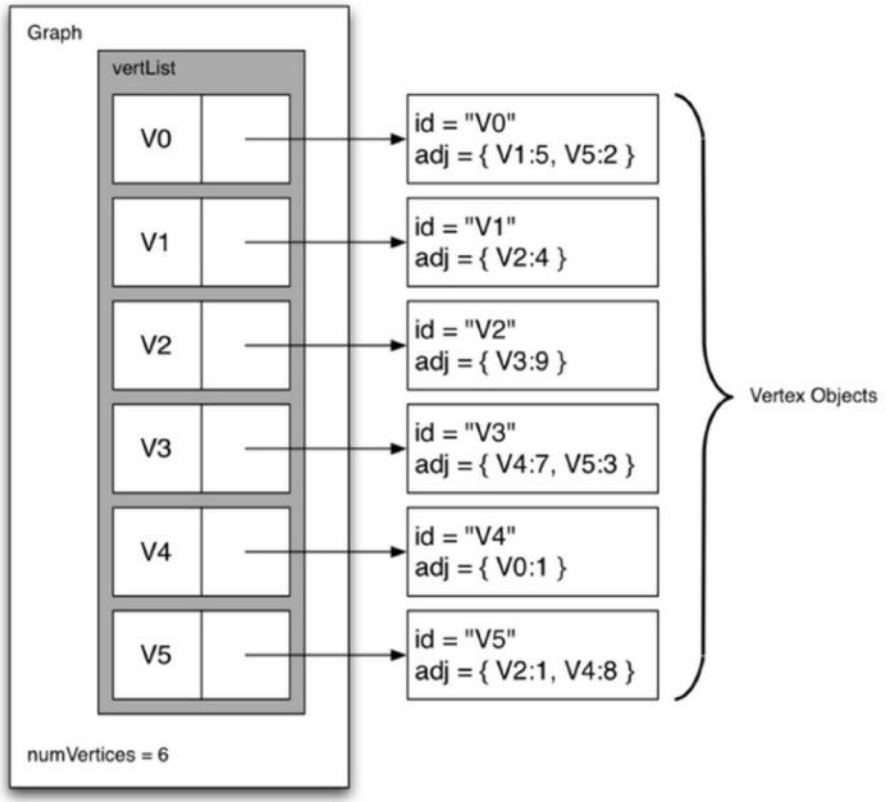


邻接列表Adjacency List

维护一个包含所有顶点的主列表（master list）

主列表中的每个顶点，再关联一个与自身有边连接的所有顶点的列表





词梯问题和BFS

- **问题抽象：单词关系图**

  图节点：单词；

  边：单词差一个字母的关系

- **问题的解**

  找到从A单词到B单词的最短路径

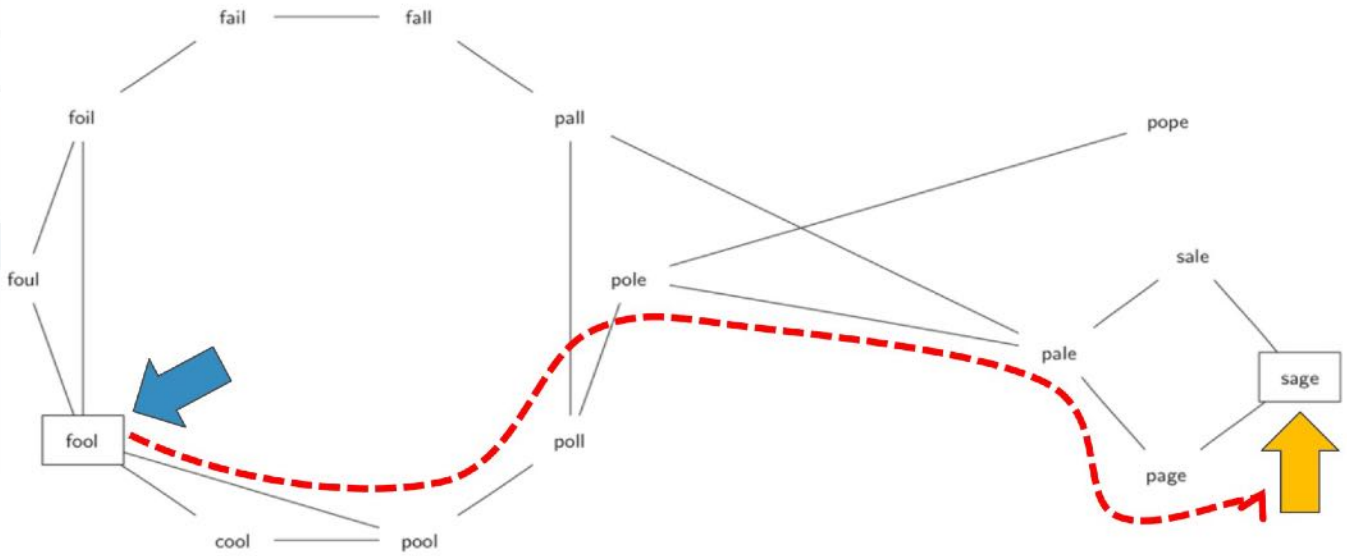
- **几个关键问题**

  建图的优化

  广度优先搜索BFS

❖ **下图是从FOOL到SAGE的词梯解，所用的图是无向图，边没有权重**

FOOL到SAGE的每条路径都是一个解



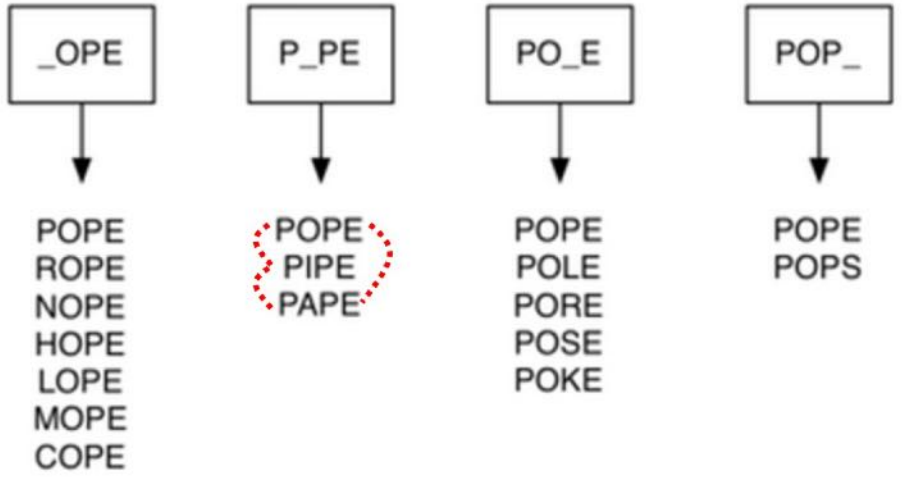
词梯问题：构建单词关系图的优化

- 改进的算法是创建大量的桶，每个桶可以存放若干单词

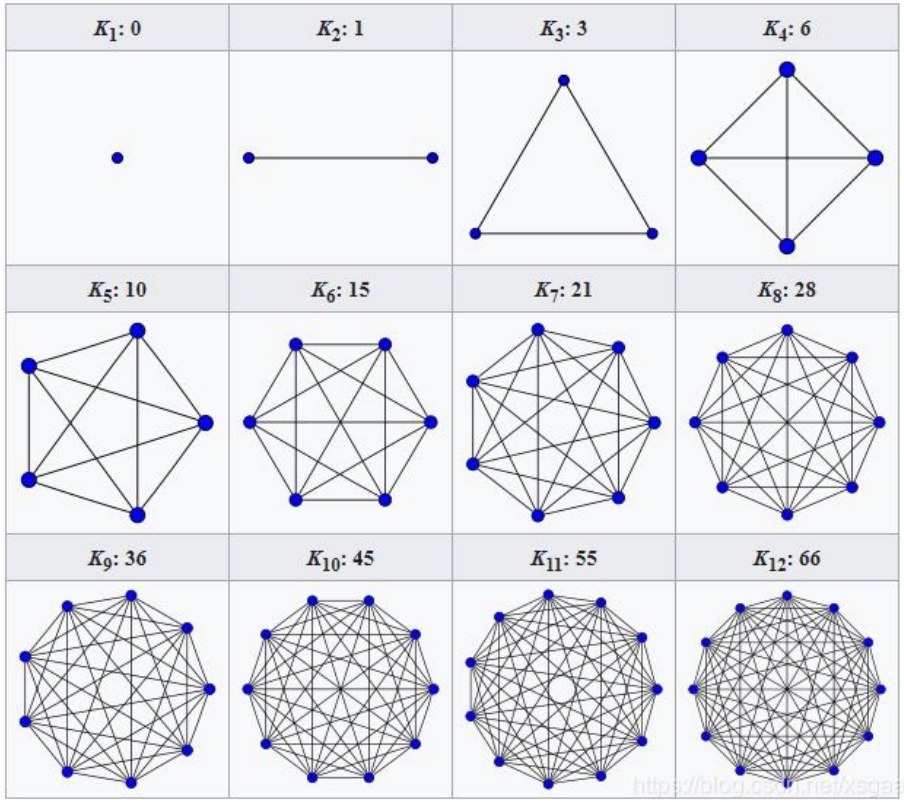
  桶标记是去掉1个字母，通配符“_”占空的单词

- 所有匹配标记的单词都放到这个桶里

  所有单词就位后，再在同一个桶的单词之间建立边即可



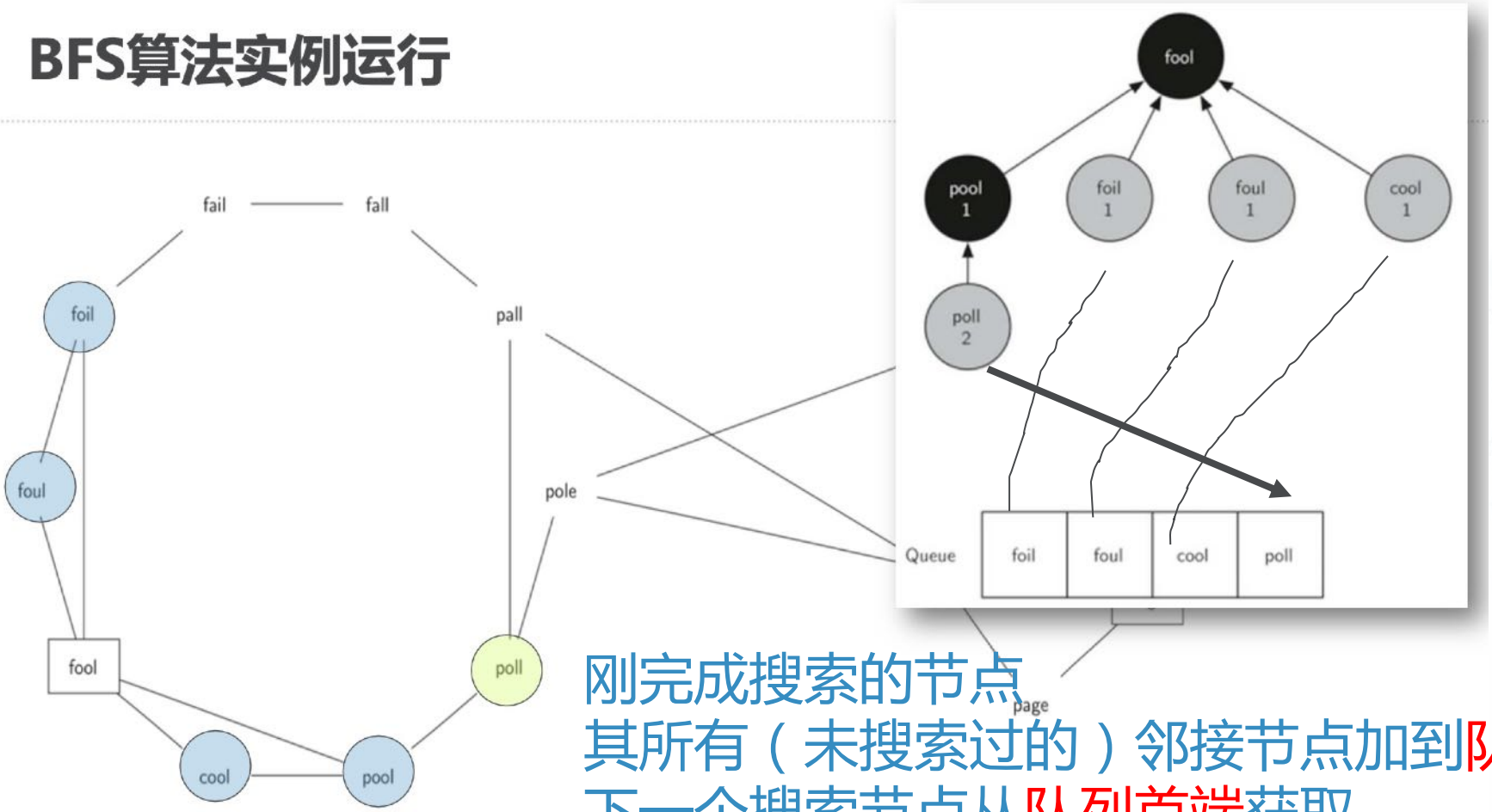
每个桶内部是完全图



词梯问题：广度优先搜索BFS

- 用到了队列Queue数据结构

BFS算法实例运行



刚完成搜索的节点
其所有（未搜索过的）邻接节点加到队列尾端
下一个搜索节点从队列首端获取

骑士周游问题和DFS

- **问题抽象：棋盘格关系图**

  图节点：棋盘格

  边：棋盘格能合法走到的关系

- **问题的解**

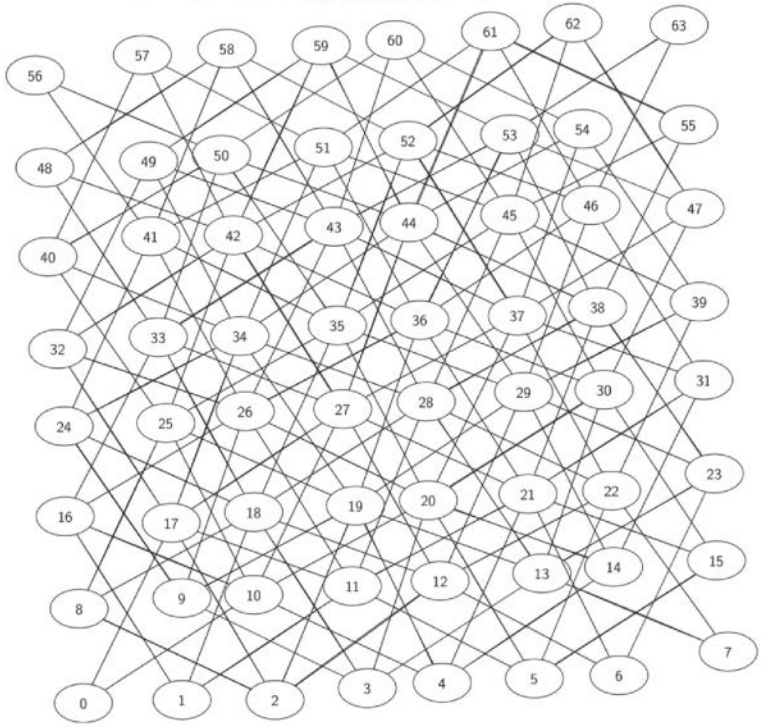
  找到棋盘格A不重复走过所有格子的某条路线

- **几个关键问题**

  深度优先搜索DFS

  启发式规则优化搜索性能

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



骑士周游问题：深度优先搜索DFS

- **用到了栈Stack数据结构**
  深入层次的“回溯”

❖ **深度优先搜索解决骑士周游的关键思路**

如果沿着单支深入搜索到无法继续（所有合法移动都已经被走过了）时

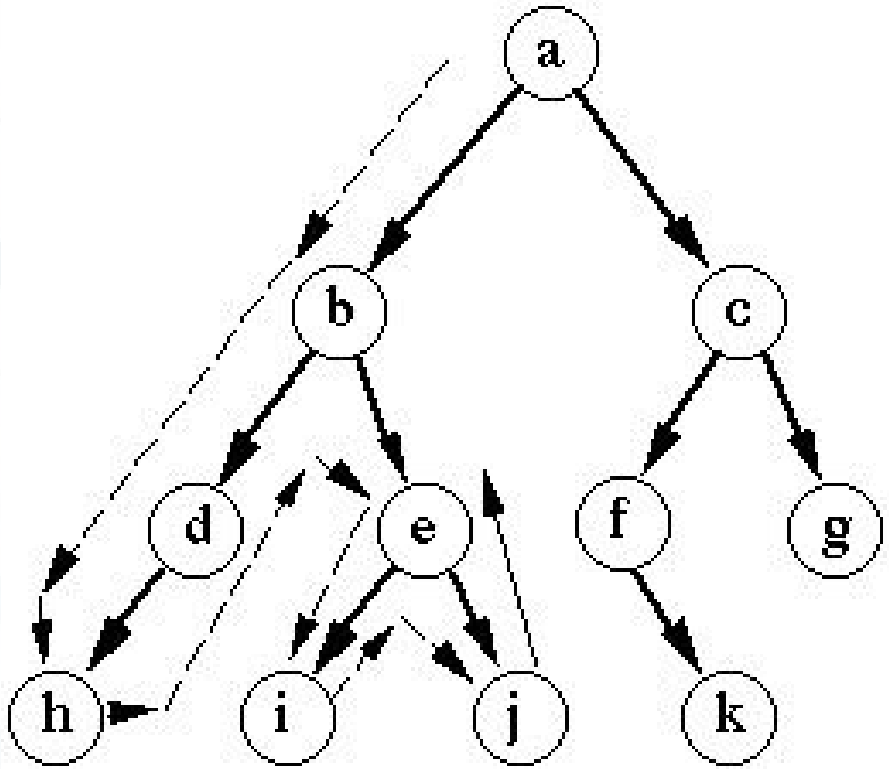
路径长度还没有达到预定值（8×8棋盘为63）

那么就清除颜色标记，返回到上一层

换一个分支继续深入搜索

❖ **引入一个栈来记录路径**

并实施返回上一层的回溯操作



Depth-first search

骑士周游问题：启发式规则

DFS搜索的次序

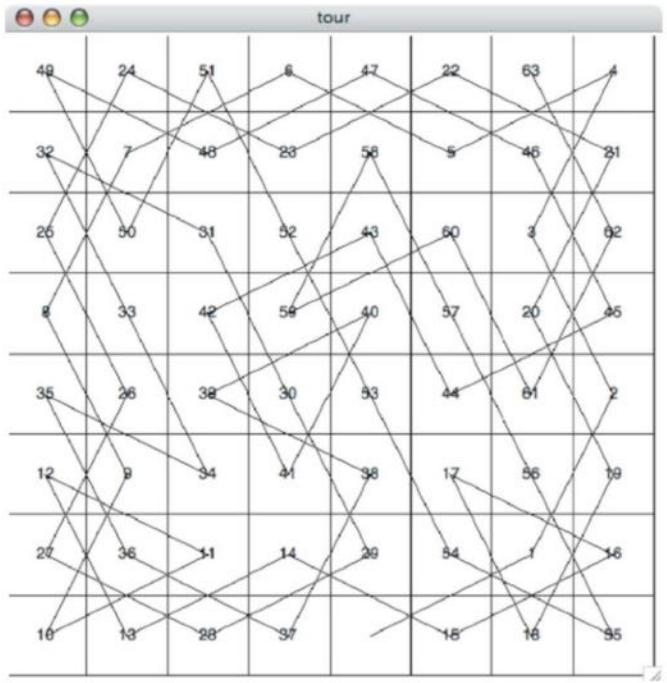
对算法复杂度没有影响

对实际搜索的性能至关重要

❖新的算法，仅修改了遍历下一格的次序

将u的合法移动目标棋盘格排序为：具有最少合法移动目标的格子优先搜索

```
def orderByAvail(n):
    resList = []
    for v in n.getConnections():
        if v.getColor() == 'white':
            c = 0
            for w in v.getConnections():
                if w.getColor() == 'white':
                    c = c + 1
            resList.append((c,v))
    resList.sort(key=lambda x: x[0])
    return [y[1] for y in resList]
```



启发式规则对智能算法的贡献

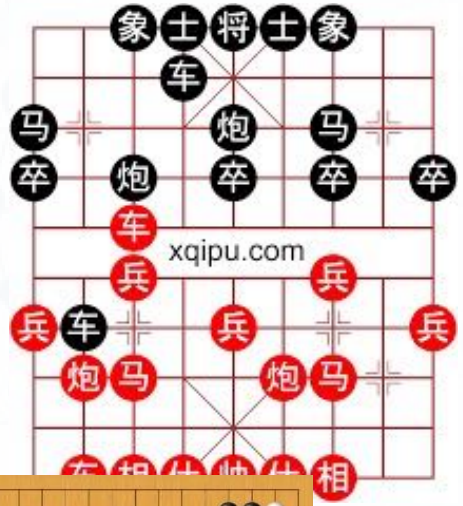
- **采用先验的知识来改进算法性能的做法，称作为“启发式规则heuristic”**

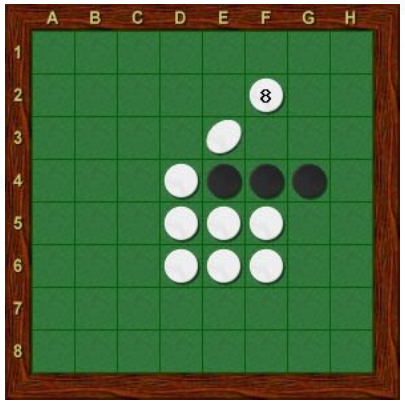
启发式规则经常用于人工智能领域；

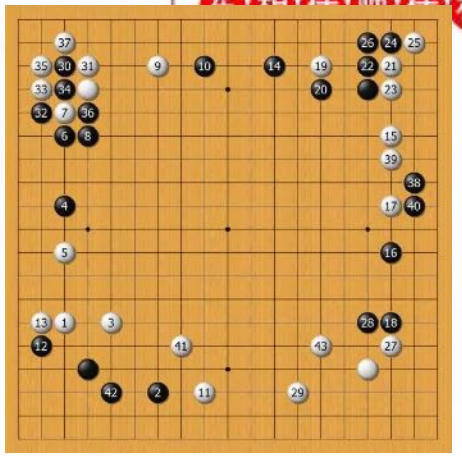
可以有效地减小搜索范围、更快达到目标等等；

如棋类程序算法，会预先存入棋谱、布阵口诀、高手习惯等“启发式规则”，能够在最短时间内从海量的棋局落子点搜索树中定位最佳落子。

例如：黑白棋中的“金角银边”口诀，指导程序优先占边角位置等等







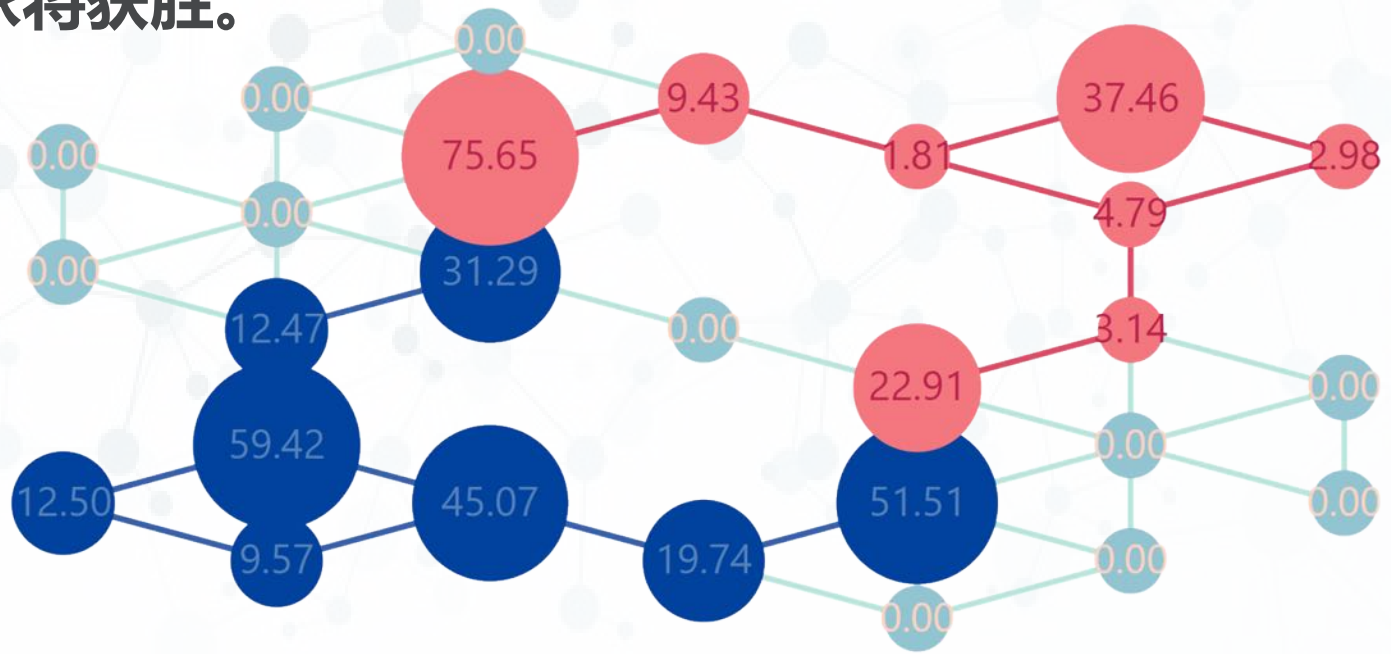
【期末大作业】星际群落

- 星际群落简介
- 星际群落规则
- 作业任务描述
- 作业时间进度
- 算法开发指南

Stellaris: Federations

星际群落简介

- 本大作业主题围绕星际群落的生命演化游戏展开。
- 游戏中，双方玩家将操纵己方单位向不同节点移动，单位在移动后将与所在位置的敌方单位（如果存在的话）发起战斗并争夺领地和资源。
- 率先占领对方大本营的玩家将获胜。



星际群落规则

- **具有时滞的回合制AI对抗游戏**
  每个回合分为三个阶段：运输、战斗、生产
- **游戏场景为对称的双向有向图，由节点和节点之间相连的通道构成。**
- **节点可分为两个大本营节点和普通节点两种，各节点每回合都能生产兵力。**
- **获胜条件为，在回合上限内率先占领对方大本营节点**
  若双方在同回合攻破对方大本营，则该回合结束之后拥有兵力总数量更多者获胜
  若达回合上限后无玩家攻破对方大本营，则该回合结束之后拥有兵力总数量更多者获胜

星际群落规则

每回合第一阶段为运输阶段。

玩家可以对每个已控制的节点发出命令，向不同节点送出不同数量的兵力。

同一个节点可以同时向若干个邻接节点分摊运输兵力。

运输有兵力损耗，损耗量为运输量的正平方根；损耗完的战力如下：

$$gain = \max(send - \sqrt{send}, 0)$$

平台获取到双方的运输指令之后，会同时执行完所有的运输指令，然后进入后续阶段。

星际群落规则

每回合第二阶段为战斗阶段

由对战平台根据第一阶段的运输结果自动执行。

若某节点上同时出现双方兵力，会触发战斗机制，兵力较少者被全歼，兵力较多者取胜，其残余兵力占领此节点。运算规则如下：假定双方在此节点的兵力分别为a和b，战斗后双方剩余兵力为a'和b'，其中：

```
.    if a>b:
.        a', b' = (a*a-b*b)**0.5, 0
.    else:
.        a', b' = 0, (b*b-a*a)**0.5
```

星际群落规则

每回合第三阶段为生产阶段

由对战平台根据第二阶段的战斗结果自动执行。

若当前兵力小于阈值，则兵力增加；多余阈值时则减少。生产阶段后的兵力计算如下：

```
.    if x < powerLimit:
.        New = x + (1 - x/powerLimit) * x * spawnRate
.    else:
.        New = powerLimit + despawnRate * (x - powerLimit)
```

无主节点内永远不会生产兵力。

星际群落规则

游戏数据结构及玩家须知属性

规定玩家1的大本营节点编号为1号，玩家2的大本营节点编号为N号，其中N=len(nodes)-1为总节点个数。nodes[i]存储的是编号为i的节点，nodes[0]无定义。

节点Node类有以下信息：

节点的编号。self.number=number

节点归属势力。self.belong=belong

节点的兵力数量。self.power=(power1,power2

邻接节点的信息。self.nextinfo=[node_id1,node_id2,...]

星际群落规则

玩家需要编写并提交一个python文件

其中应包含player_class类，至少要实现：

- 成员方法player_func(self, map_info)，与
- 初始化方法__init__(self, player_id)。

map_info为提供的地图GameMap类实例

- 存储当前回合开始前地图的状态，以及玩家编号player_id

player_func方法需要返回一个列表，包含若干元组。

- 每一个元组代表一条运输指令
- 元组构成为(from_node, to_node, num)
- 代表该玩家在运输阶段的决策。

星际群落规则

- **异常处理：当任何一方的代码出现异常行为时，就会触发异常处理机制，忽略该玩家本回合的所有运输行为。**
- **常见的异常行为包括但不限于：**

  玩家返回的列表中，从一个节点输送出去的兵力总和超过该节点存储的兵力(透支)；

  玩家返回的列表中，存在并不直接相连的两个节点之间的运输操作；

  玩家返回的列表中，出现非法或不合理的数据类型；

  对于单回合，某玩家的算法在时间限制config.MAX_TIME范围内没有返回结果。

星际群落规则

- **测试与调试**
- **技术组提供若干个公开且基础的AI文件，供各小组强化自己的算法。**
- **各小组可以按文档叙述，运行本地调试工具。**

  成功运行之后，会生成复盘数据存储于output.json文件中。

- **复盘数据可视化**

  方法一：运行server.py进行复盘数据可视化。

  方法二：访问https://mi.js.org/dsa21vis/battleground.html。将output.json文件拖入网页界面，就可以按网页指示进行复盘数据可视化。

星际群落规则

模拟赛

所有同学都可以注册个人用户，上传自己的文件至代码竞技场自由对战。

代码竞技场的网页链接：http://gis4g.pku.edu.cn/ai_arena/game/5/。

模拟赛从即日起开始到6月11日结束。模拟赛天梯分不计入大作业成绩。

正式赛

各小组以小组账号登录，上传一份正式参赛代码文件

暂定于6月10日18:00发放小组账号，上传时间期限为6月11日18:00

分为天梯赛和淘汰赛两个环节

- 6月12日-6月14日为天梯赛，代码竞技场选取天梯前八名出线，进入淘汰赛决赛。
- 6月15日在课堂进行淘汰赛决赛，决出冠亚季军，发放奖品、纪念品等。

算法开发指南

详见github代码仓库

github：https://github.com/pkulab409/pkudsa.stellar

国内码云镜像：https://gitee.com/chbpku/pkudsa.stellar

码云镜像国内高速访问，适合没有梯子的同学

各组必看文档

规则文档，开发文档

代码竞技场

模拟赛期间可供衡量自己代码的算法强度；正式赛期间作为作业评分的依据

http://gis4g.pku.edu.cn/ai_arena/

作业任务描述

- **本次大作业需要组队完成，原则上每组人数为3或4人。**

  各组以自愿组队为主；未完成自愿组队的同学将由老师和助教随机分组

  各组需指明组长一名，职责包含召集作业过程讨论、汇总代码和报告、代表小组参加竞赛

- **每个小组需要根据游戏规则，分析问题，设计算法，调试参数，撰写大作业实习报告。**

- **各组的大作业成绩由编程、报告、竞赛成绩等项目构成。**

  评分适用于全组同学

  每组有额外3分加分，由组长组织本组民主评议，奖励1～2名表现突出的组员（含组长）。

  另外，组长有权对实习过程中表现差的同学提出批评及降分建议。

作业任务描述

- **编程：依托星际群落基础设施代码，用Python编写对战算法**

  算法应能根据当前局势和回合阶段，返回本方的响应

  要求应用本课所学到的数据结构与算法组合，并具有一定的复杂度和智能。

  要求代码结构清晰、格式规范、注释合理。

- **报告：撰写算法实现过程的实验报告**

  包括算法思想阐述、程序代码说明、测试过程报告、小组分工和实验过程总结等部分

  要求实验报告图文并茂、内容丰富、结构清晰、写作规范、逻辑性强。

- **竞赛：参加 pkudsa.stellar 星际群落算法竞赛**

  与其他小组的算法对战，根据输赢获得竞赛排名

  要求对战过程无明显bug

作业时间进度

- 即日开始实习作业各环节
- 包含组队、开发算法，编程测试，挑战天梯，撰写报告
- 6月15日（周二）课上进行算法竞赛的淘汰赛

下课！