独辟蹊径

队名：寻幽（迪杰斯特拉派）
成员：季冲（320582199301296018）
王颖（321084199302034825）
学校：南京理工大学

内容提要

算法原理： 图深度优先搜索 + 剪枝 + 启发式搜索

算法实现： C++

程序说明： 输入输出格式说明、数据存储

结果分析： 根据实验结果分析效率及准确度

总结： 对比及可优化部分

算法原理

- 图的深度优先搜索能够通盘地读取图信息：主要需要决定从哪里开始读，依照什么顺序读，到哪里为止。
- 这里，选择源点作为搜索的开始点，按照深度优先的顺序，到终点则得到一条路径，而后评估当前路径是否符合题目的各项要求，存储结果。
- 而后回溯递归，当目前的不完整路径的评价指标已经不如存储的完整路径则剪枝，继续回溯返回。
- 细节：当访问过一个节点后并不像传统DFS标记visited防止重复访问。

算法原理

- 剪枝策略
  - 搜索过程中维护一个当前最优解
  - 每次进行下一步搜索前比较一下

    若Cost of当前已走路径 > Cost of 当前最优解，直接回溯
- 启发式搜索

  下一跳节点优先级排序策略：（从高到低）

  1. End节点
  2. 构成绿边
  3. 绿色节点
  4. Cost较小

程序实现

- 输入输出
- 图存储　　采用邻接矩阵
- 约束条件存储　　采用数组 vector<int> constraints;

程序实现

- 输入输出
  - 输入：
    - 节点数目 N (所有节点编号从0~N-1)
    - 图的邻接矩阵 board (board[i][j]=cost of $edge(i, j)$, board[i][j]=-1 means blocked)
    - 起点和终点 (start, end)
    - 红边(禁止通行, board[i][j]=-1)
    - 绿边(必经边)
    - 绿点(必经点)
    - 最大跳数 $maxHop$ (最多经过的储物间个数)
  - 输出：
    - 最优路径，若无解则输出No Paths!

程序实现

- 约束条件存储

采用数组表示 vector<int> constraints;

用-1表示非约束，0表示是约束，大于0表示约束已满足

利用Cantor pairing function[1]: $\pi : \mathbb{N}^2 \rightarrow \mathbb{N}$

$$\pi(k_1, k_1) = \frac{1}{2}(k_1 + k_2) \times (k_1 + k_2 + 1) + k2$$

从而将$edge(i, j)$编码成一个唯一的非负整数

如$edge(2, 4)$为一条绿边，则constraints[π(2, 4)+N] = 0;

如$node(7)$ 为一个绿点，则constraints[7] = 0;

[1] Reference http://en.wikipedia.org/wiki/Pairing_function

总结

- 利用深度优先搜索＋剪枝＋启发式搜索的策略可以较快地找到满足的要求的解，但是由于启发式搜索策略带来了额外的开销，相比不用启发式策略，得出最优解的效率降低了。
- 程序实现中利用pair function使用数组方式存储约束条件的方法，相比使用其他数据结构去存储，大大提高了效率。
- 进一步工作展望
  - 考虑动态规划的实现方式，用F(*st, end, cons*)表示约束集为cons下从节点*st->end*的最优解

$$F(st,end,cons) = min\{F(i,end,cons_i), i \in neighbor(st)\}$$

总结

- 分工情况
  - 建模：季冲，王颖
  - 程序：季冲
  - 报告：季冲王颖
- 队名：寻幽
- 成员：季冲 (32058219301296018)
- 王颖 (32108419930203 4825)
- 学校：南京理工大学