4.栈与队列

试探回溯法：迷宫寻径

No matter where they take us,

We'll find our own way back.

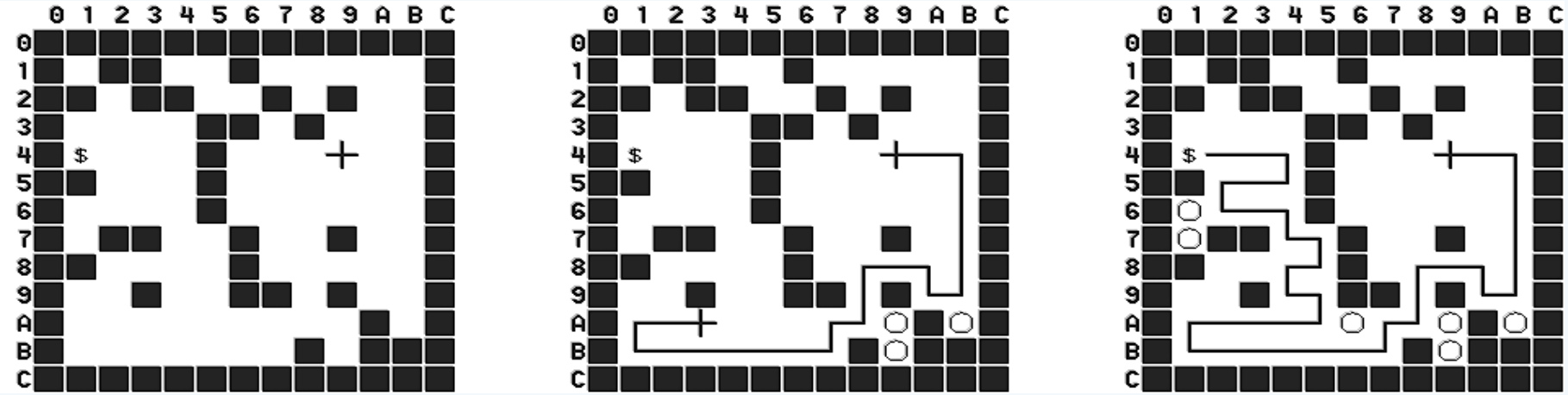
邓俊辉

deng@tsinghua.edu.cn

迷宫寻径

- 路径规划（path planning），系人工智能等领域的基本问题
- n×n方格组成的迷宫，四周方格构成围墙，中间若干方格为障碍物

  机器人漫游其间，每步只能运动到东、南、西、北方向的某一邻格
- 指定的起点s和终点t，在其间找出一条四连通的通路（如果存在）



算法

```
❖ bool labyrinth( Cell Laby[MAX][MAX], Cell* s, Cell* t ) {
    Stack<Cell*> path; //用栈记录通路（Theseus的线绳）
    s->incoming = UNKNOWN; s->status = ROUTE; path.push(s); //从起点出发
    do { //不断试探、回溯，直到抵达终点，或者穷尽所有可能
       Cell* c = path.top(); if (c == t) return true; //找到通路；否则...
       while ( NO_WAY > ( c->outgoing = nextESWN(c->outgoing) ) ) //查找另一
          if ( AVAILABLE == neighbor(c)->status ) break; //尚未试探的方向
       if ( NO_WAY <= c->outgoing ) //若所有方向都已尝试过，则向后回溯一步
          { c->status = BACKTRACKED; c = path.pop(); } //（Theseus的粉笔）
       else //否则，向前试探一步
          { path.push( c = advance(c) ); c->outgoing = UNKNOWN; c->status = ROUTE; }
    } while ( !path.empty() );
    return false;
 }
```

数据结构

❖ struct Cell { //迷宫单元

```
    int x, y; //坐标
    Status status; //类型
    ESWN incoming, outgoing; //进入、走出方向
};
```

❖ 相关类型

```
//状态：可用、在当前路径上、所有方向均尝试失败后回溯过、不可使用（墙）
typedef enum { AVAILABLE, ROUTE, BACKTRACKED, WALL } Status;
//单元的相对邻接方向：未定、东、南、西、北、无路可通
typedef enum { UNKNOWN, EAST, SOUTH, WEST, NORTH, NO_WAY } ESWN;
//依次转至下一邻接方向
inline ESWN nextESWN( ESWN eswn ) { return ESWN (eswn + 1); }
```

进一步的考虑

- 如何降低最坏情况的概率？

  采用随机策略，等概率试探各方向

- 如何支持八连通运动规则？

  改写neighbor()，扩充四个方向

- 如何找出更短的通路？

  环路：尽可能发现并消去

  弯路：尽可能发现并消去

  贪心：终点方向优先试探

  ...

- 通用算法