产生式系统的搜索

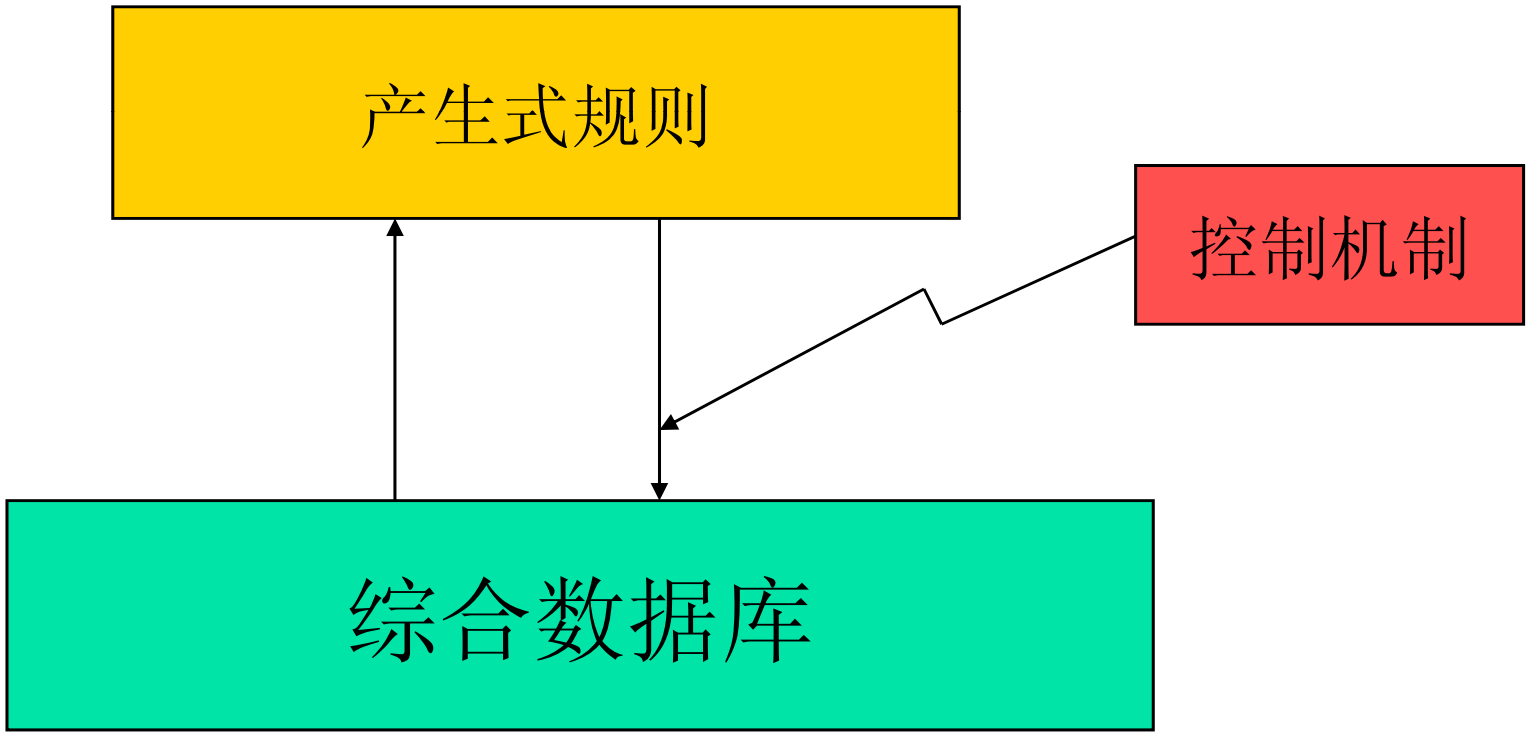
张文生

中国科学院自动化研究所

内容

- 回溯策略
- 图搜索
- 无信息搜索
- 启发式搜索(A*)
- A*算法的可采纳性

回顾



- 控制机制
  - 控制策略
    - 激励---点燃
  - 两类
    - 不可撤回的控制策略：
    - 试探性控制策略
      - 回溯型
      - 图搜索
  - 具体手段
    - 冲突删除策略

状态

- 任一时刻，综合数据库的情况；

2	3	7
	5	1
4	8	6

{A，B，C，D}

（c, a, b, 0, 0）

状态空间

- 状态空间
  - 所有可能的状态的全体.

2	3	7
	5	1
4	8	6

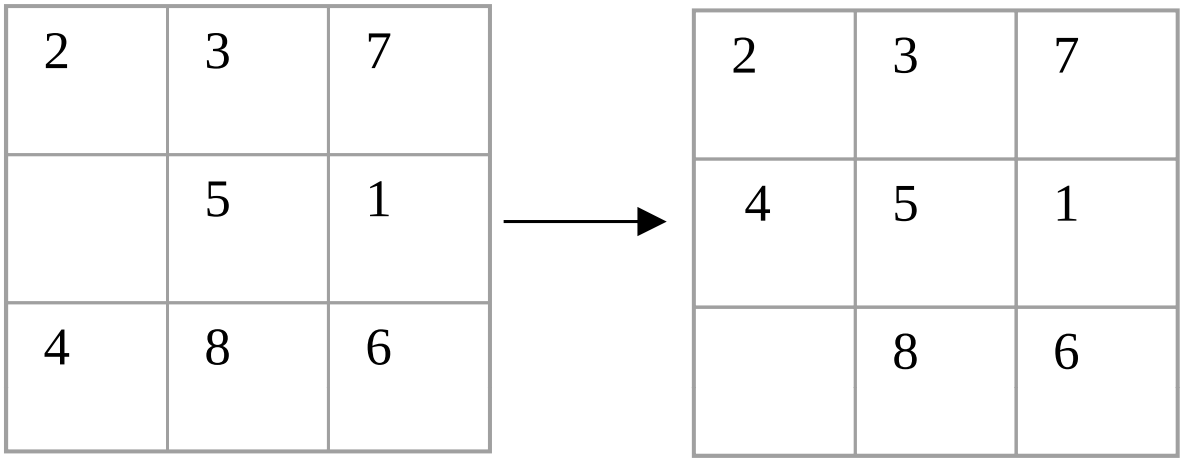
5	8	6
	1	2
7	4	3

……

1	2	4
	6	5
7	8	3

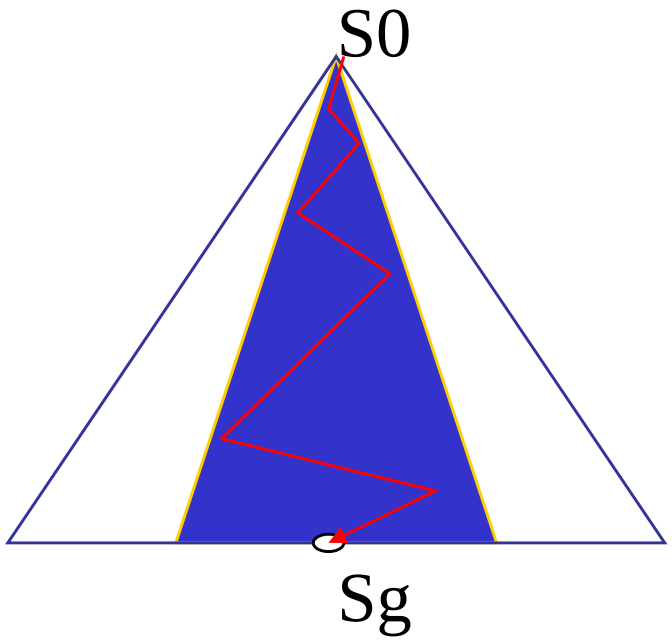
状态转移

- 初始状态
- 目标状态
- 状态转移
  - 规则



搜索(search)

- 路径
  - 状态序列
- 搜索
  - 寻找从初始状态到目标状态的路径;



搜索的必要性

- **AI为什么要研究search?**
  - 问题没有直接的解法;
    - 解方程组;
    - 定理证明;
  - 需要探索地求解;

搜索与检索的区别

- 状态是否动态生成;
  - 检索: 静态;
    - 在数据库中检索某人的纪录;
  - 搜索: 动态生成;
    - 下棋

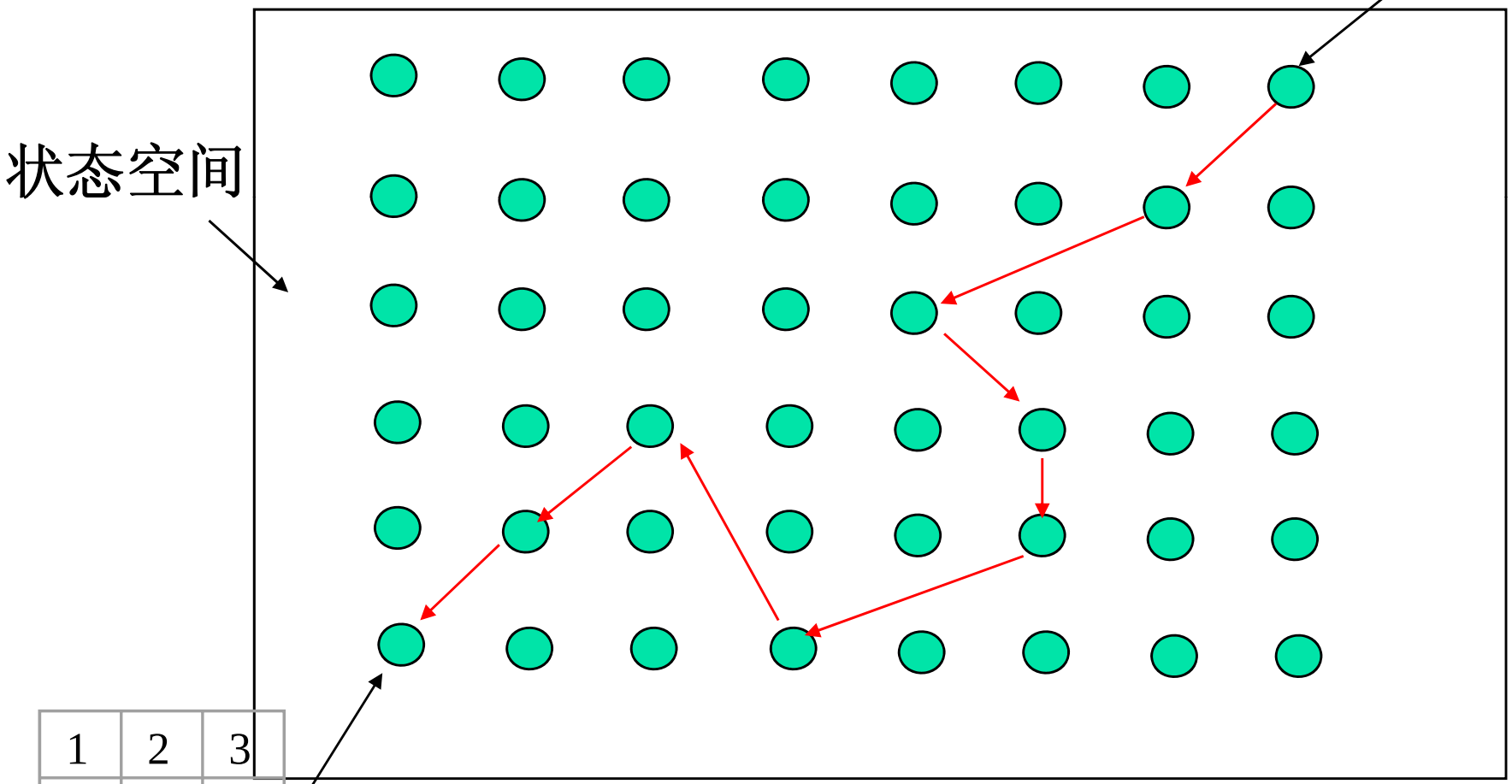
几个问题

- 目标状态是否确定?
  - 确定: 定理证明, **eight-puzzle**
  - 不确定: 求积分, 下棋;
    - 确定目标的性质;
- 问题的解: 路径(解路径) / 目标状态;
  - 需要路径:下棋
  - 不需要路径:电路设计
  - 需要 / 不需要: 诊病
- 约束条件
  - 目标状态不确定时, 用来约束目标状态的性质;
  - **X+Y=4:** 非整数解 / 整数解

- 多解性;
  - **X+Y=4:**整数解
- 最优解
  - 评价标准/判断准则;
  - **min(x*y)**
  - 北京->上海: 时间最短/费用最少
- 最优解是否唯一?
  - 下棋

搜索问题

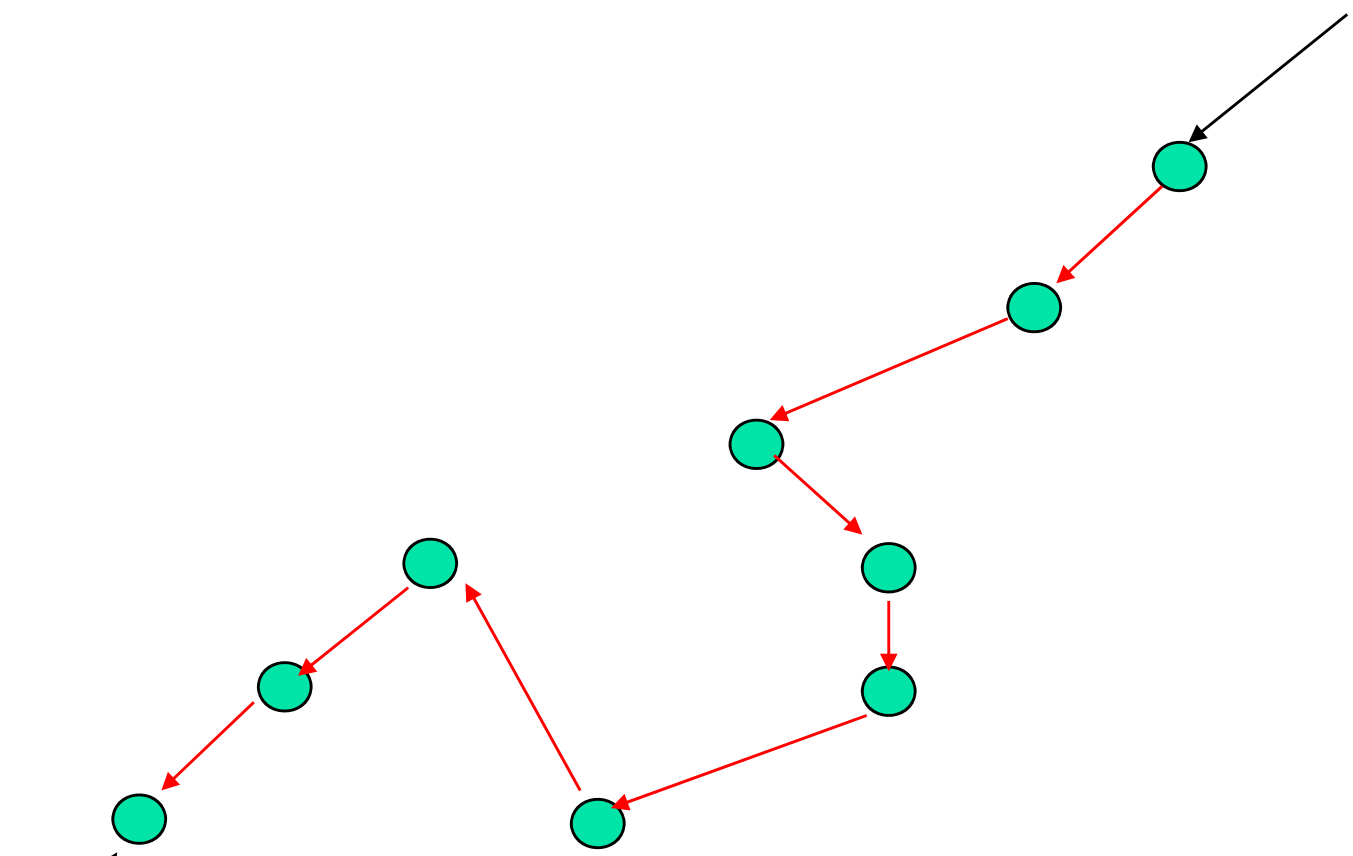
2	3	7
	5	1
4	8	6



1	2	3
8		4
7	6	5

搜索不是检索

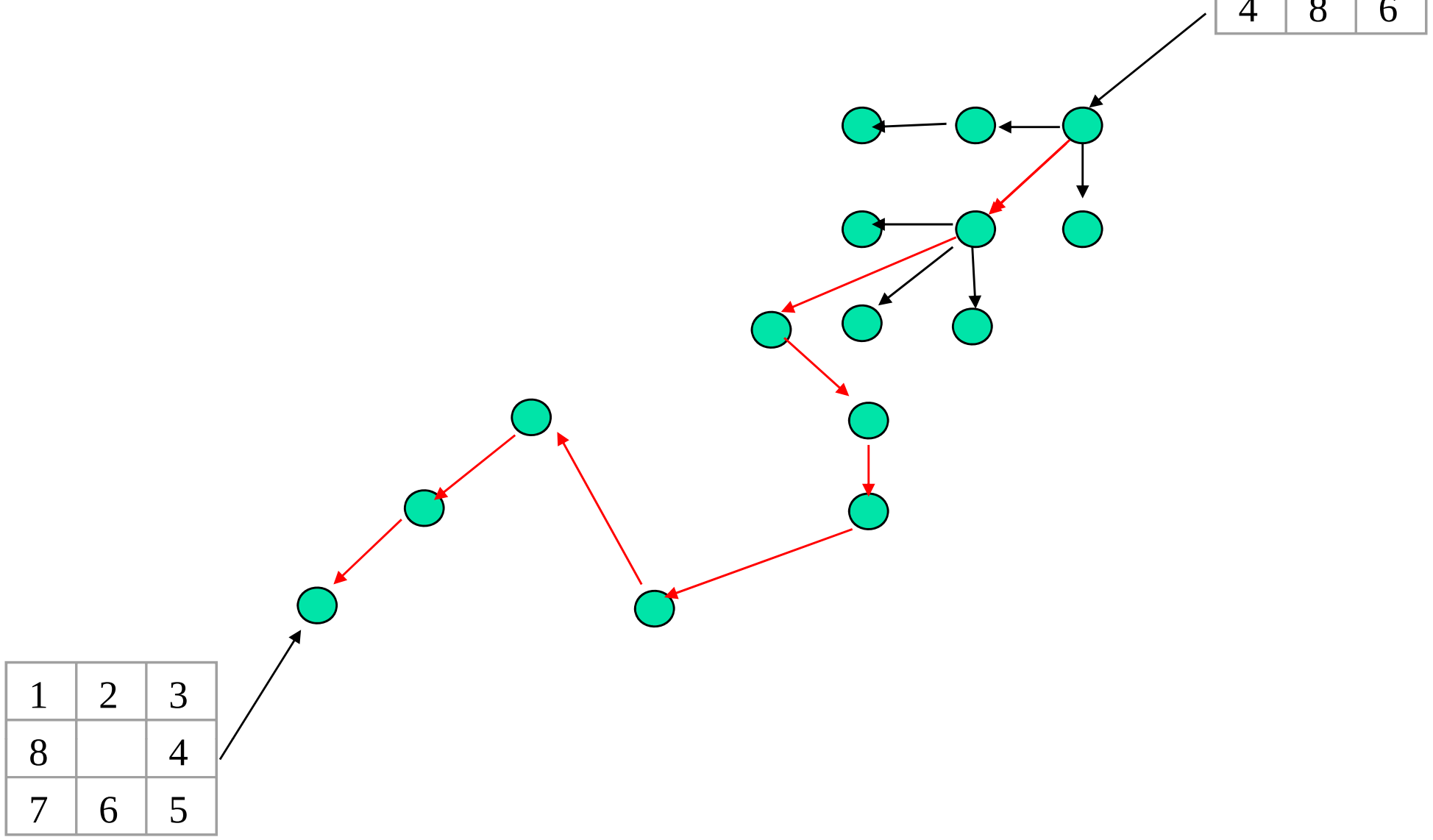
2	3	7
	5	1
4	8	6



1	2	3
8		4
7	6	5

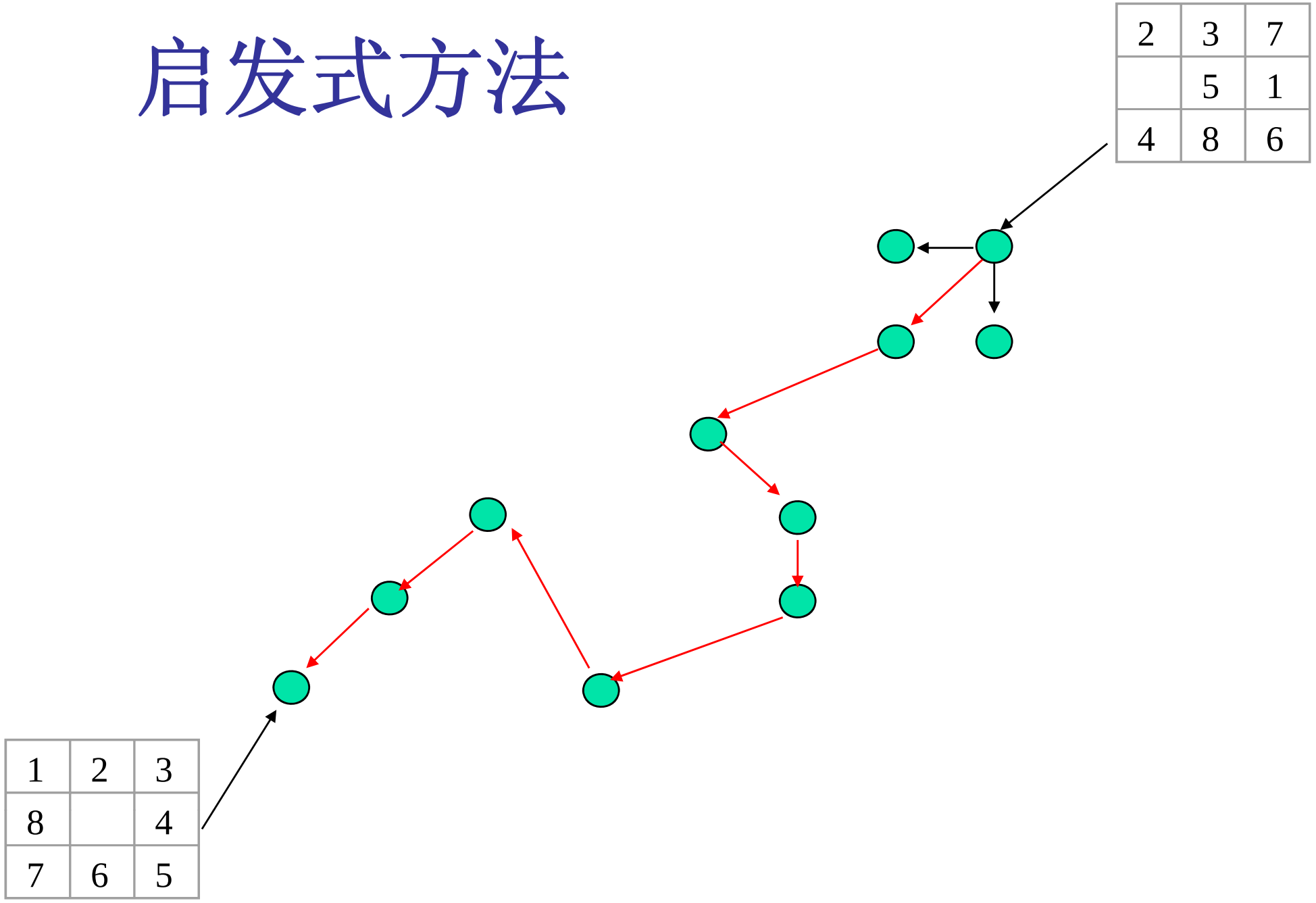
难点

2	3	7
	5	1
4	8	6



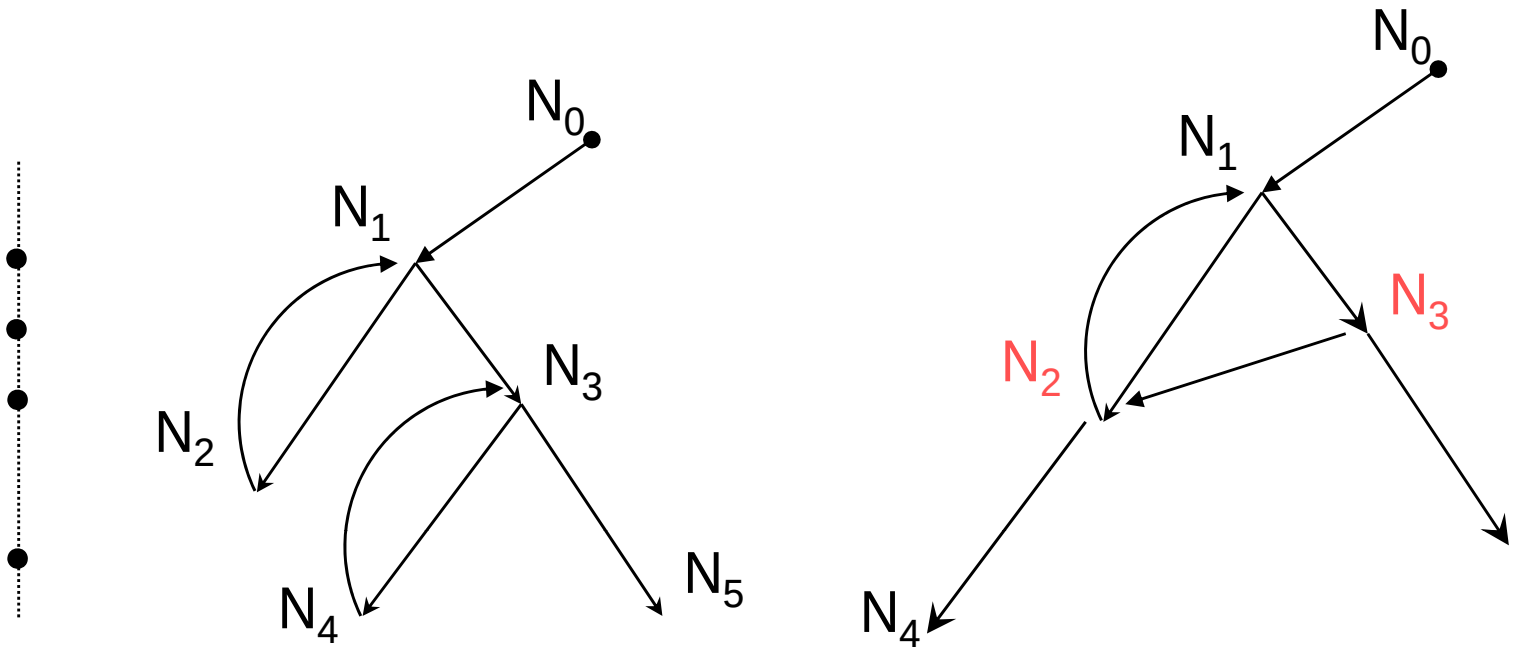
1	2	3
8		4
7	6	5

启发式方法



控制策略

- 不可撤回的控制策略;
- 试探性控制策略
  - 回溯型
  - 图搜索



不可撤回的控制策略

- 例子: **eight-puzzle**
  - 评价函数:**f**

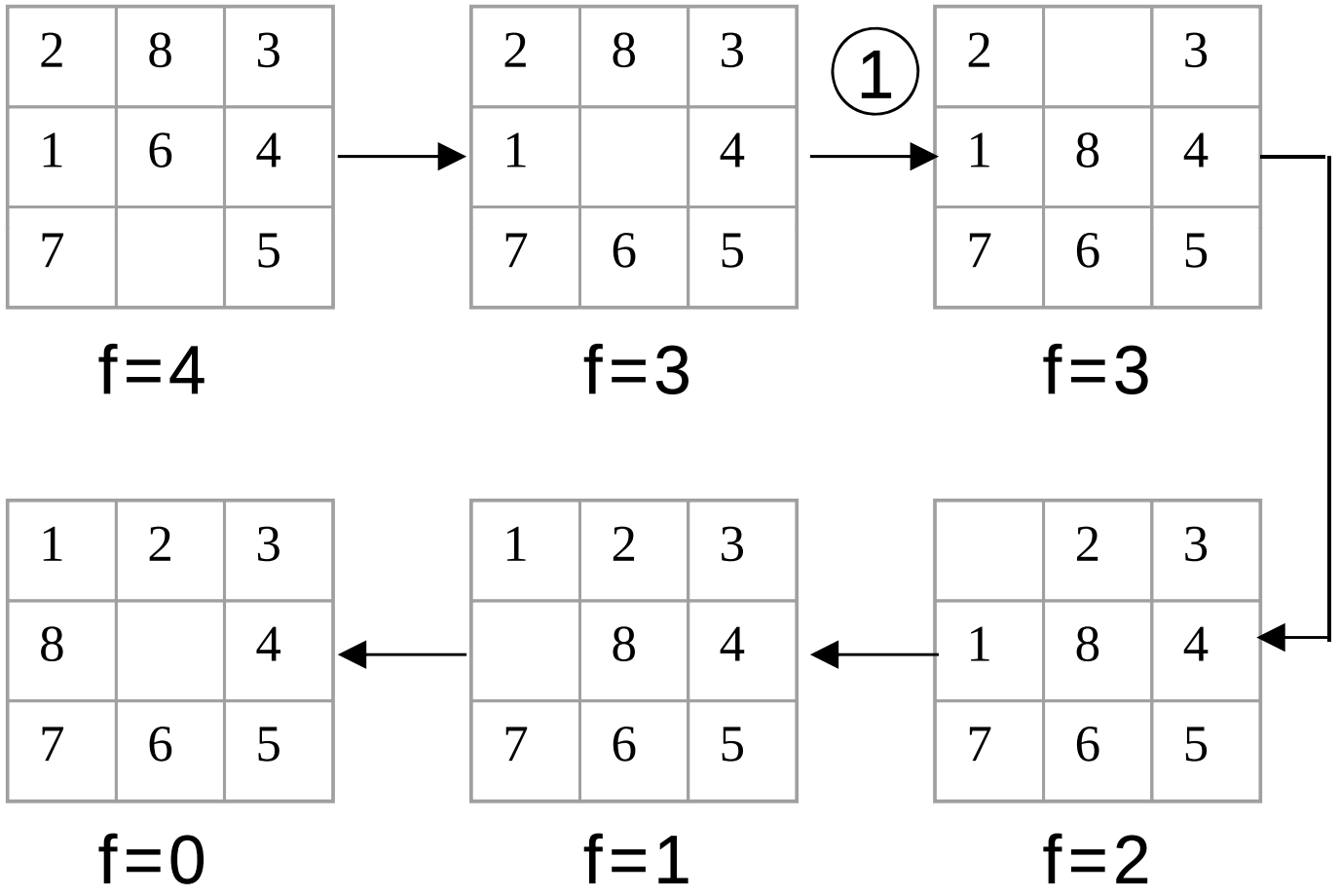
2	8	3
1	6	4
7		5

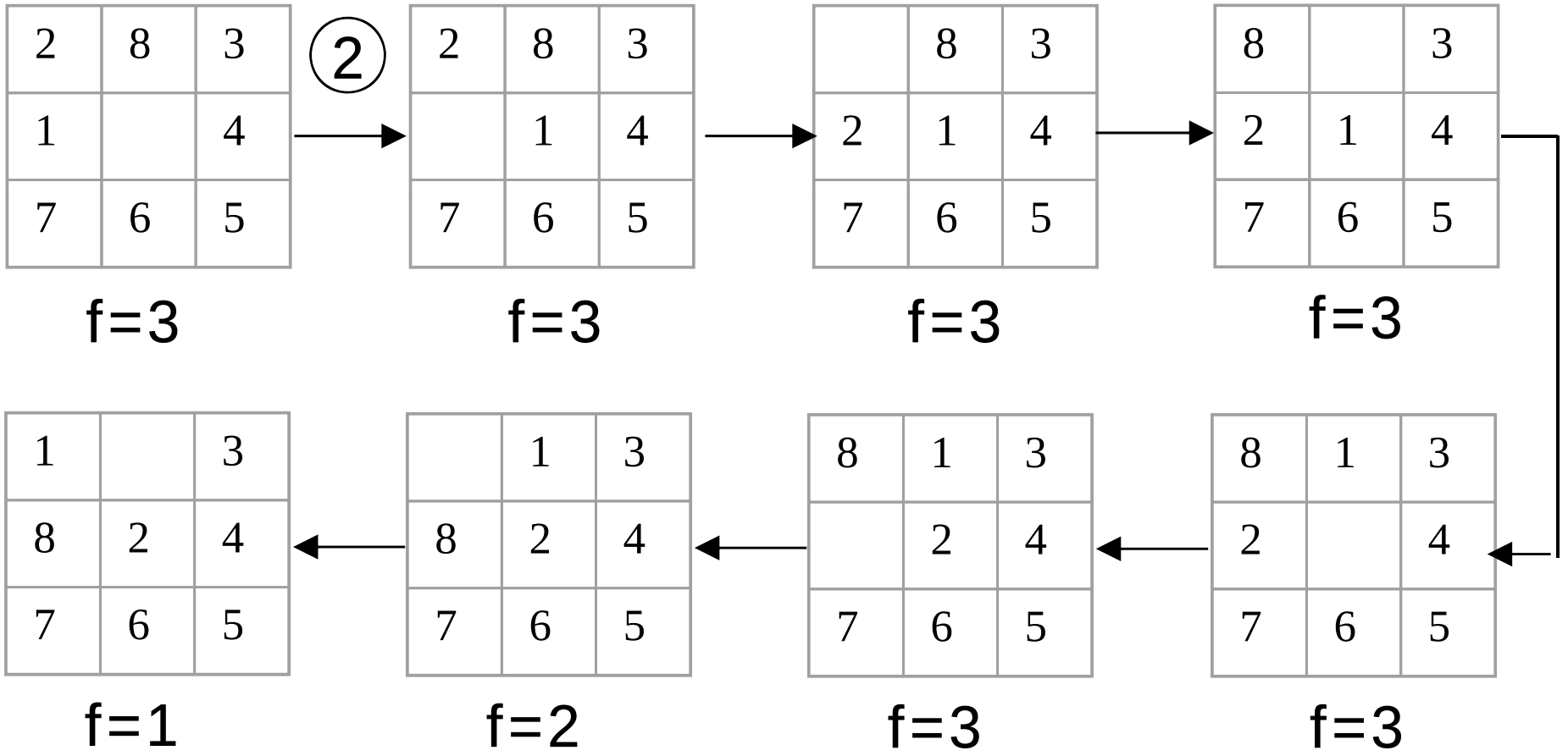
与

1	2	3
8		4
7	6	5

的差异为**4**

规定: 评价函数非增;


2 8 3
1 6 4
7 5
f=4
2 8 3
1 4
7 6 5
f=3
1
2 3
1 8 4
7 6 5
f=3
1 2 3
8 4
7 6 5
f=0
1 2 3
8 4
7 6 5
f=1
2 3
1 8 4
7 6 5
f=2


2 8 3
1 4
7 6 5
2
f=3
2 8 3
1 4
7 6 5
f=3
8 3
2 1 4
7 6 5
f=3
8 3
2 1 4
7 6 5
f=3
8 1 3
2 4
7 6 5
f=3
8 1 3
2 4
7 6 5
f=3
1 3
8 2 4
7 6 5
f=2
1 3
8 2 4
7 6 5
f=1

可能无解

1	2	5
	8	4
7	6	3

f=2

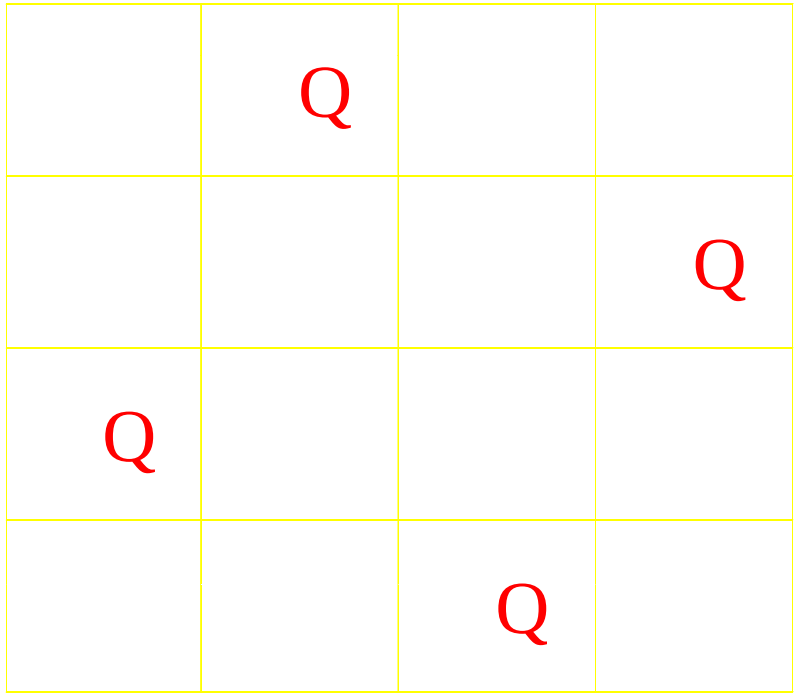
1	2	3
	8	4
7	6	5

目标

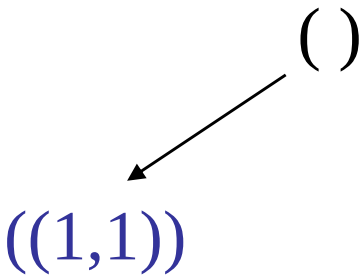
- 回溯策略
- 图搜索
- 无信息搜索
- 启发式搜索
- A*算法的可采纳性

回溯策略

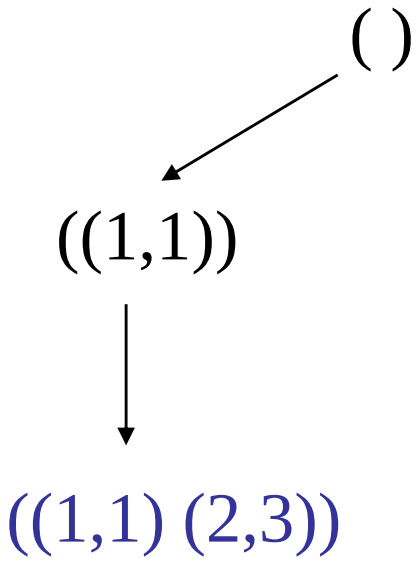
- 例：四皇后问题



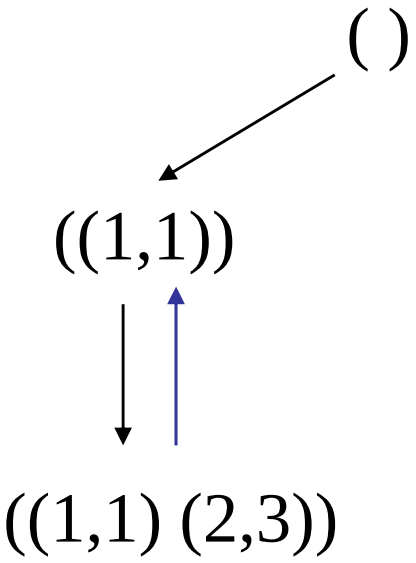
( )



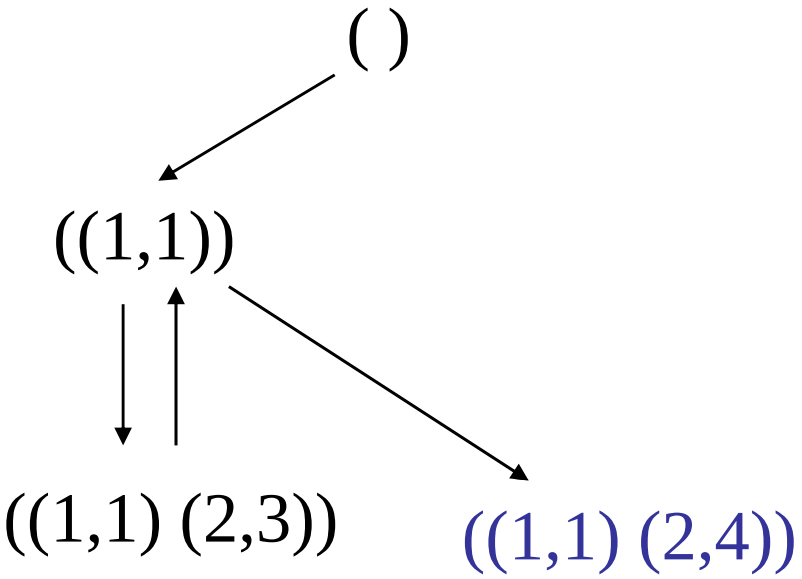
Q			



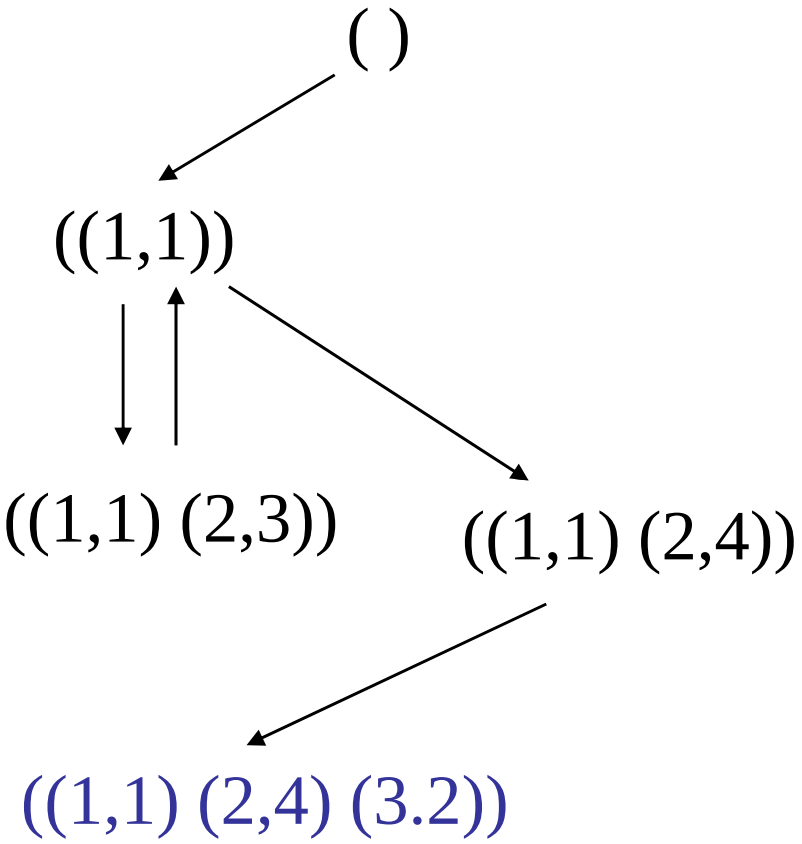
Q			
		Q	


( )
((1,1))
((1,1) (2,3))

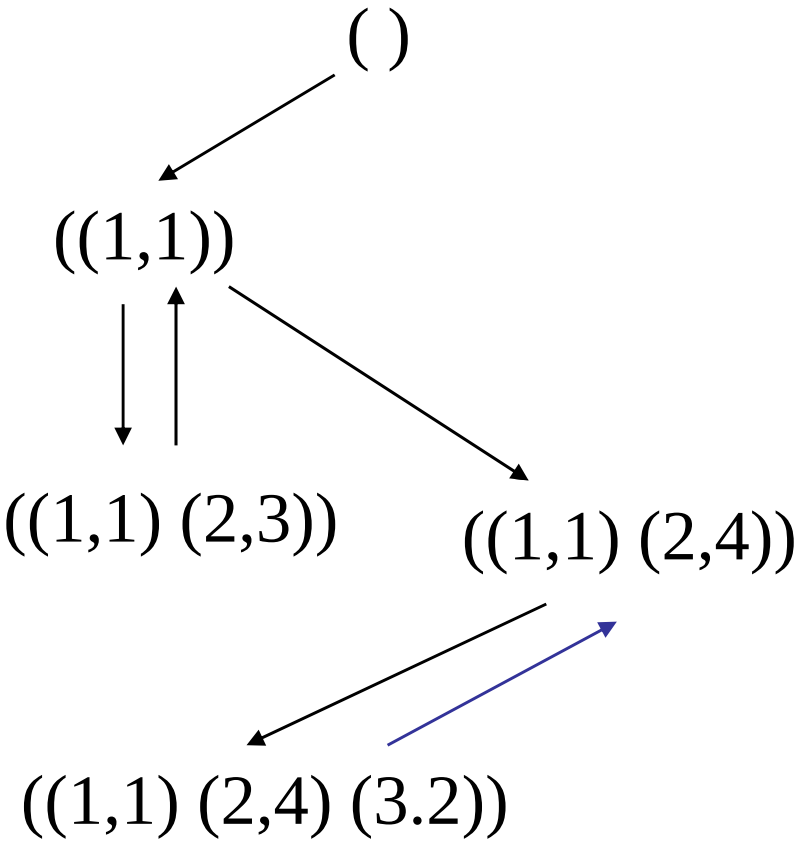

Q



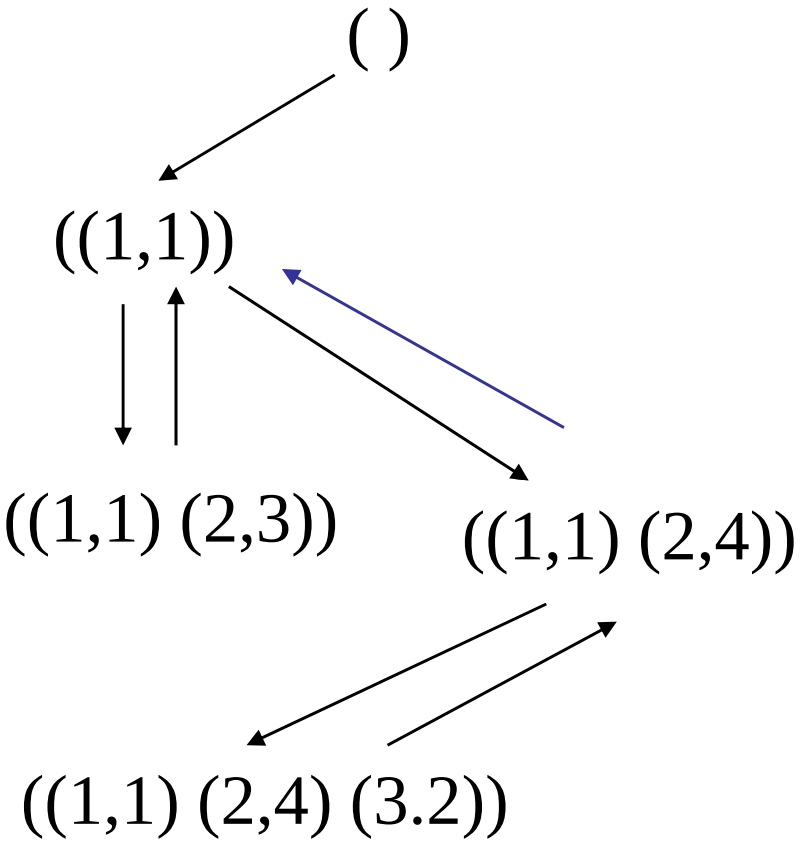
Q			
			Q



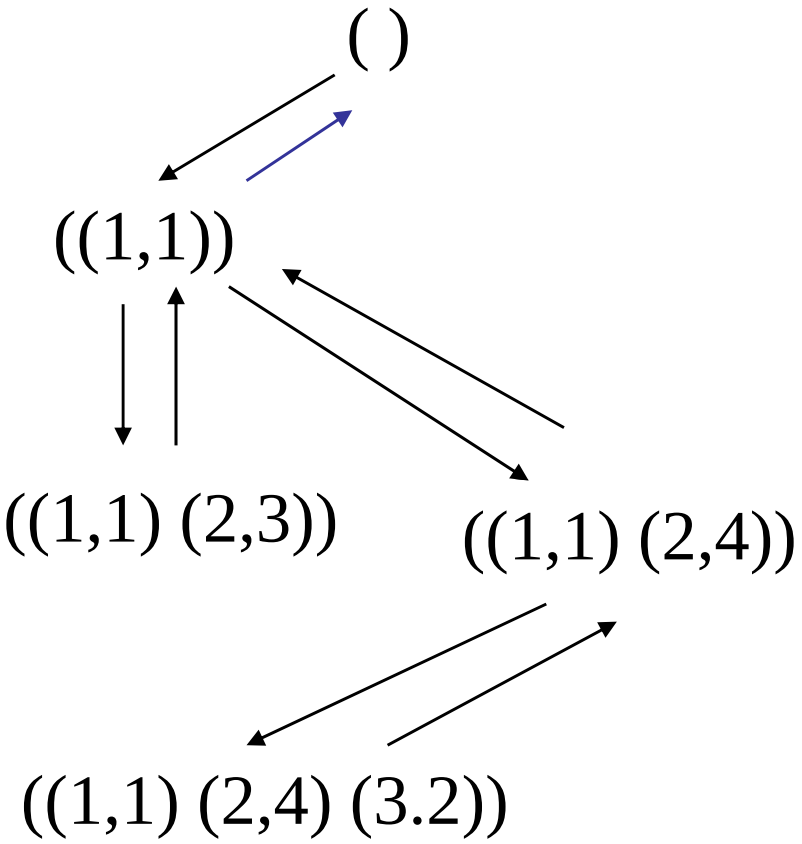
Q			
			Q
	Q		

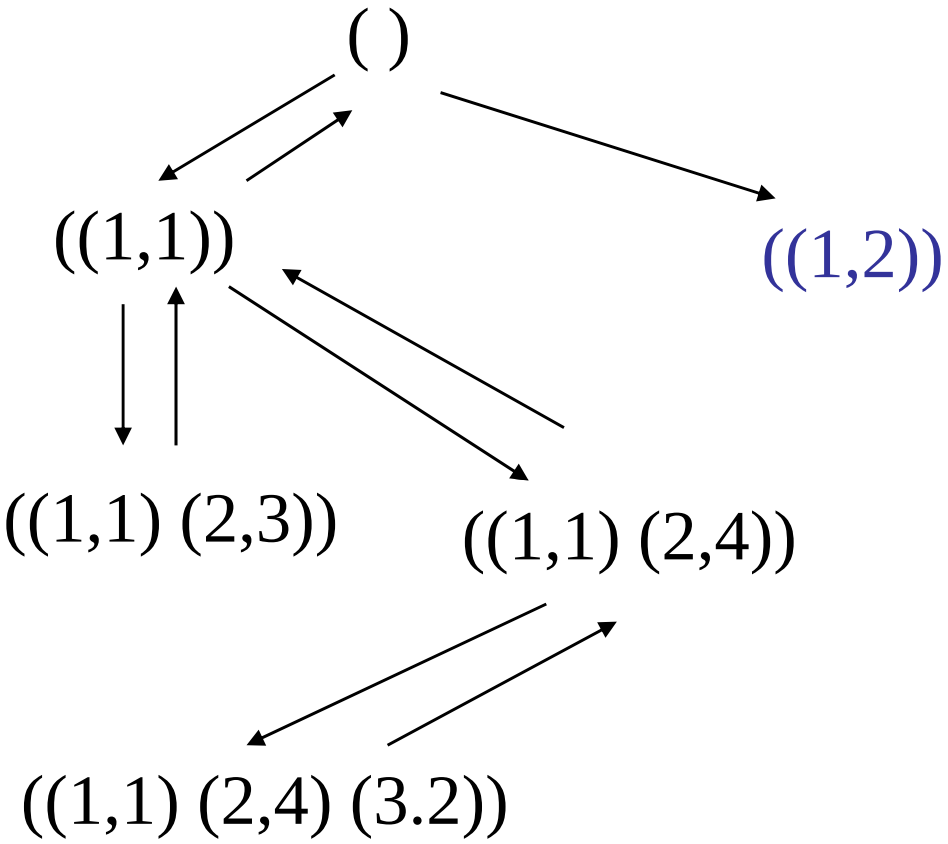


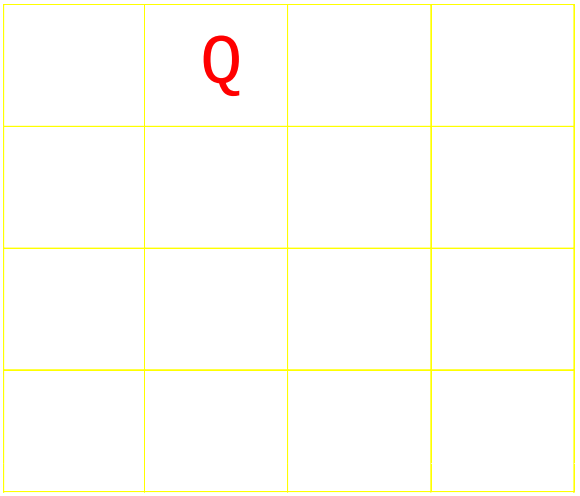
Q			
			Q

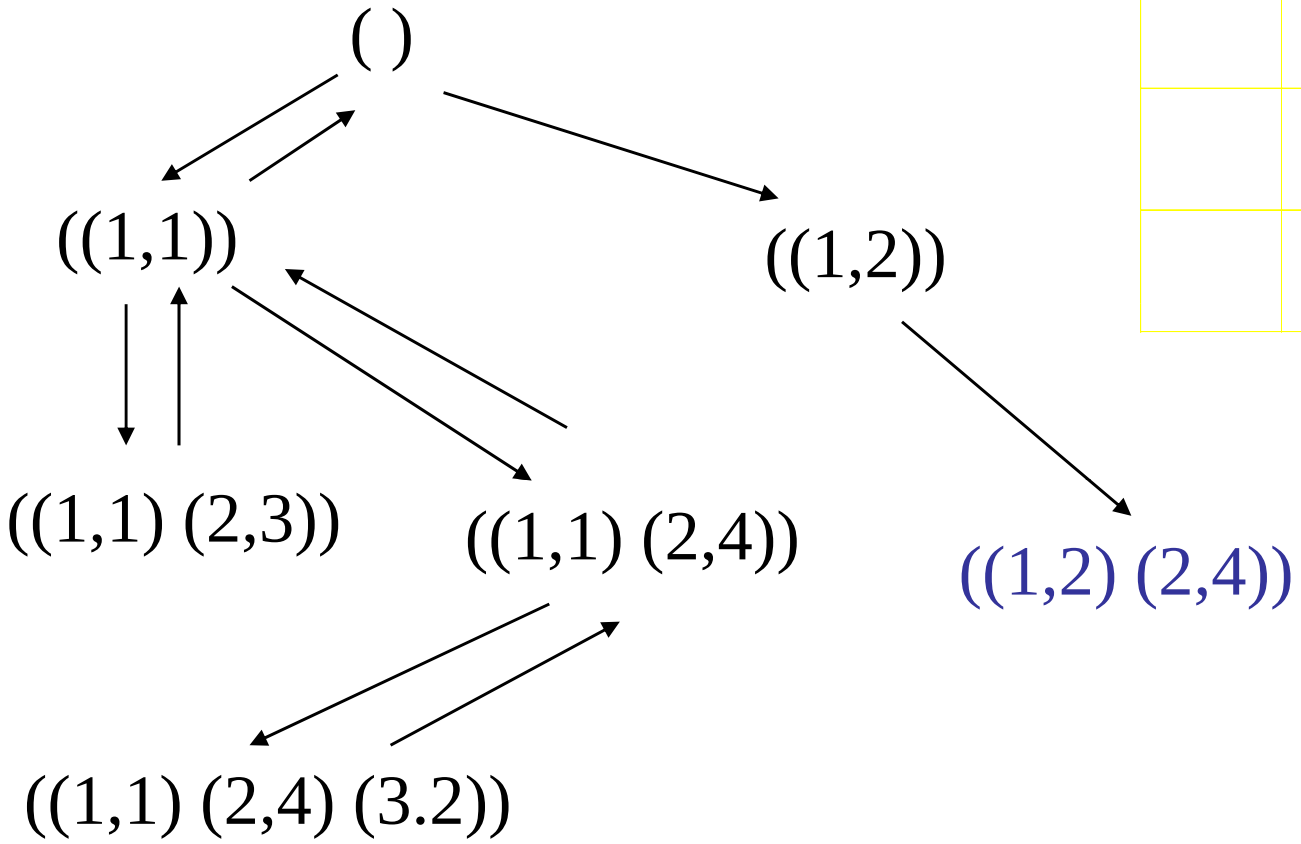

( )
((1,1))
((1,1) (2,3))
((1,1) (2,4))
((1,1) (2,4) (3.2))


Q

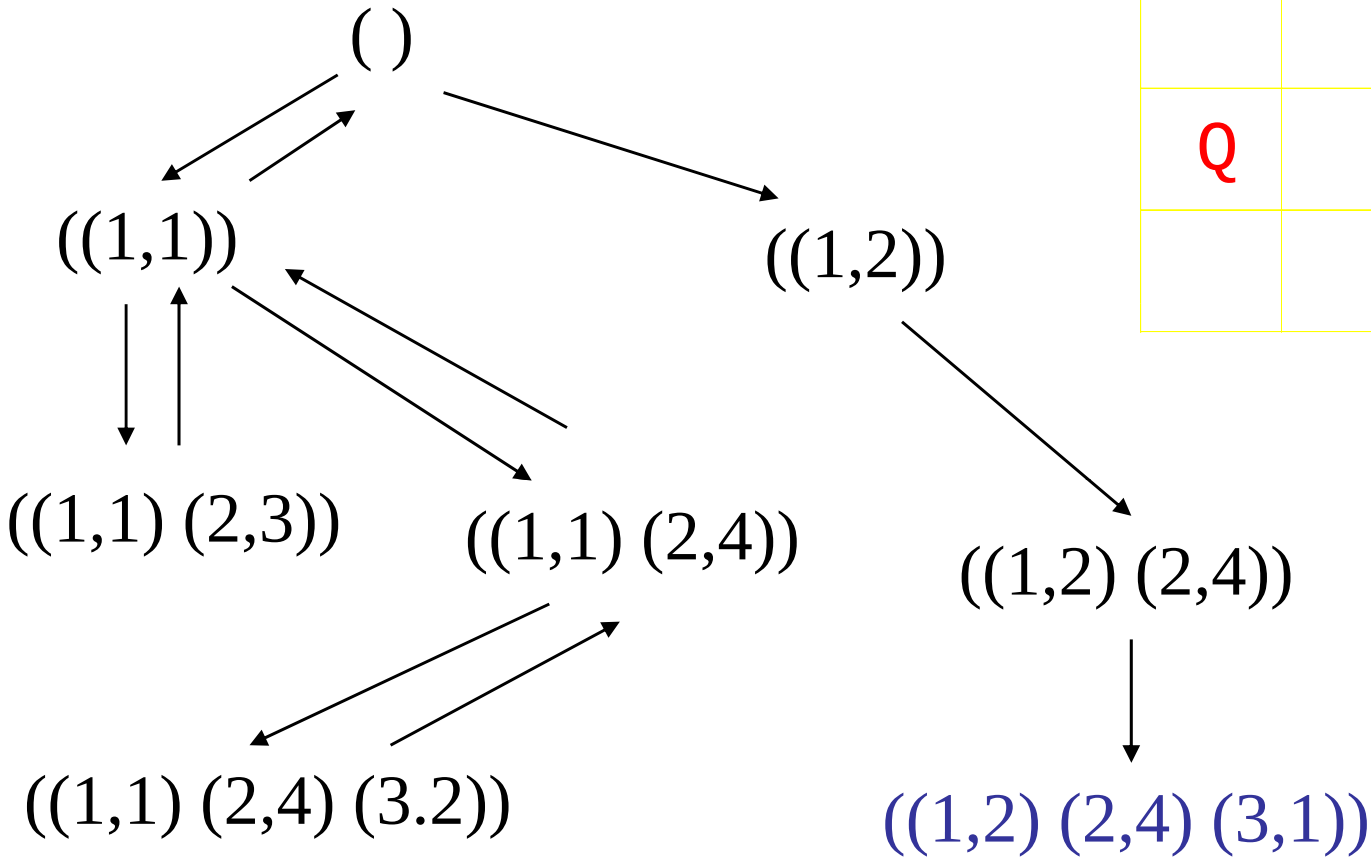

( )
((1,1))
((1,1) (2,3))
((1,1) (2,4))
((1,1) (2,4) (3.2))


( )
((1,1))
((1,2))
((1,1) (2,3))
((1,1) (2,4))
((1,1) (2,4) (3.2))

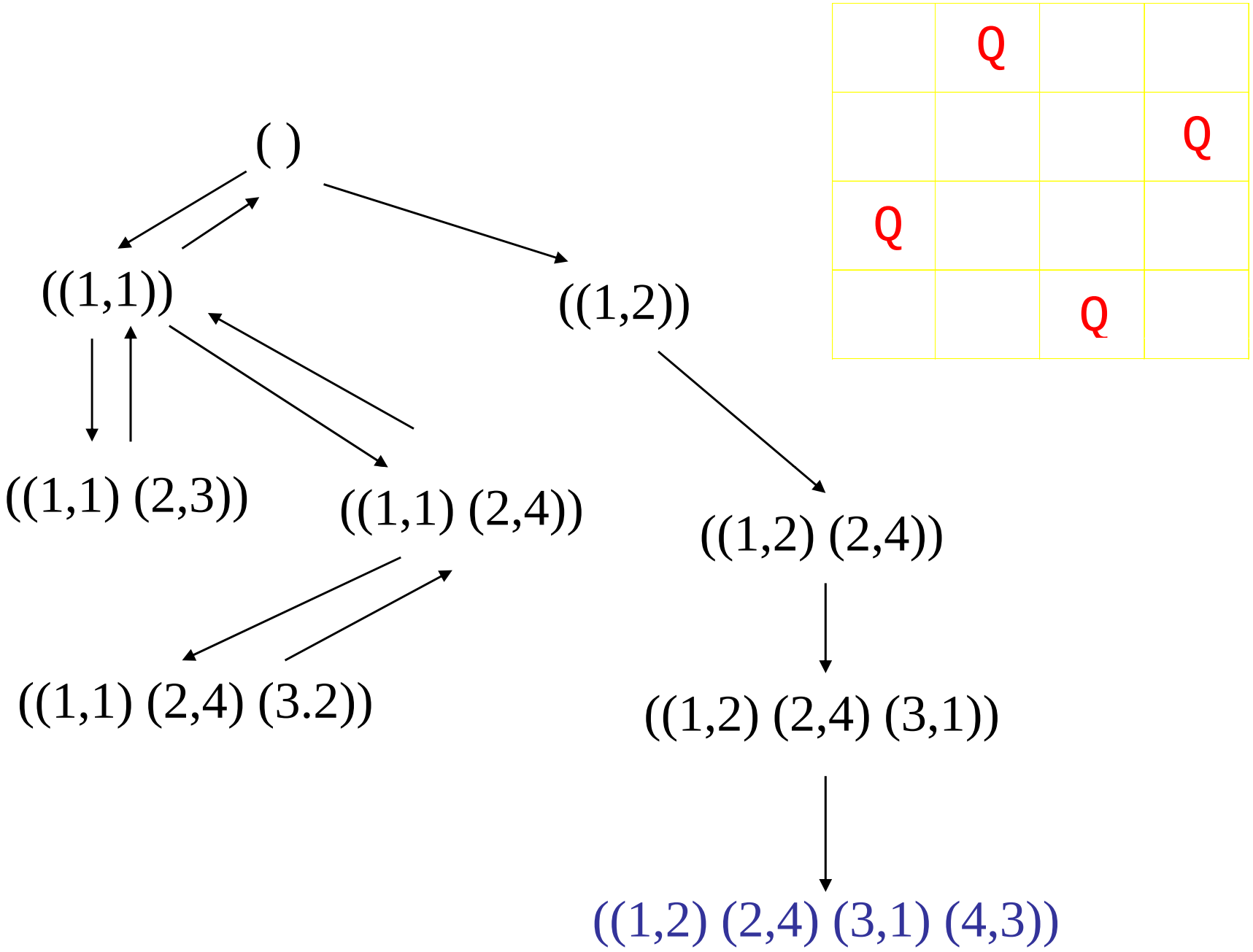

Q



	Q		
			Q



	Q		
			Q
Q			


( )
((1,1))
((1,2))
((1,1) (2,3))
((1,1) (2,4))
((1,2) (2,4))
((1,1) (2,4) (3.2))
((1,2) (2,4) (3,1))
((1,2) (2,4) (3,1) (4,3))
Q
Q
Q
Q

回溯搜索算法

BACKTRACK（DATA）

DATA： 当前状态。

返回值：从当前状态到目标状态的路径
（以规则表的形式表示）
或**FAIL**。

回溯搜索算法

```
   BACKTRACK(DATA)
1      IF Term(DATA) RETURN NIL;
2      IF Deadend(DATA) RETURN FAIL;
3      Rules:=Apprules(DATA);
4  LOOP: IF Null(Rules) RETURN FAIL;
5      R:=First(Rules);
6      Rules:=TAIL(Rules);
7      Rdata:=Gen(R, DATA);
8      Path:=BACKTRACK(Rdata);
9      IF Path =FAIL GO LOOP;
10     Else RETURN Cons(R, Path);
```

分析节点的情况

- 失败节点：返回**FAIL**
  - 步骤**2:** 领域相关条件判断
  - 步骤**4:** 无规则可用时
- 成功节点
  - 步骤**1:** 叶节点, 返回**NIL**
  - 步骤**10:**中间节点, 返回包含**R**的路径
- 如果成功，返回一条包含**R**的路径,（$R_{i1}, R_{i2},\ldots\ldots R_{in}$）

存在问题及解决办法

- 问题和解决方法：
  - 深度问题
    - 对搜索深度加以限制
  - 死循环问题
    - 状态重复: **A→B，B→C, C→A**
    - 记录从初始状态到当前状态的路径

修正的回溯搜索算法1

BACKTRACK1（DATALIST）

DATALIST：从初始到当前的状态表（逆向）

返回值：从当前状态到目标状态的路径

（以规则表的形式表示）

或**FAIL**。

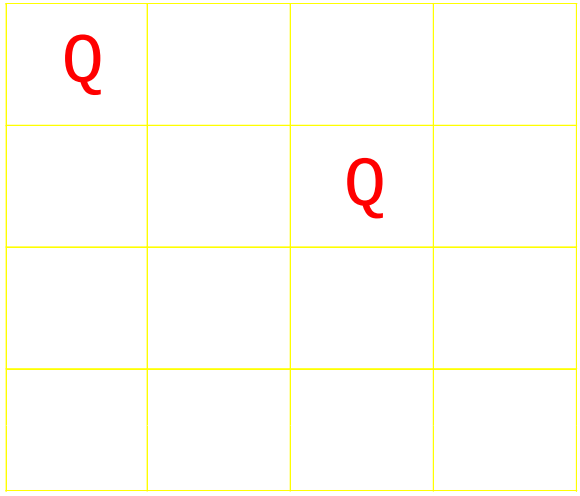
修正的回溯搜索算法1

```
1     DATA:=FIRST(DATALIST)
2     IF MEMBER(DATA, TAIL(DATALIST))
      RETURN FAIL;
3     IF TERM(DATA) RETURN NIL;
4     IF DEADEND(DATA) RETURN FAIL;
5     IF LENGTH(DATALIST)>BOUND
      RETURN FAIL;
6     RULES:=APPRULES(DATA);
7  LOOP: IF NULL(RULES) RETURN FAIL;
8     R:=FIRST(RULES);
```

```
9     RULES:=TAIL(RULES);
10    RDATA:=GEN(R, DATA);
11    RDATALIST:=CONS(RDATA, DATALIST);
12    PATH:=BACKTRCK1(RDATALIST)
13    IF PATH=FAIL GO LOOP;
14    RETURN CONS(R, PATH);
```

一些深入的问题

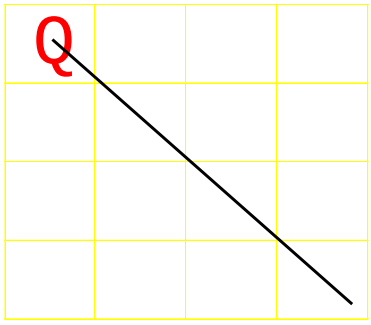
- 失败原因分析、多步回溯



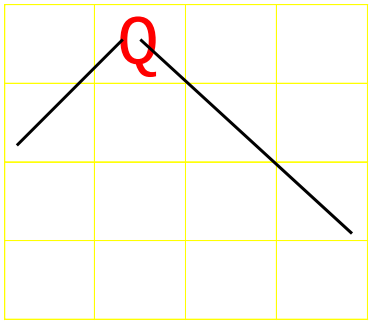
- 回溯搜索中知识的利用

  基本思想：

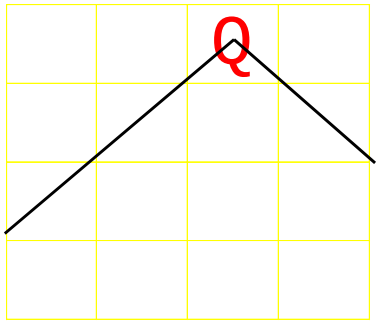
  尽可能选取划去对角线上位置数最少的。



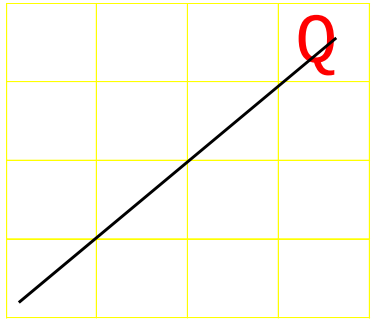
4



3



3



4

- 回溯策略
- 图搜索
- 无信息搜索
- 启发式搜索
- A*算法的可采纳性