中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

解空间搜索

张文生　研究员 首席教授

中国科学院自动化研究所
中科院大学人工智能学院

2020-10-16

要进行问题求解，首先要讨论的是对问题及其解的精确定义，我们将通过一些实例来说明如何描述一个**问题**及其**解**。接着我们介绍一些求解此类问题的通用的搜索算法。首先讨论**无信息的**（uninformed）搜索算法——无信息是指算法除了问题定义本身没有任何其他信息。尽管这些算法有的可以用于求解任何问题，但此类算法效率都不好。另一方面，**有信息**（Informed）的搜索算法，利用给定的知识引导能够更有效地找到解。

这章中我们会把任务环境简化，限定问题的解是一组有固定顺序的行动。更一般的情况是 Agent 的行动决策将随着未来感知数据的改变而改变。

讲课内容

一、问题求解

二、回溯式策略

三、图搜索策略

四、无信息搜索

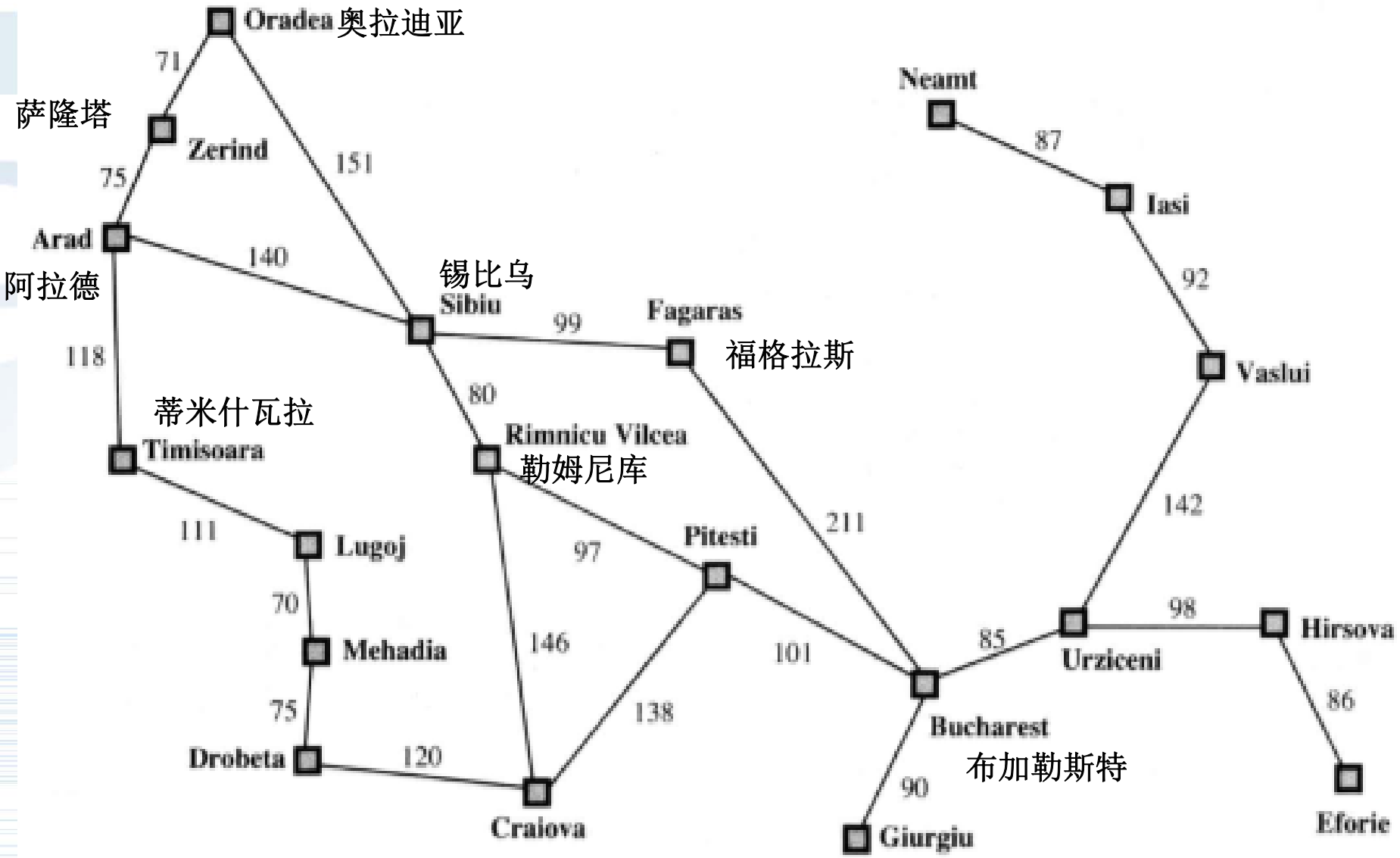
五、启发式搜索

五、A*算法与性质

问题求解的假设和概念

为达到目标，寻找这样的行动序列的过程被称为**搜索**。搜索算法的输入是问题，输出是问题的解，以行动序列的形式返回问题的**解**。解一旦找到，它所建议的行动将会付诸实施。这被称为**执行**阶段。那么，我们就完成了对 Agent 的简单设计，即“形式化、搜索、执行”，如图 3.1 所示。在完成对目标和对待求解问题的形式化之后，Agent 调用搜索过程进行问题求解。然后 Agent 用得到的解来导引行动，按照问题求解给出的解步骤逐一实施——通常是执行序列中的第一个行动——从序列中删除已完成的步骤。一旦解被执行，Agent 将形式化新的目标。

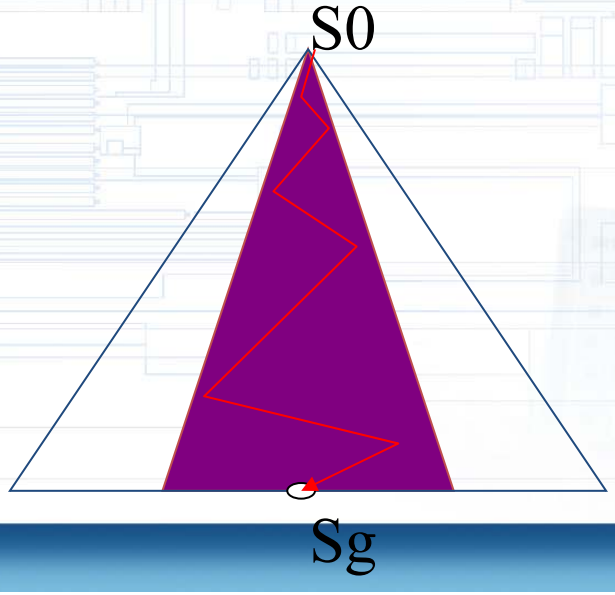
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```



罗马尼亚主要城市旅行简图

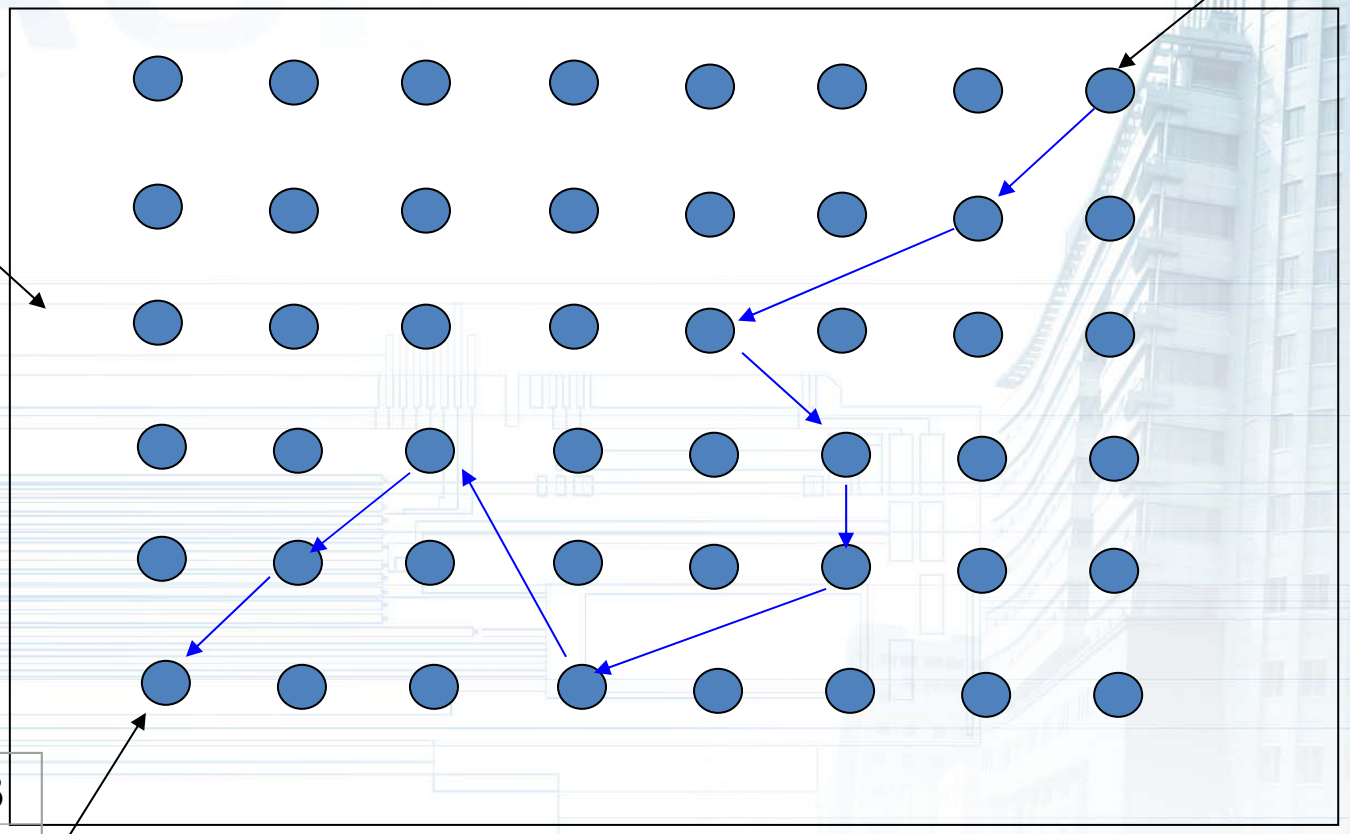
搜索

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



搜索问题

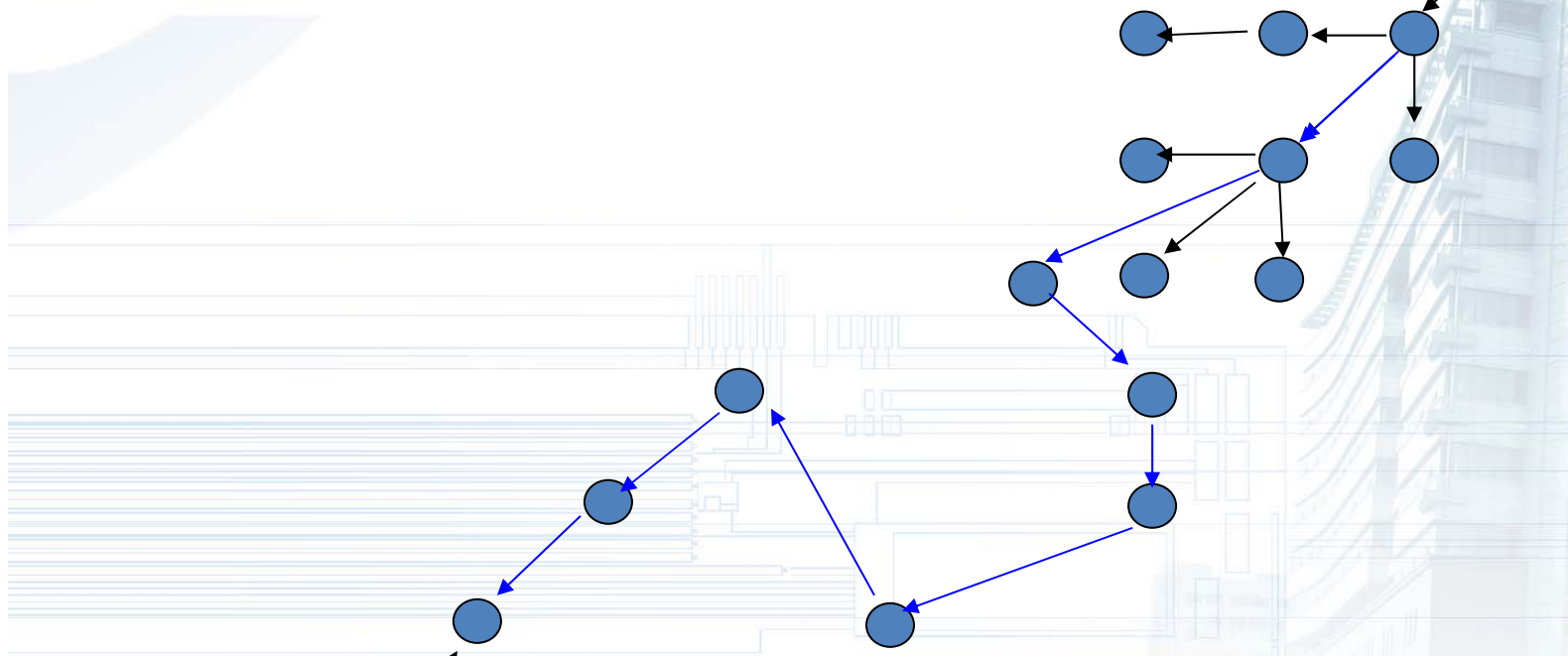
2	3	7
	5	1
4	8	6



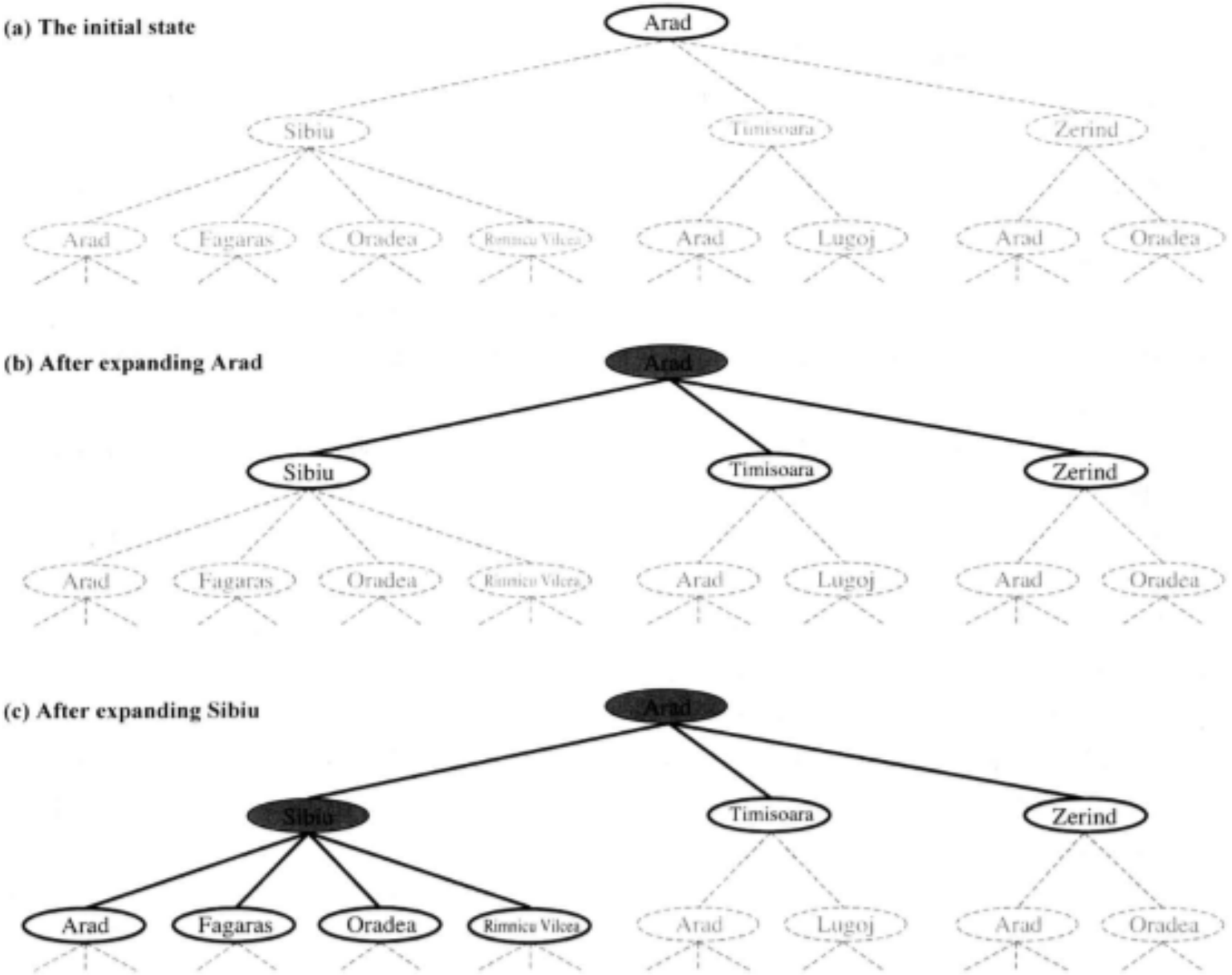
1	2	3
8		4
7	6	5

难点

2	3	7
	5	1
4	8	6



1	2	3
8		4
7	6	5



求解罗马尼亚问题的部分搜索树。要注意的是已被扩展过的结点用阴影表示；
已经生成但未被扩展的结点用粗实线表示；尚未生成的结点用浅虚线表示

```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        expand the chosen node, adding the resulting nodes to the frontier
```

```
function GRAPH-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    initialize the explored set to be empty
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier
        if the node contains a goal state then return the corresponding solution
        add the node to the explored set
        expand the chosen node, adding the resulting nodes to the frontier
            only if not in the frontier or explored set
```

一般的树搜索和图搜索算法的非形式化描述。用粗斜体标出的图搜索算法的部分内容是指需要额外处理重复状态

讲课内容

一、问题求解

二、回溯式策略

三、图搜索策略

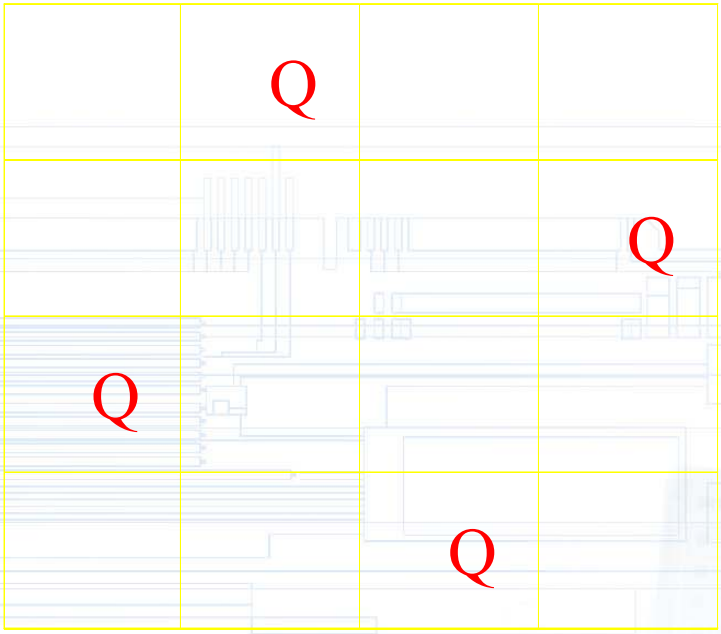
四、无信息搜索

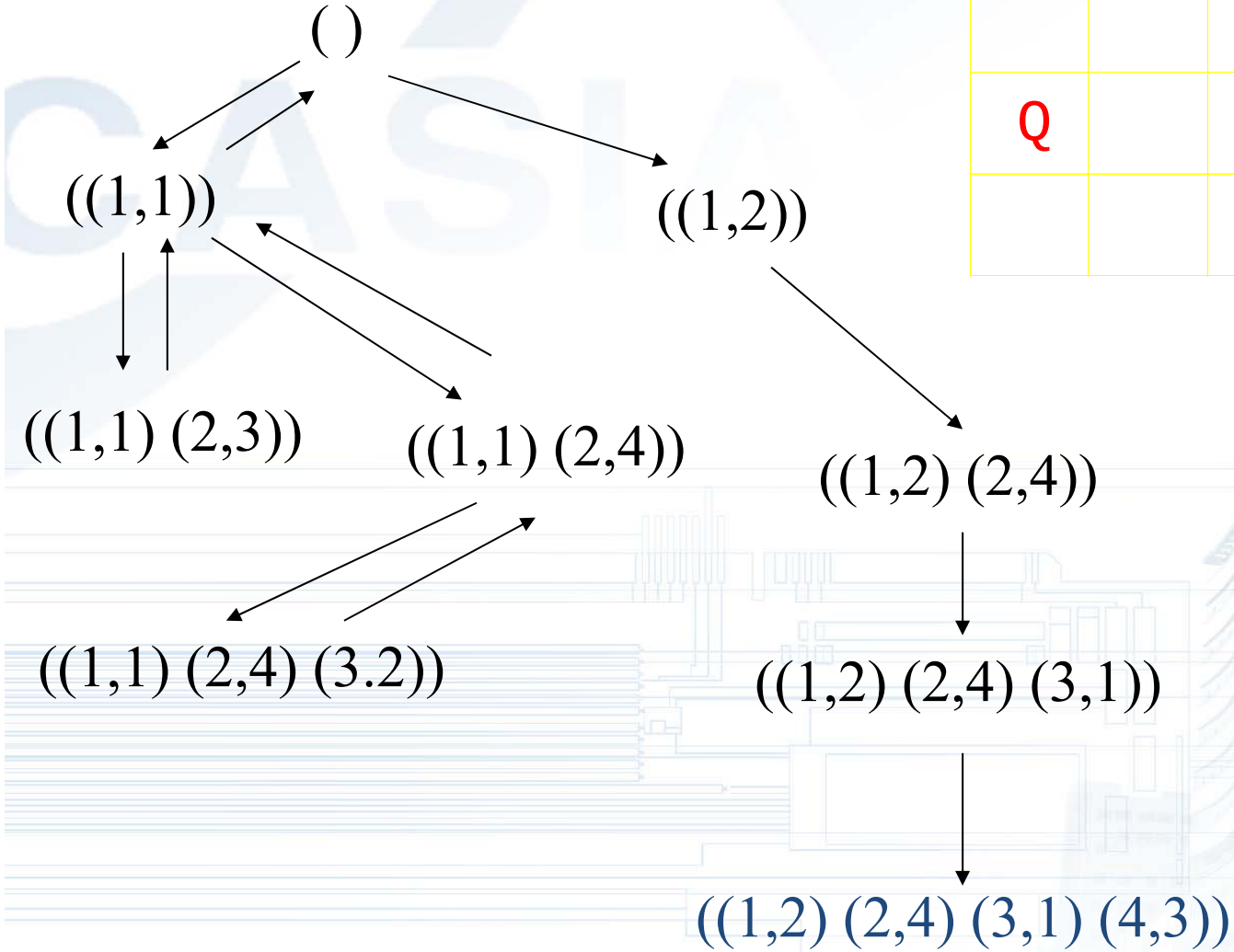
五、启发式搜索

五、A*算法与性质

回溯式策略

- 例：四皇后问题





	Q		
			Q
Q			
		Q	

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);
```

修正的回溯搜索算法


```
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);
```

讲课内容

一、问题求解

二、回溯式策略

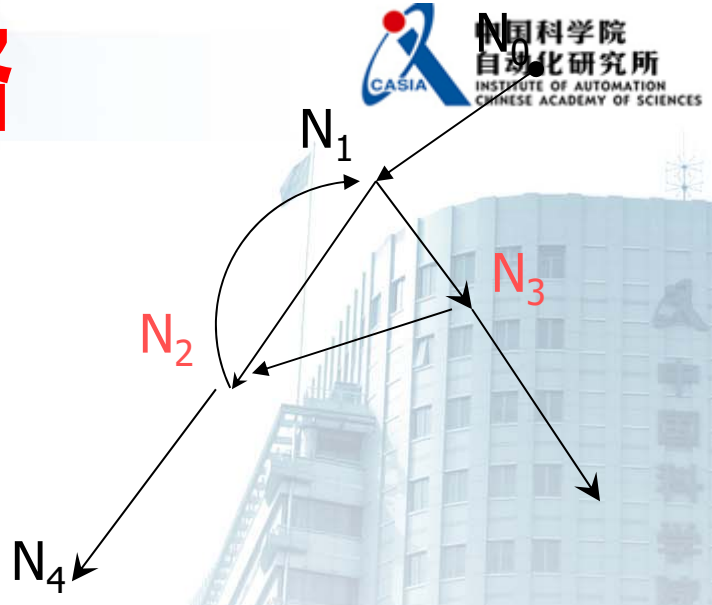
三、图搜索策略

四、无信息搜索

五、启发式搜索

五、A*算法与性质

图搜索策略

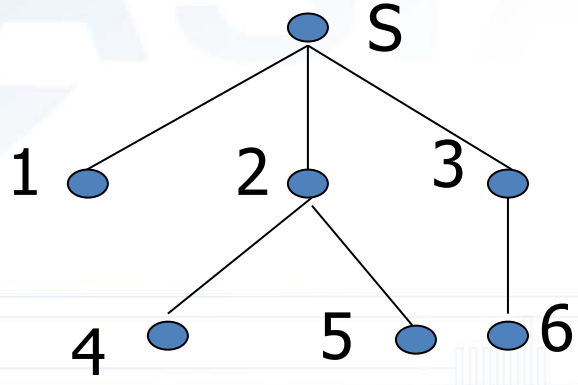


- 问题的引出
  - 八皇后问题 / 找最短路径
- 回溯与图搜索的区别
  - 回溯: 放弃的状态永远放弃;
  - 图搜索: 放弃的状态以后还可能再用;
- 算法:
  - 回溯搜索：只保留从初始状态到当前状态的一条路径。
  - 图搜索：保留所有已经搜索过的路径。

图搜索的思路

- OPEN表
  - 已经生成但未扩展节点
- CLOSED表
  - 已扩展节点
- 扩展节点i生成节点j
- 指针
- 调整指针

例子

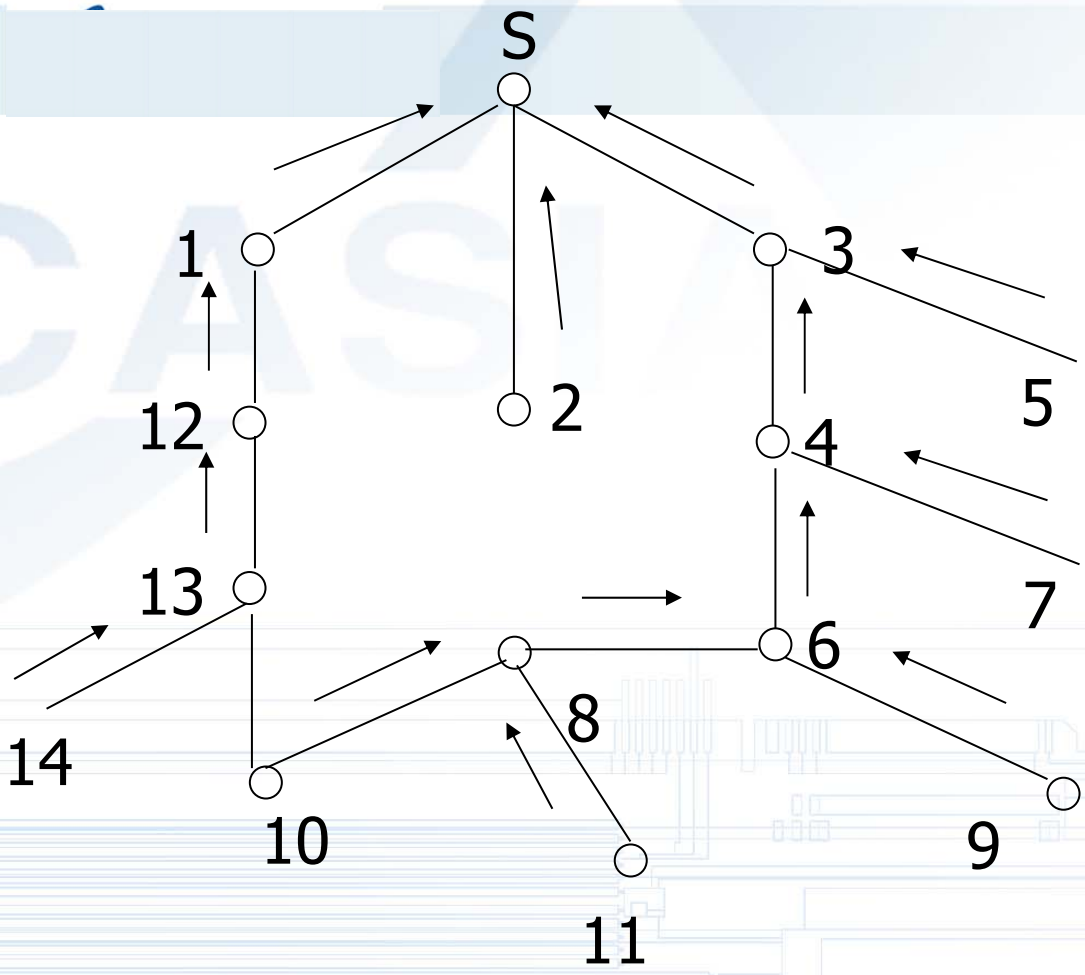


OPEN	CLOSE
{S}	{}
{}	{S}
{1,2,3}	{S}
{2,1,3}	{S}
{1,3}	{S,2}
{1,3,4,5}	{S,2}
{3,1,4,5}	{S,2}
{1,4,5}	{S,2,3}
{1,4,5,6}	{S,2,3}

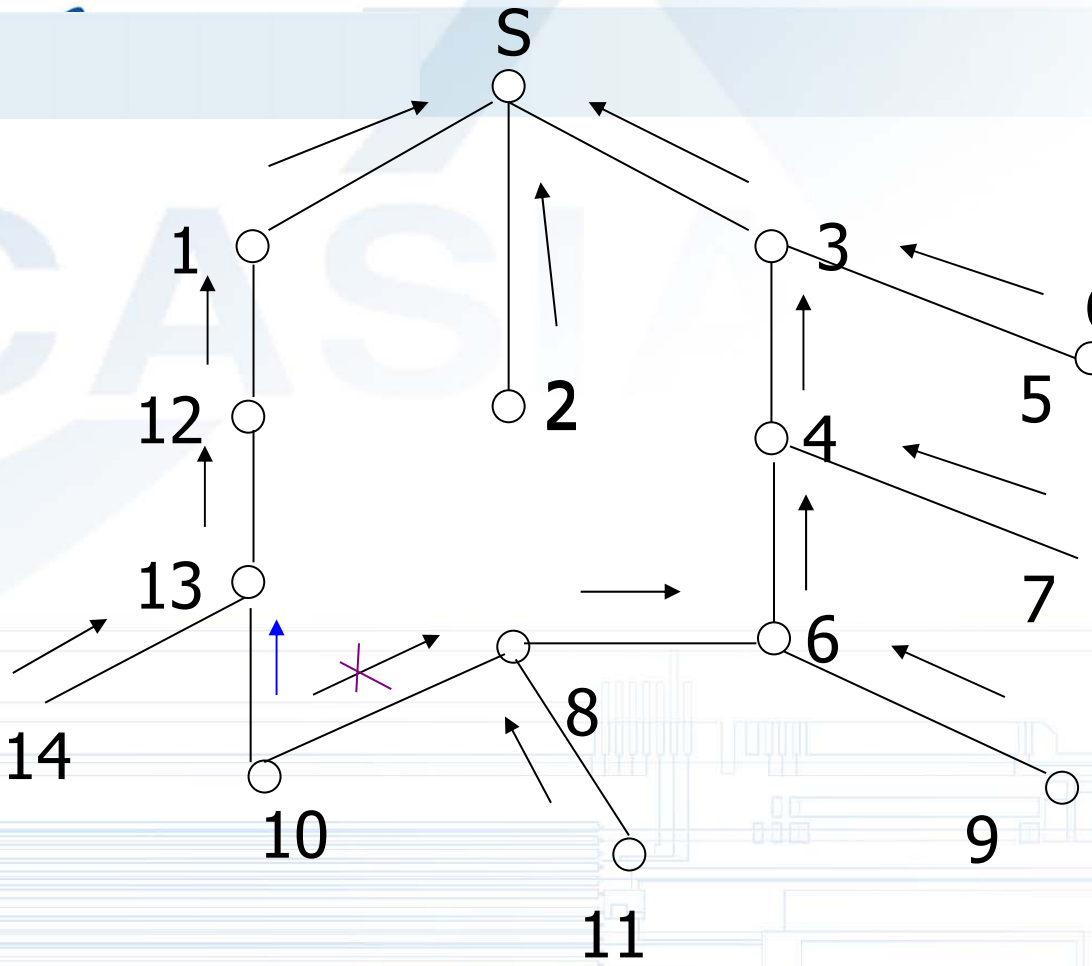
GRAPHSEARCH

1．建立一个只有起始节点S组成的图G，把S放到OPEN表中；
2．建立一个CLOSED表，置为空；

3．While(!NULL (OPEN))
a) 从OPEN表中取出（并删除）第一个节点n放入CLOSED表；
b) 如果n是目标节点，成功结束；
c) 扩展节点n，产生节点n的不是n的祖先的后继节点集合 M，把M中的这些成员作为n的后继加入G中；
d) 对M的那些既不在OPEN表中又不在CLOSED表中的成员，加入到OPEN表中，并建立它们到n的指针；对那些在OPEN表中的成员，决定是否调整它们的指针；对那些在CLOSED表中的成员，决定是否调整它们的指针，并决定是否调整它们在G中的后裔的指针；
e) 对OPEN表中的节点排序；
4．返回FAIL；

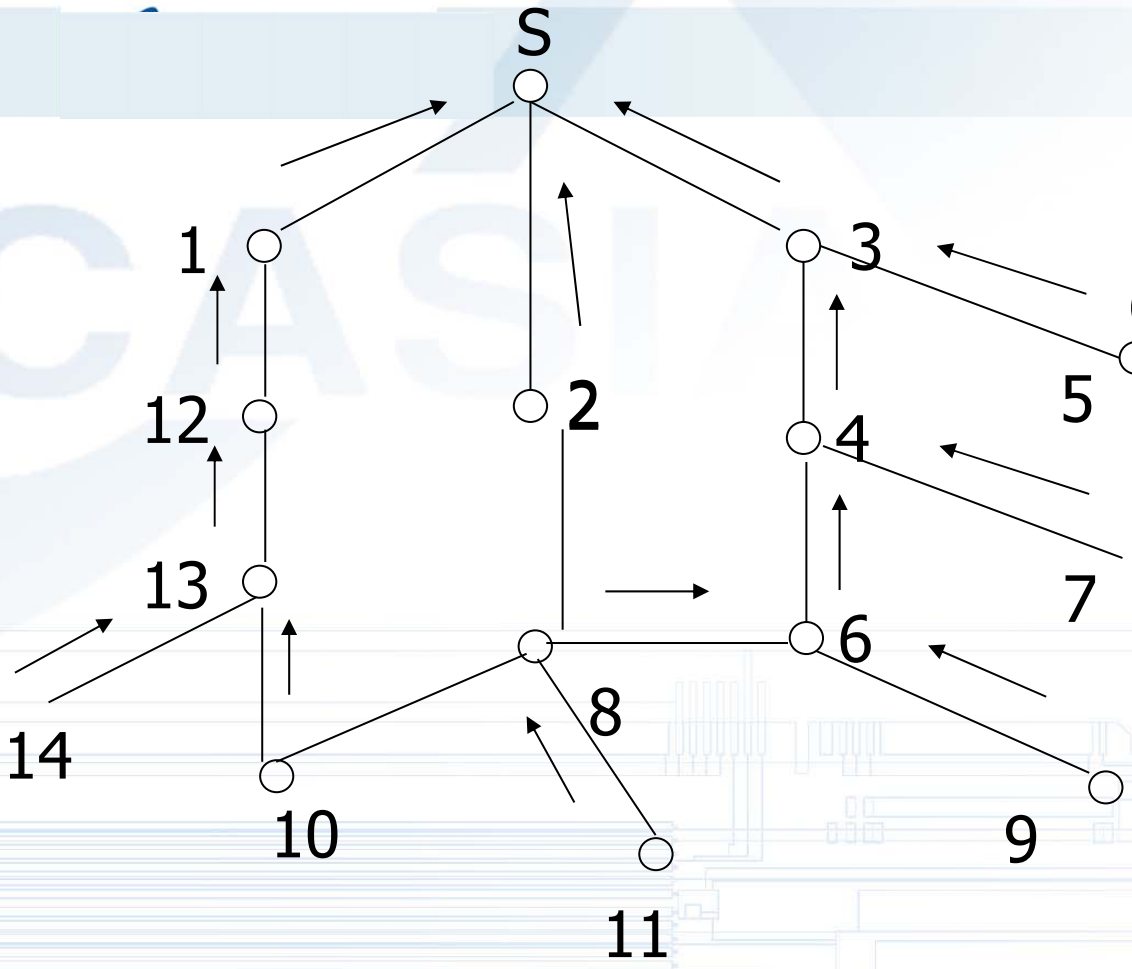


OPEN	CLOSE
{S}	{}
{1,2,3}	{S}
{3,1,2}	{S}
{4,5,1,2}	{S,3}
{6,7,5,1,2}	{S,3,4}
{8,9,7,5,1,2}	{S,3,4,6}
{10,11,9,7,5,1,2}	{S,3,4,6,8}
{13,10,11,9,7,5,2}	**{S,3,4,6,8,1,12}**
{14,10,11,9,7,5,2}	**{S,3,4,6,8,1,12,13}**

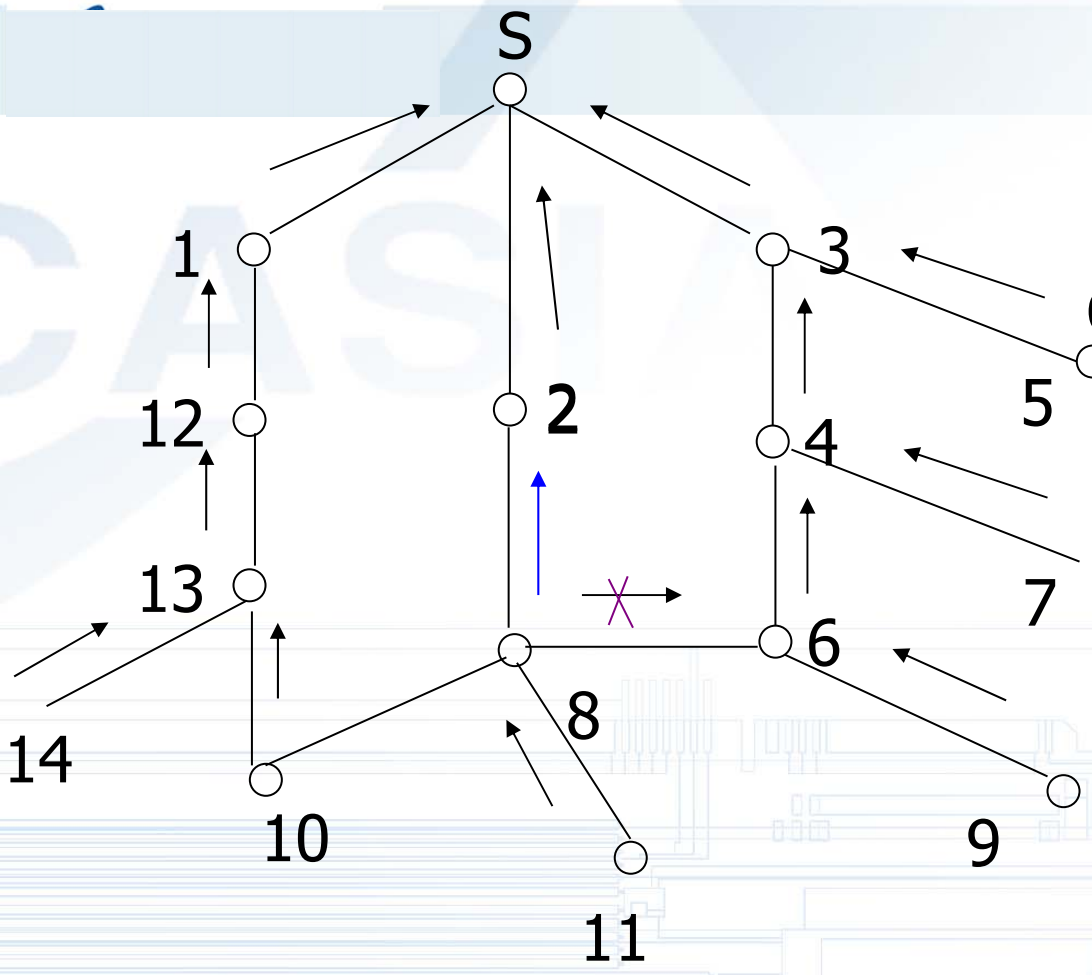


OPEN	CLOSE
{S}	{}
{1,2,3}	{S}
{3,1,2}	{S}
{4,5,1,2}	{S,3}
{6,7,5,1,2}	{S,3,4}
{8,9,7,5,1,2}	{S,3,4,6}
{10,11,9,7,5,1,2}	{S,3,4,6,8}
{13,10,11,9,7,5,2}	{S,3,4,6,8,1,12}

OPEN表中的节点修改指针

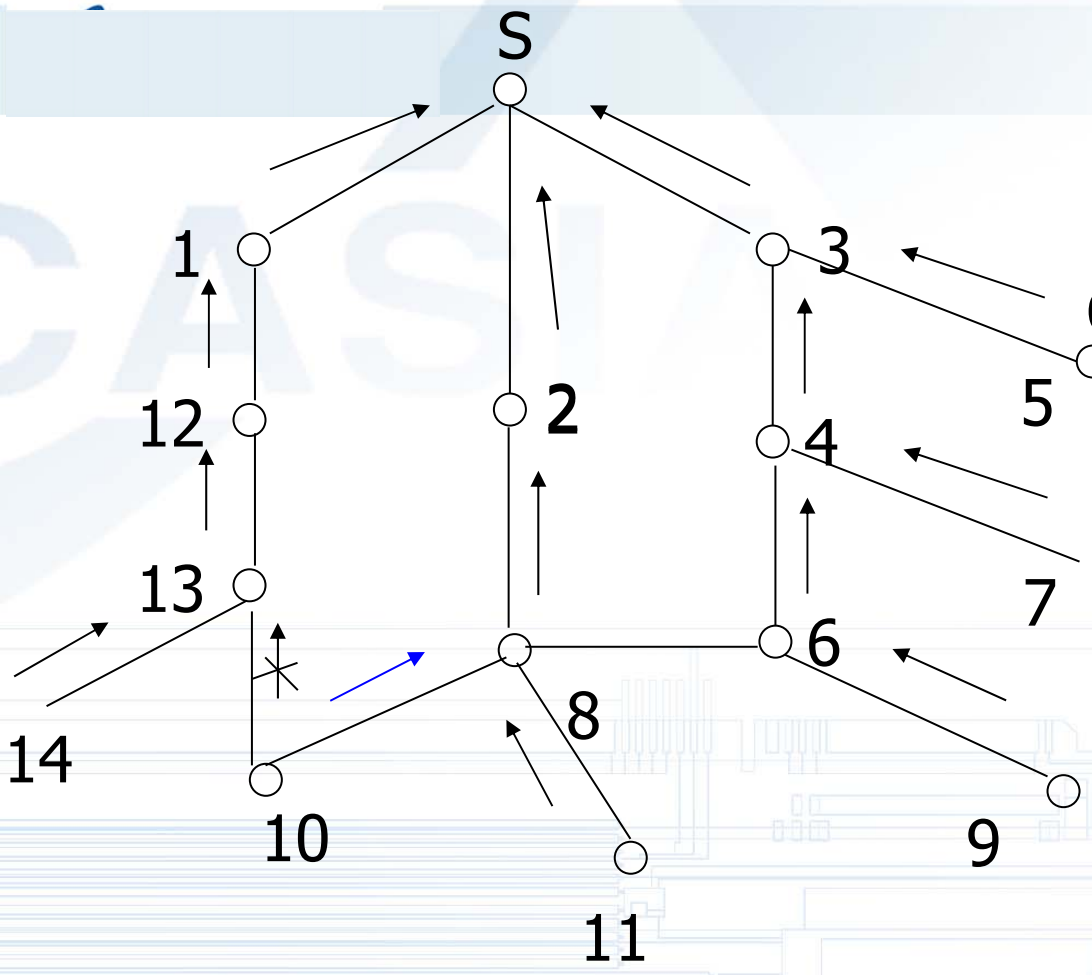


OPEN	CLOSE
{S}	{}
{1,2,3}	{S}
{3,1,2}	{S}
{4,5,1,2}	{S,3}
{6,7,5,1,2}	{S,3,4}
{8,9,7,5,1,2}	**{S,3,4,6}**
{10,11,9,7,5,1,2}	**{S,3,4,6,8}**
{13,10,11,9,7,5,2}	**{S,3,4,6,8,1,12}**
{13,10,11,9,7,5}	**{S,3,4,6,8,1,12,2}**



OPEN	CLOSE
{S}	{}
{1,2,3}	{S}
{3,1,2}	{S}
{4,5,1,2}	{S,3}
{6,7,5,1,2}	{S,3,4}
{8,9,7,5,1,2}	{S,3,4,6}
{10,11,9,7,5,1,2}	{S,3,4,6,8}
{13,10,11,9,7,5,2}	**{S,3,4,6,8,1,12}**
{13,10,11,9,7,5}	**{S,3,4,6,8,1,12,2}**

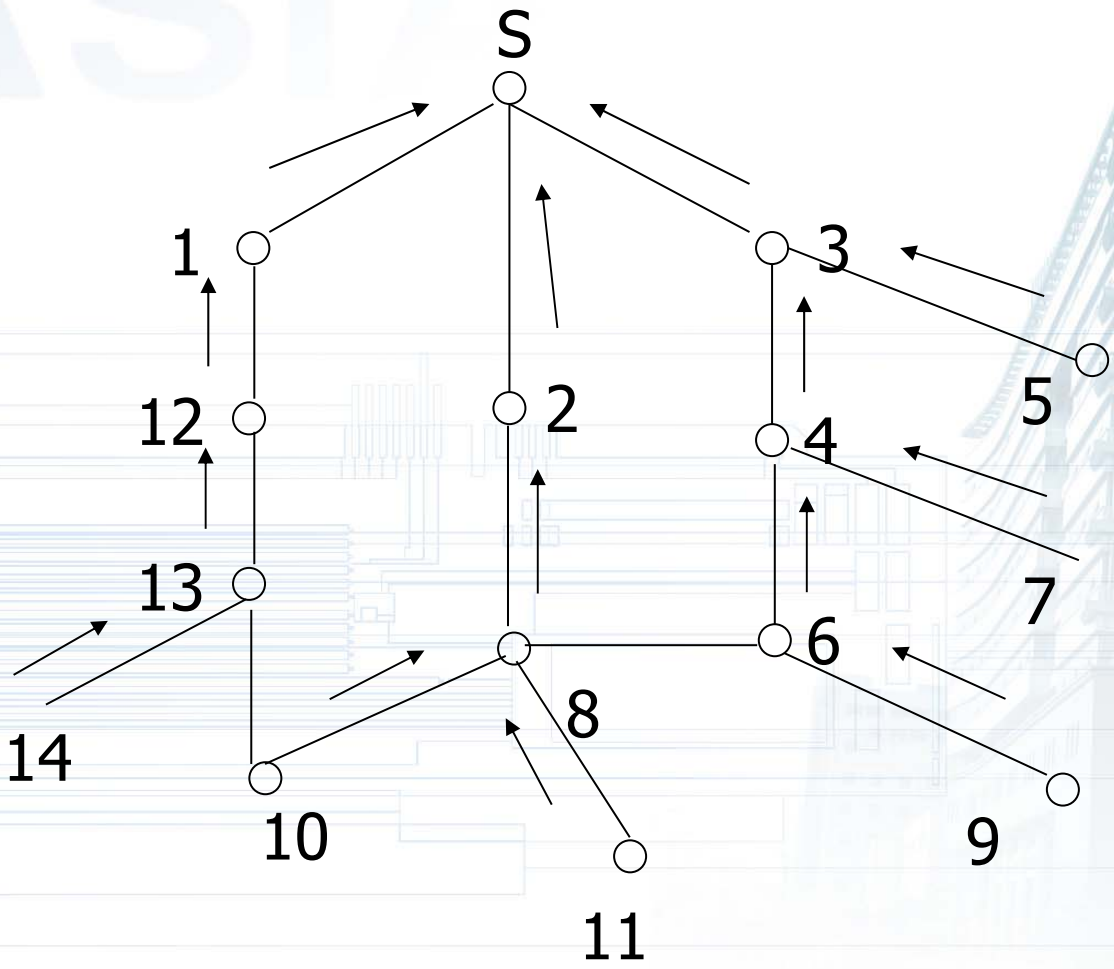
CLOSE表中的节点修改指针



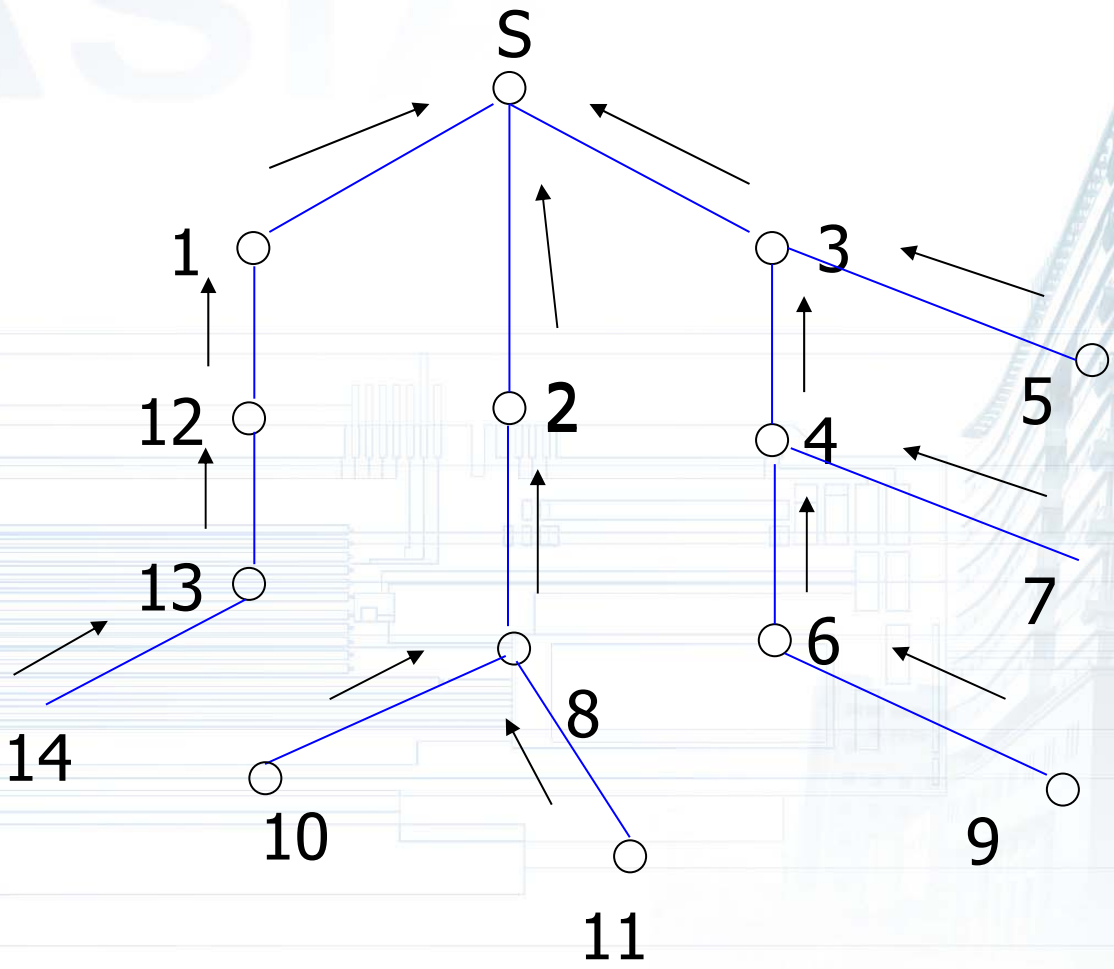
OPEN	CLOSE
{S}	{}
{1,2,3}	{S}
{3,1,2}	{S}
{4,5,1,2}	{S,3}
{6,7,5,1,2}	{S,3,4}
{8,9,7,5,1,2}	{S,3,4,6}
{10,11,9,7,5,1,2}	**{S,3,4,6,8}**
{13,10,11,9,7,5,2}	**{S,3,4,6,8,1,12}**
{13,10,11,9,7,5}	**{S,3,4,6,8,1,12,2}**

CLOSE表中的节点**(8)**的后裔(10)修改指针

搜索图的直观表示



搜索树的直观表示



讲课内容

一、问题求解

二、回溯式策略

三、图搜索策略

四、无信息搜索

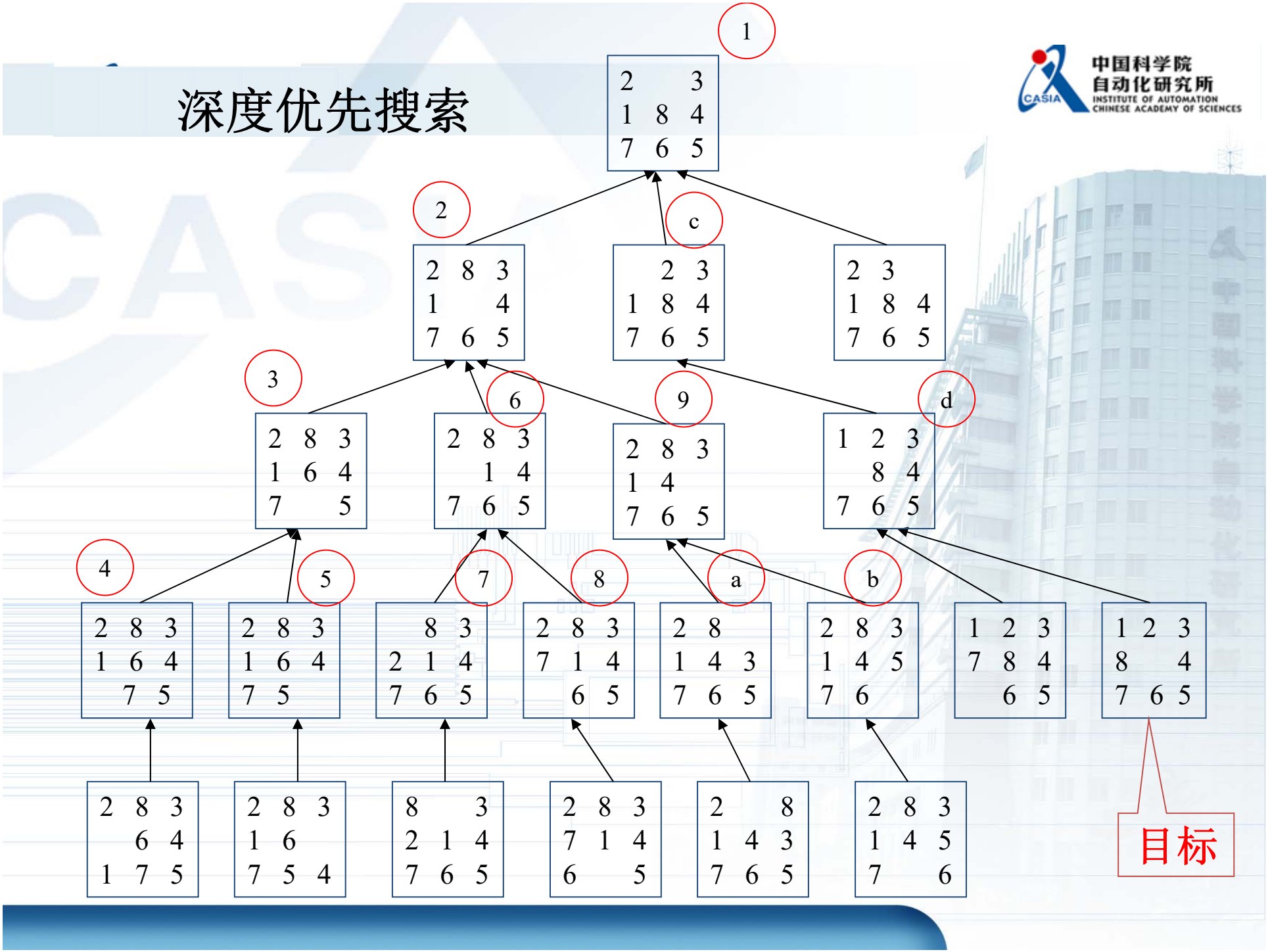
五、启发式搜索

五、A*算法与性质

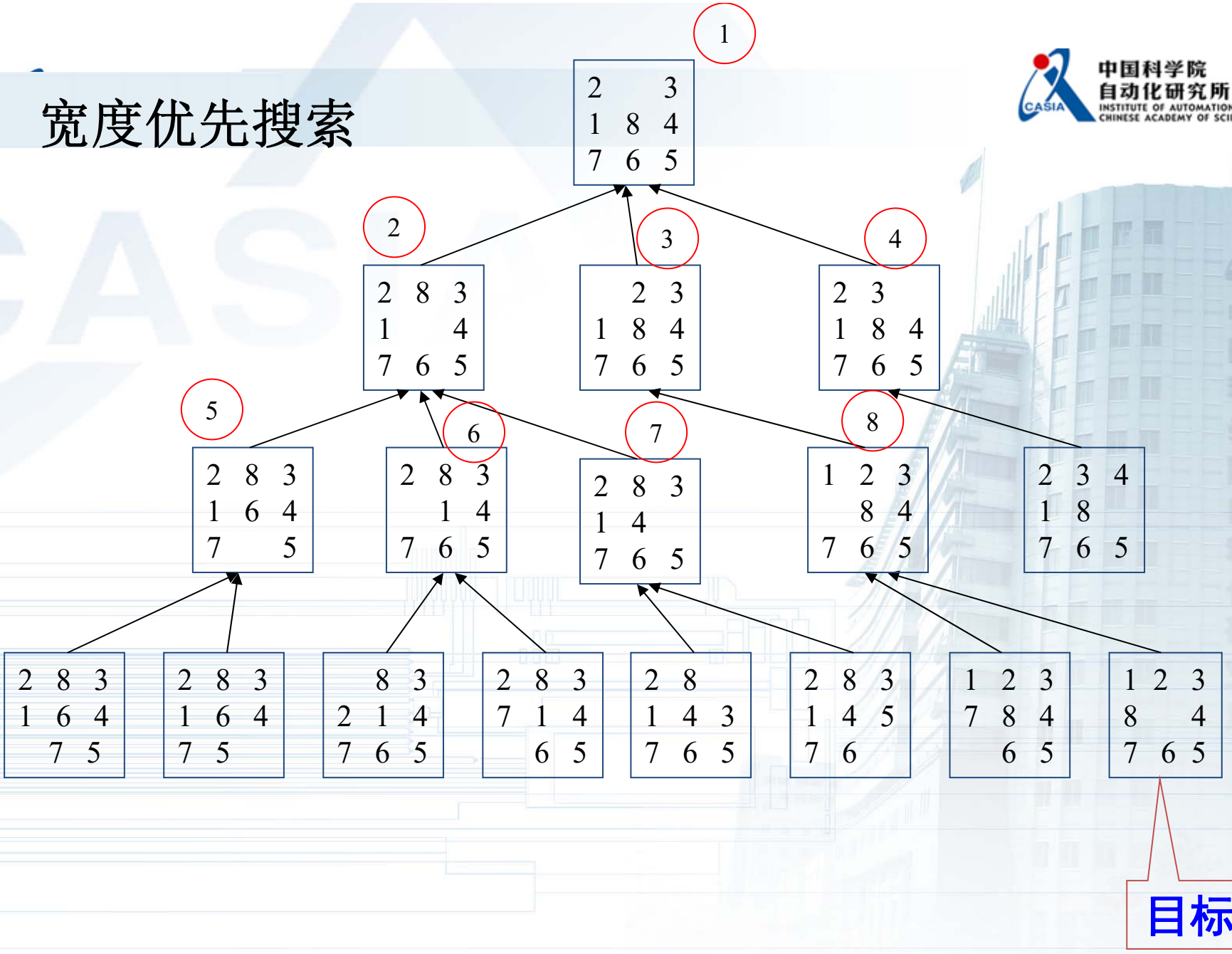
无信息图搜索

- 无信息图搜索
  - 如果在GRAPHSEARCH中，对节点的排序不使用与问题相关的信息，则称为无信息图搜索。
- 深度优先搜索
  - 不同的节点，深度越大，在OPEN表中越靠前；
  - 深度相同，任意排序；
- 宽度优先搜索
  - 不同的节点，深度越小，在OPEN表中越靠前；

深度优先搜索


1
2 3
1 8 4
7 6 5
2
2 8 3
1 4
7 6 5
c
2 3
1 8 4
7 6 5
2 3
1 8 4
7 6 5
3
2 8 3
1 6 4
7 5
6
2 8 3
1 4
7 6 5
9
2 8 3
1 4
7 6 5
d
1 2 3
8 4
7 6 5
4
2 8 3
1 6 4
7 5
5
2 8 3
1 6 4
7 5
7
8 3
2 1 4
7 6 5
8
2 8 3
7 1 4
6 5
a
2 8
1 4 3
7 6 5
b
2 8 3
1 4 5
7 6
1 2 3
7 8 4
6 5
1 2 3
8 4
7 6 5
2 8 3
6 4
1 7 5
2 8 3
1 6
7 5 4
8 3
2 1 4
7 6 5
2 8 3
7 1 4
6 5
2 8
1 4 3
7 6 5
2 8 3
1 4 5
7 6
目标

宽度优先搜索


1
2 3
1 8 4
7 6 5
2
2 8 3
1 4
7 6 5
3
2 3
1 8 4
7 6 5
4
2 3
1 8 4
7 6 5
5
2 8 3
1 6 4
7 5
6
2 8 3
1 4
7 6 5
7
2 8 3
1 4
7 6 5
8
1 2 3
8 4
7 6 5
2 3 4
1 8
7 6 5
2 8 3
1 6 4
7 5
2 8 3
1 6 4
7 5
8 3
2 1 4
7 6 5
2 8 3
7 1 4
6 5
2 8
1 4 3
7 6 5
2 8 3
1 4 5
7 6
1 2 3
7 8 4
6 5
1 2 3
8 4
7 6 5
目标

讲课内容

一、问题求解

二、回溯式策略

三、图搜索策略

四、无信息搜索

五、启发式搜索

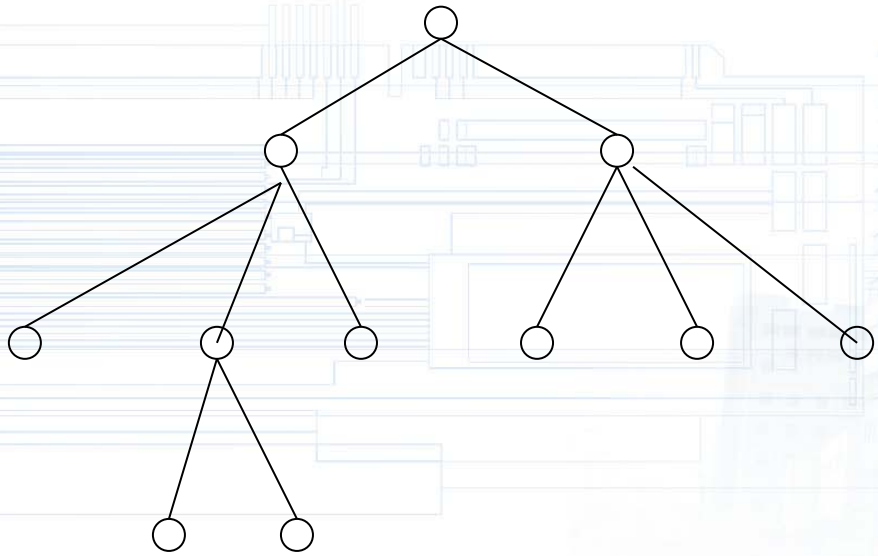
五、A*算法与性质

启发式搜索

- 无信息搜索一般需要产生大量的节点，因而效率较低
- 启发式信息
  - 为提高效率，可以使用一些问题相关的信息，以减小搜索量，这些信息就称为启发式信息
- 启发式搜索
  - 使用启发式信息指导的搜索过程称为启发式搜索
  - 启发式图搜索: 对节点排序
- 例子:
  - 国科大怀柔校区 → 国科大玉泉路校区

启发式的基本思想

- 定义一个评价函数f(n)，对当前的搜索状态进行评估，找出一个最有希望的节点来扩展。



例子: eight-puzzle

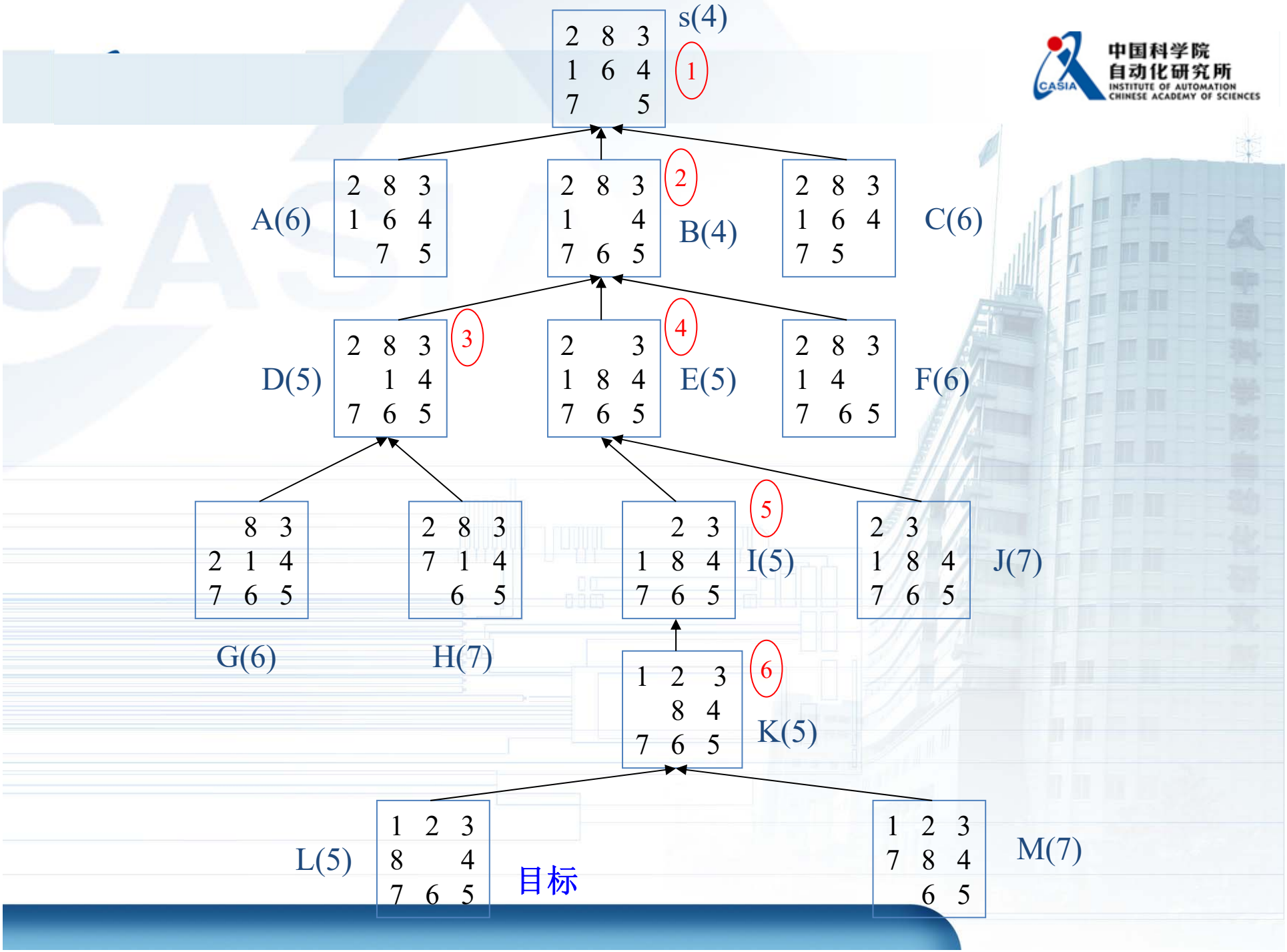
- 评价函数
  - **f(n) = d(n) + W(n)**
  - **d(n): 节点n的深度;**
  - **W(n):与目标相比, 错位的数字数目;**

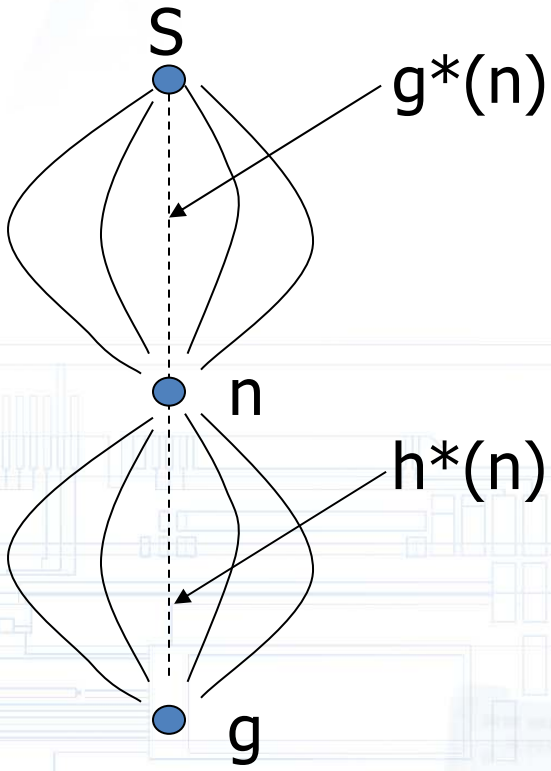
2	8	3
1	6	4
7		5

初始状态

1	2	3
8		4
7	6	5

目标状态


s(4)
1
A(6)
2
B(4)
C(6)
3
D(5)
4
E(5)
F(6)
5
I(5)
J(7)
G(6)
H(7)
6
K(5)
L(5)
目标
M(7)


S
g*(n)
n
h*(n)
g

符号约定

- g*(n)：从s到n的最短路径的耗散值
- h*(n)：从n到g的最短路径的耗散值

- f*(n)=g*(n)+h*(n)：从s经过n到g的最短路径的耗散值

- g(n)、h(n)、f(n)分别是g*(n)、h*(n)、f*(n)的估计值

- h(n): 启发函数

启发式搜索：A算法

- f(n) = g(n) + h(n)
- g(n)取实际走过的路径的费用和
- 每一条弧上的费用大于一个小正数ε
- 节点排序是按照f(n)从小到大排

讲课内容

一、问题求解

二、回溯式策略

三、图搜索策略

四、无信息搜索

五、启发式搜索

五、A*算法与性质

启发式搜索：A*算法

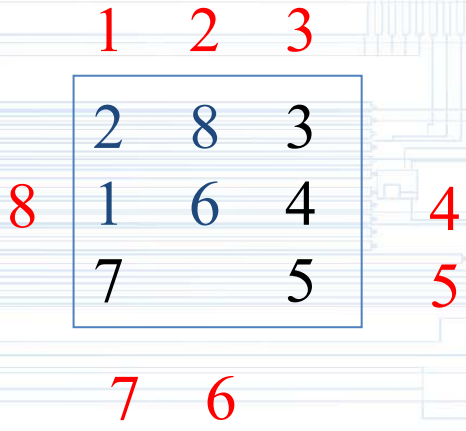
- 在A算法中，如果满足条件：

  $0 \le h(n) \le h^*(n)$

  则A算法称为A*算法。

A*算法的说明

- 8数码问题
  - h(n) = “不在位”的将牌个数
  - h(n) = 将牌“不在位”的距离和



将牌1：1
将牌2：1
将牌6：1
将牌8：2

可采纳性

- 可采纳性：
  - 对任一个图，存在从S到目标的路径，如果一个搜索算法总是结束在一条从S到目标的最佳路径上，则称此算法是可采纳的。
  - 一条: 多条

A*算法的基本性质

- 性质一：
  - GRAPHSEARCH算法对有限图终止。
    - 算法当OPEN表为空时结束。
    - 图中的任何一个节点都只进入OPEN表一次，并且图中节点是有限的，所以OPEN表一定会被取空，导致算法结束。

- 性质二：
  - 在A*算法结束前的任意时刻，OPEN表中都至少有一个节点n,节点n在从初始节点S到目标的最佳路径上, 并且A*算法已经发现了这条路径, 并且节点n满足$f(n) \leq f^*(S)$。

- **性质三：**
  - 如果存在从S到目标节点的路径，则A*扩展的节点n，一定满足：

$$f(n) \le f^*(S)$$

- **性质四：**
  - 如果存在从S到目标节点的路径，则A*对无限图结束。
- **性质五：**
  - A_1与A_2是具有不同启发函数的A*算法，假设$h_1 > h_2$，则A_1具有更多的启发式信息，由A_1扩展的节点必然被A_2扩展。
- **性质六：**
  - A*是可采纳的。

本次课程作业

- 教科书 P99—101
  - 3.2
  - 3.9

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

非常感谢同学们的光临

欢迎交流提问