人工智能

Artificial Intelligence

中南大学 计算机学院

课程内容

1	人工智能概论	学科定义、起源与发展历程、 人工智能与人类智能、应用场景与挑战
2	**知识表示方法**	**状态空间表示、问题归约表示 谓词逻辑表示、语义网络表示**
3	确定性推理技术	图搜索技术、消解原理 规则演绎、产生式系统
4	计算智能	神经计算、模糊计算、遗传算法、 粒群优化、蚁群优化
5	机器学习	神经学习和决策树学习

知识表示方法 Methodologies of Knowledge Representation

◆ 从字面上理解

知识表示 = **知识 (Knowledge; 对世界的描述)**

\+ **表示 (Representation;知识的编码方式)**

- **知识的表示**就是对知识的一种描述，或者说是对知识的一组约定，一种计算机可以接受的用于描述知识的数据结构。【百度】
- 为了能让智能系统理解、处理知识，并完成基于知识的任务，首先得对知识进行构建模型，即**知识表示**。【教材】

知识表示方法

◆什么是知识?

- 数据、信息≠知识
- 知识层次

- DATA：信息的载体和表示。用一组符号及其组合表示信息。
- INFORMATION：数据的语义。数据在特定场合下的具体含义。
- KNOWLEDGE：信息关联后形成的信息结构。事实&规则经过加工、整理后的信息.

Q：所有的知识都是正确的?

知识表示方法

◆什么是知识?

数据	一些无关联的事实(fact)	例：下雨了
信息	建立了事实间的联系后形成信息，提供对what, who, when,where 等问题的回答	例：温度下降了15度后就下雨了
知识	当能建立模式之间的联系后便涌现了知识，提供对how的回答	例：如果湿度非常大并且温度下降显著，大气无法保留湿气便下雨了
智慧	能描述模式之间关系的规律，提供对why的回答	例：理解下雨、蒸发、空气状况、温度等级及它们的变化之间的相互作用

知识表示方法

◆ 知识的属性

- 真伪性
- **相对性**
- 不完全性
- **不确定性**
- 可表示性
- 可存储性、可传递性和可处理性
- 相容性

知识表示方法

不同类型的**知识**需要**不同**的表示方式、**不同**的表示方法需要**不同**的求解技术

◆ 知识表示是问题求解的基础

- 问题求解是人工智能的核心问题之一
- 问题求解的目的
  - 机器自动找出某问题的正确解决策略
  - 更进一步，能够举一反三，具有解决同类问题的能力

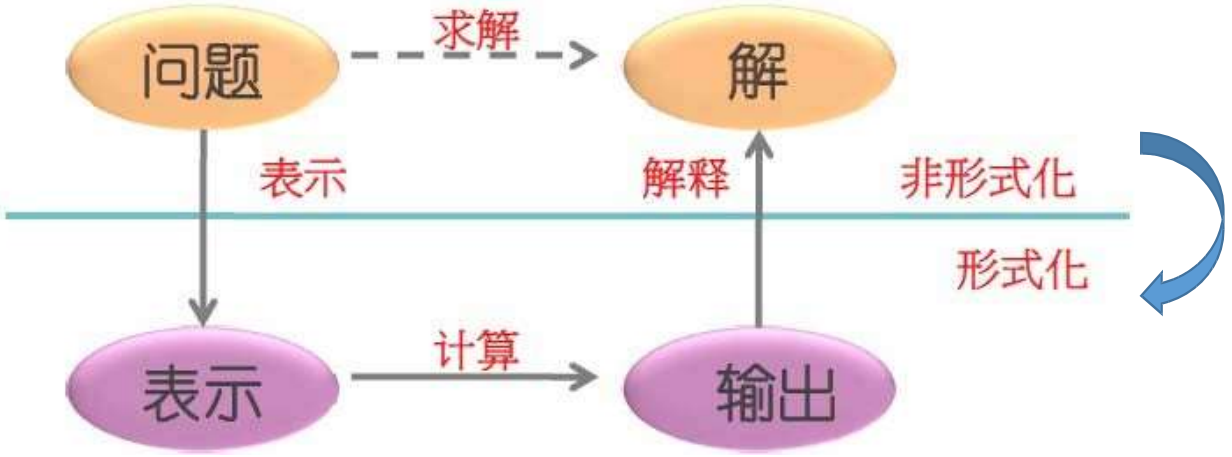
◆ 智能系统中

- **知识**是对世界的描述（决定系统的能力）
- **表示**是知识的编码方式（决定系统的性能）

知识表示方法

◆ 知识表示的过程

- 非形式化的自然语言描述 → 形式化的易于被计算机理解



知识表示方法

状态空间表示

问题归约表示

谓词逻辑表示

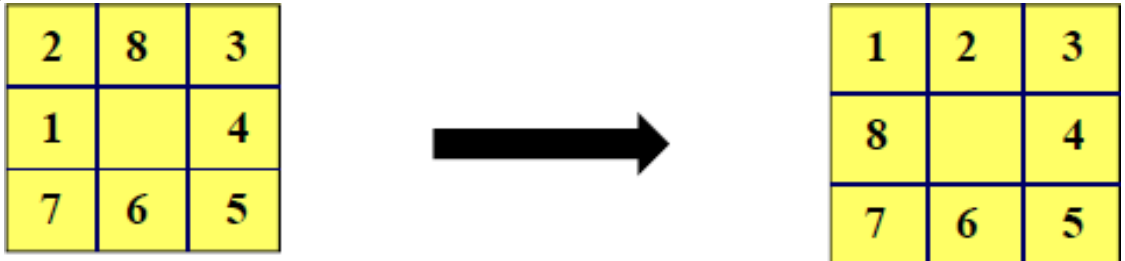
语义网络表示

- 非结构化
- 结构化

状态空间知识表示（State Space Representation）

◆ 例：八数码难题

- 在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘上还有一个空格，与空格相邻的棋子可以移到空 格中。



初始状态　　目标状态

- 如何将棋盘从某一初始状态变成最后的目标状态？

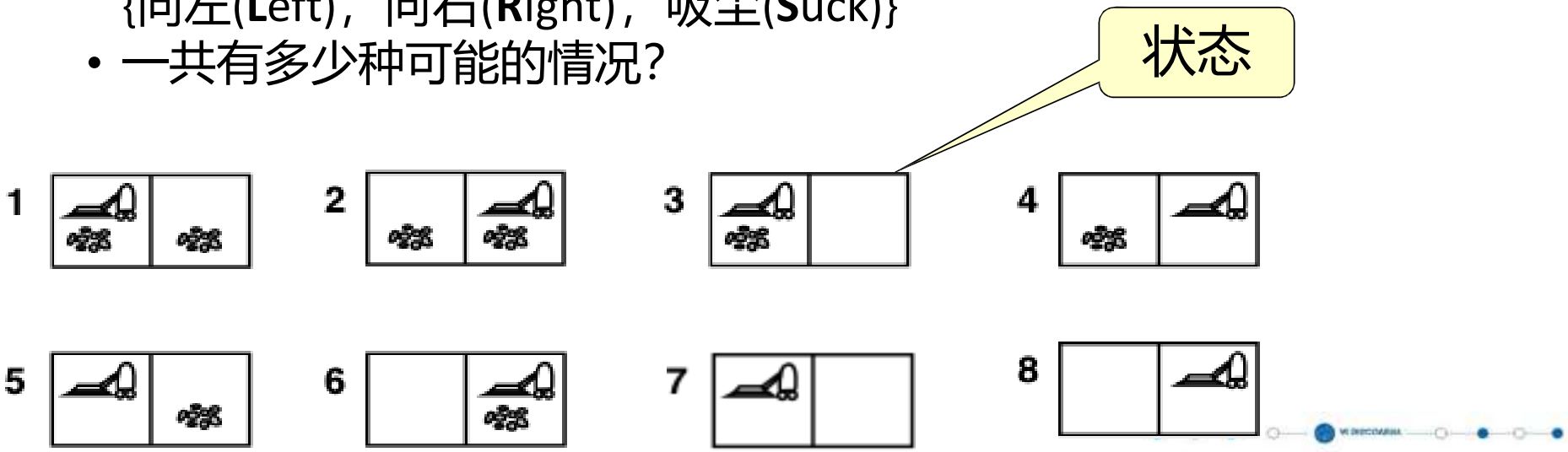
状态空间知识表示

◆ 例：怎样找到两点之间的最短路径呢?

问题有了，可怎么让计算机知道这些问题呢?

问题表示——状态空间图

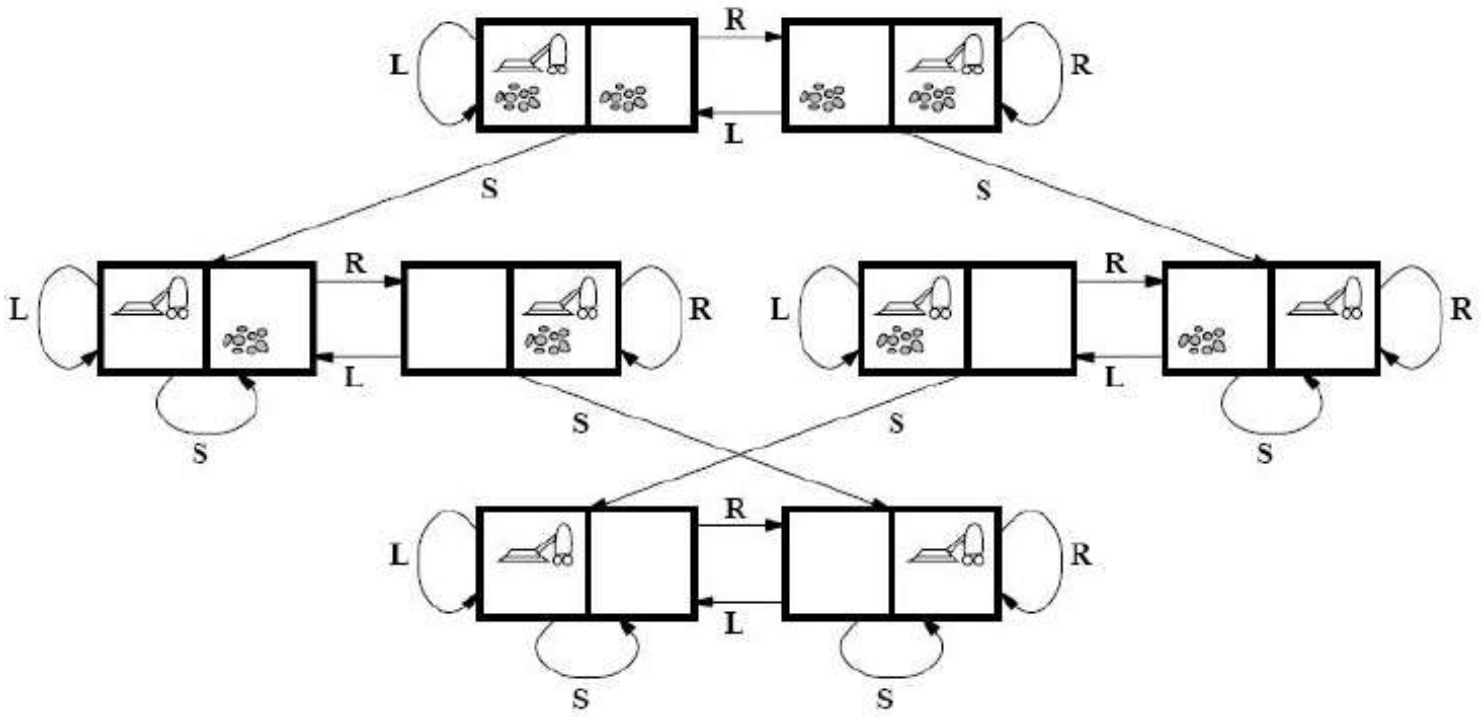
- 例：**吸尘器的世界**
  - 假设：吸尘器的世界只有两块地毯大小，地毯或者是脏的，或者是干净的
  - 吸尘器能做的动作只有三个
    {向左(**L**eft)，向右(**R**ight)，吸尘(**S**uck)}
  - 一共有多少种可能的情况?



◆例：真空吸尘器的世界

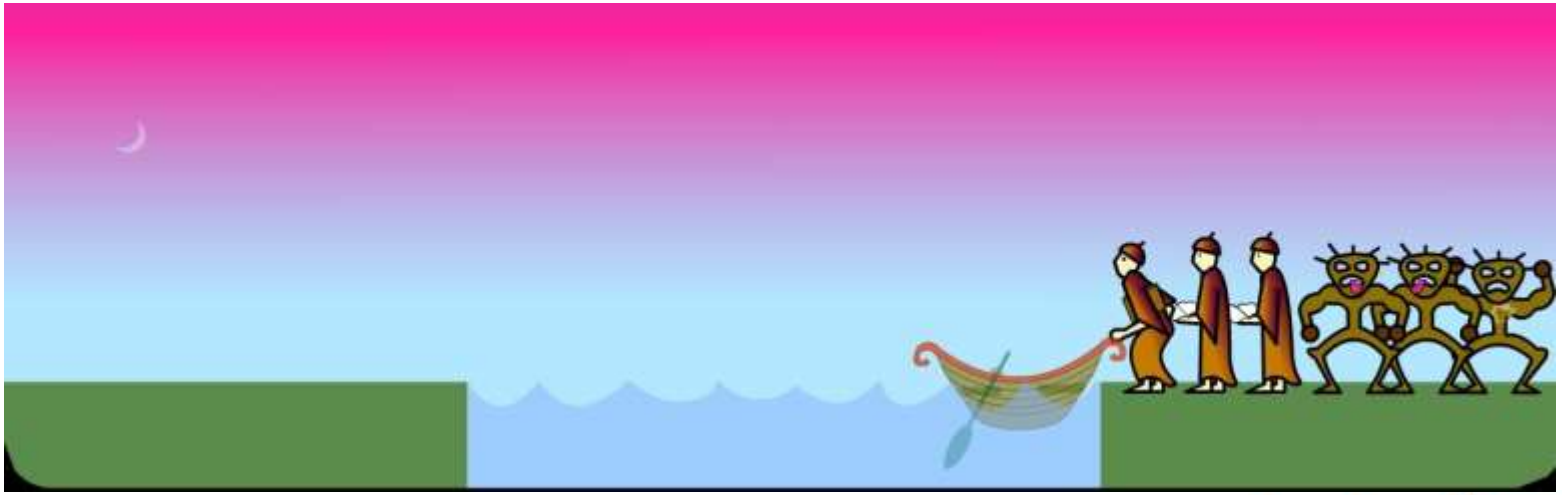
- 状态之间可以互相转换

状态空间图



- **例：传教士野人问题**（Missionaries& Cannibals，MC问题）

有三个传教士M和三个野人C过河，只有一条能装下两个人的船，在河的一方或者船上，**如果野人的人数大于传教士的人数**，那么传教士就会有危险，你能不能提出一种安全的渡河方法呢？



状态及其表示

- **状态**(State)：问题在某一时刻所处的“位置”，“情况”等.
  - 根据问题所关心的因素，一般用**向量**形式表示，每一位表示一个因素，状态可有多种表示方法：

(左岸传教士数, 右岸传教士数, 左岸野人数, 右岸野人数, 船的位置)

或

(左岸传教士数, 左岸野人数, 船的位置)

初始状态：(0, 0, 0)

目标状态：(3, 3, 1)

0：右岸

1：左岸

哪些操作能导致状态变化？

状态的转换

- **算子（Operator, 算符，操作符）**——使状态发生改变的操作
- MC问题中的算子
  - 将传教士或野人运到河对岸
  - Move-1m1c-lr：将一个传教士(m)一个野人(c)从左岸(l)运到右岸(r)
  - **所有可能操作**

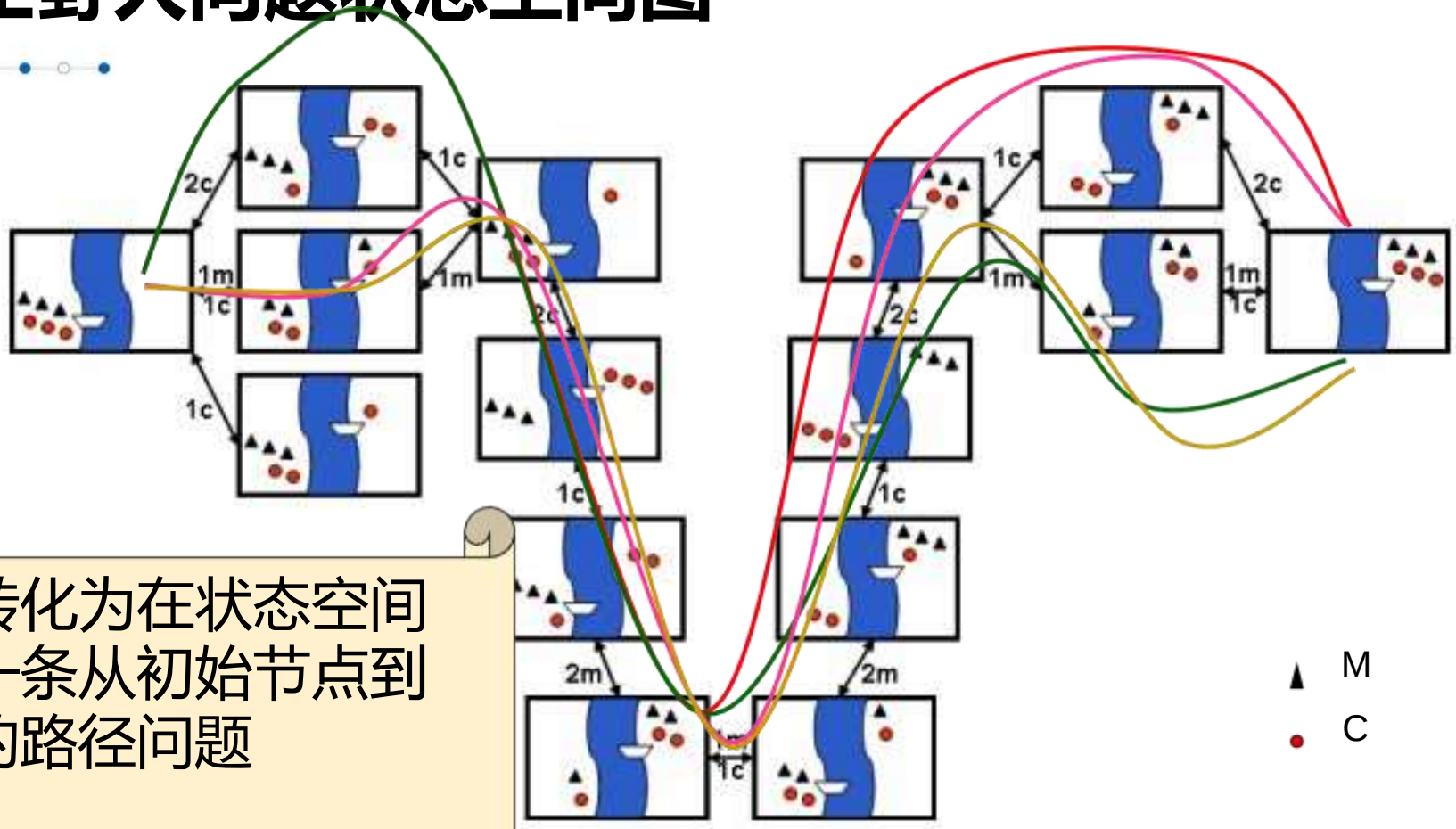
Move-1m1c-lr　Move-1m1c-rl　Move-2c-lr

Move-2c-rl　Move-2m-lr　Move-2m-rl

Move-1c-lr　Move-1c-rl　Move-1m-lr

Move-1m-rl

传教士野人问题状态空间图



求解过程转化为在状态空间图中搜索一条从初始节点到目标节点的路径问题

2.1 State Space Representation
状态空间表示

2.1.1 问题状态描述

- 状态空间法（State Space Method）:一种基于解答空间的问题表示和求解方法，它是以状态（**State**）和操作符（**Operator**）为基础的.
  - **状态** (State): 描述某类不同事物间的差别而引入的一组最少变量q_0，q_1，…，q_n的有序集合，其矢量形式如下：

$$Q = [q_0，\ q_1，\ \ldots，\ q_n]^T$$

状态变量

  - **算符 或 操作符** (Operator): 使问题从一种状态变化为另外一种状态的手段.

状态空间知识表示

- **问题的状态空间** (State Space)
  - 一个表示该问题全部可能状态及其关系的图，它包含三种说明的集合，即所有可能的问题初始状态集合**S**、操作符集合**F**以及目标状态集合**G**.

{*S*,*F*,*G*} - ***S***: set of all possible initial states of problem

F: set of operators

G: set of goal states

8 Puzzle Problem (8数码难题)

2	8	3
1	6	4
7		5

Original State

如何把初始棋局变换为目标棋局呢?

1	2	3
8		4
7	6	5

Goal State

问题的解答:
某个合适的棋子**走步序列**

Move Blank square Left

2	8	3
1	6	4
	7	5

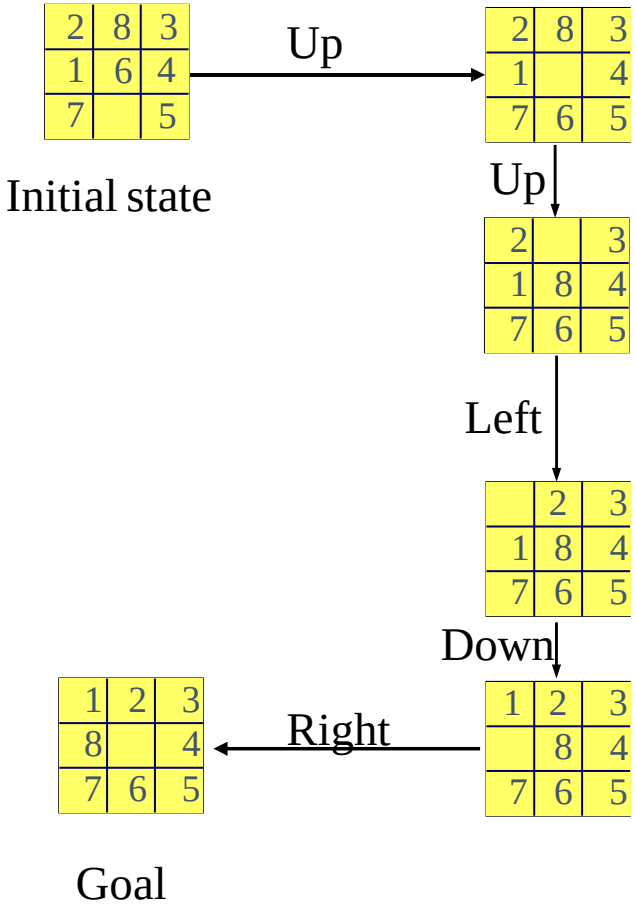
2	8	3
1		4
7	6	5

Move Blank square Up

2	8	3
1	6	4
7	5	

Move Blank square Right

8 Puzzle Problem



Solution:
A transition/operator sequence
1: Move Blank square Up
2: Move Blank square Up
3: Move Blank square Left
4: Move Blank square Down
5: Move Blank square Right

Is it optimal?
How to find it? 某种试探搜索

把初始状态可达到的各状态所组成的空间设想为一幅由各种状态对应的节点组成的图。这种图称为**状态图**

15 Puzzle Problem (15数码难题)

11	9	4	15
1	3		12
7	5	8	6
13	2	10	14

(Initial State)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(Goal State)

- 完成某个问题的状态描述，必须确认三件事：
  - ① 该状态描述方式，特别是初始状态描述
  - ② 操作符集合及其对状态描述的作用
  - ③ 目标状态描述的特性

15 Puzzle Problem

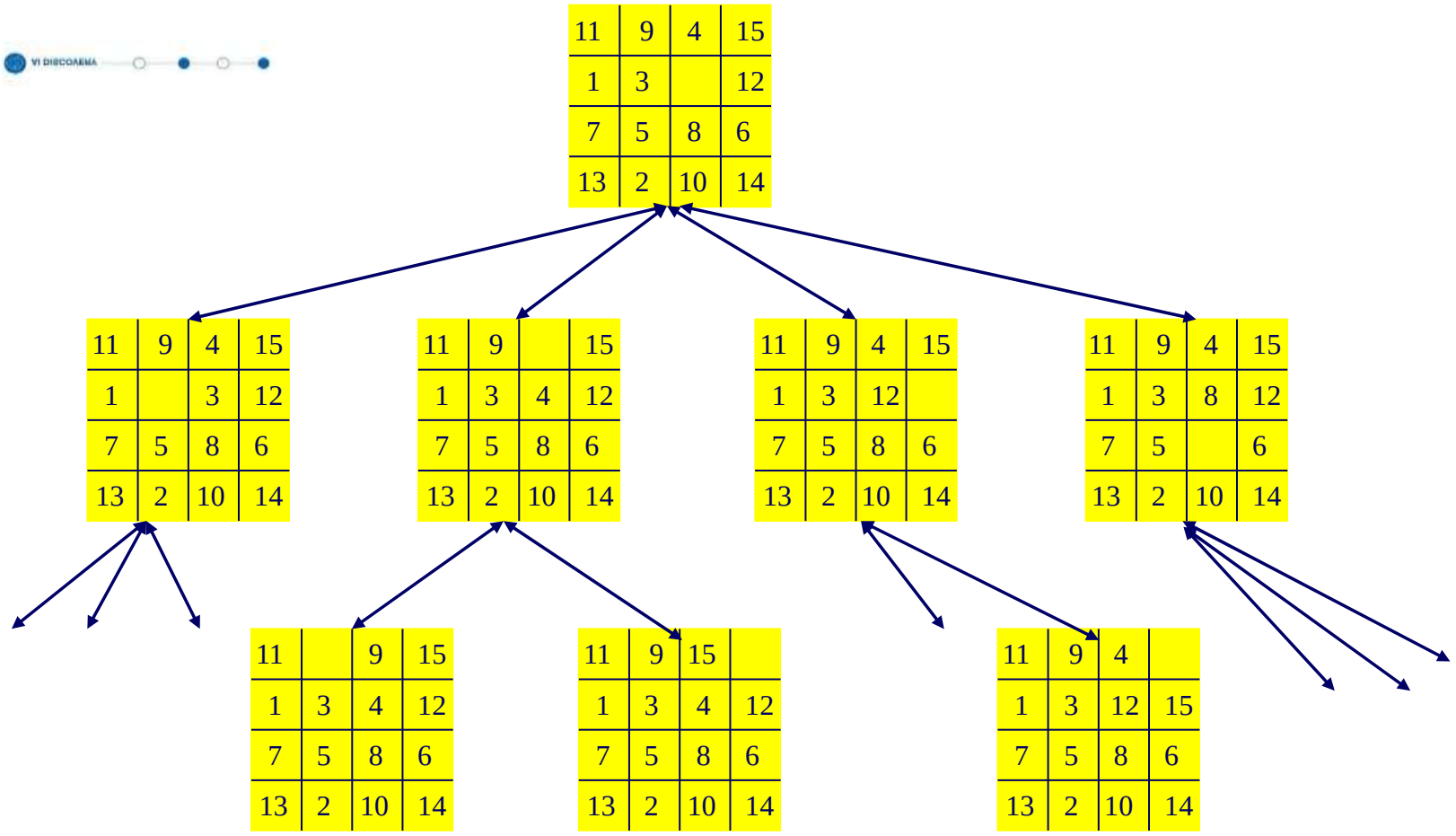
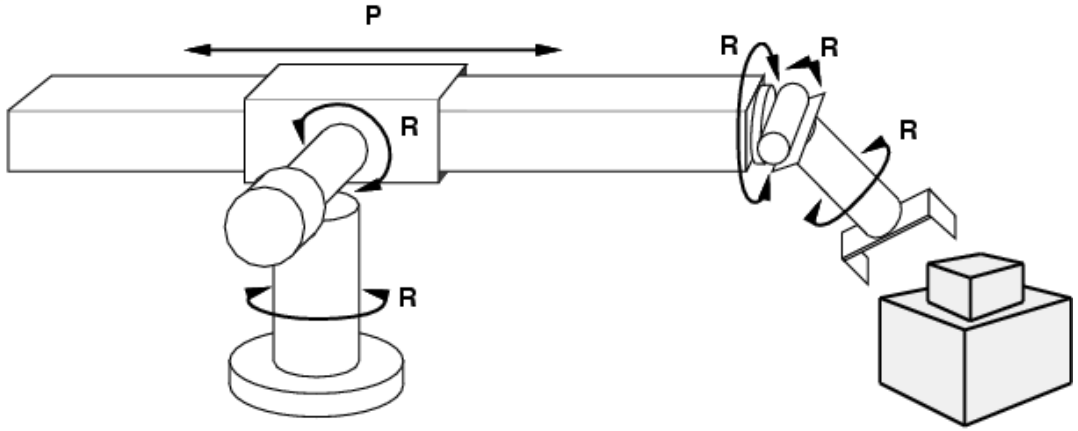


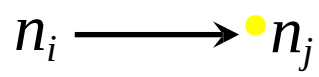
图2.2 十五数码难题部分状态空间图

Example: robotic assembly



- states?: real-valued coordinates of robot joint angles parts of the object to be assembled
- actions?: continuous motions of robot joints
- goal test?: complete assembly
- path cost?: time to execute

2.1.2 状态图示法 Graph Notion of State

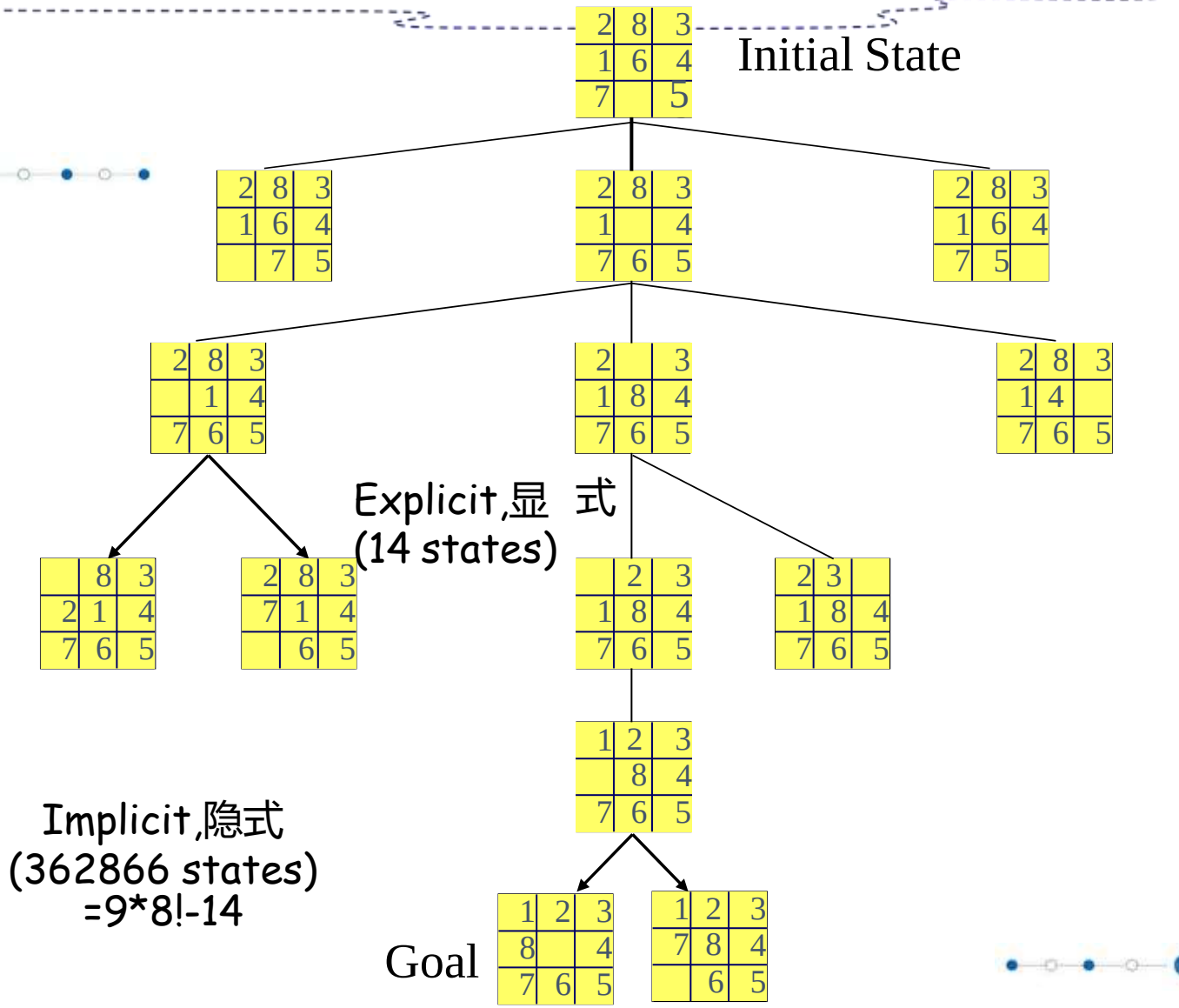
- **图的基本概念**
  - 有向图(directed graph)= Nodes + Arcs 
  - 某个节点序列(n_1， n_2， …， n_i， … n_k)

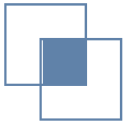
  如果对于每一个n_i指向n_j , 那么n_j 就叫做n_i ;的后继节点（后裔）

  n_j : A successor (后继、后裔) of n_i ;

  n_i : An ancestor (前驱、祖先) of n_j

  - $C(n_j, n_i)$: 表示从n_j 指向节点n_i的那段弧线的代价。
- 显示图Explicit Graph
- 隐式图Implicit Graph


Initial State
Explicit,显 式
(14 states)
Implicit,隐式
(362866 states)
=9*8!-14
Goal



状态图示法（Graph Notion of State）

- 把后继算符应用与节点的过程，就是扩展一个节点的过程
  - 搜索某个状态空间以求得算符序列的一个解答过程，就对应于使**隐式图**足够大一部分变为**显式**以便包含目标节点的过程。
- 图和状态描述的相似性

Graph Notation of State	node	arc	path	optimization	solution
State Description of Problem	state	operator	cost	minimum cost	sequence of operator

Example 1 – Route Planning
例1 - 路线规划

- 初始状态 Initial state
  - City the journey starts in Changsha
- 操作符Operators
  - Driving from city to city
- 目标状态Goal state
  - Destination city is Beijing

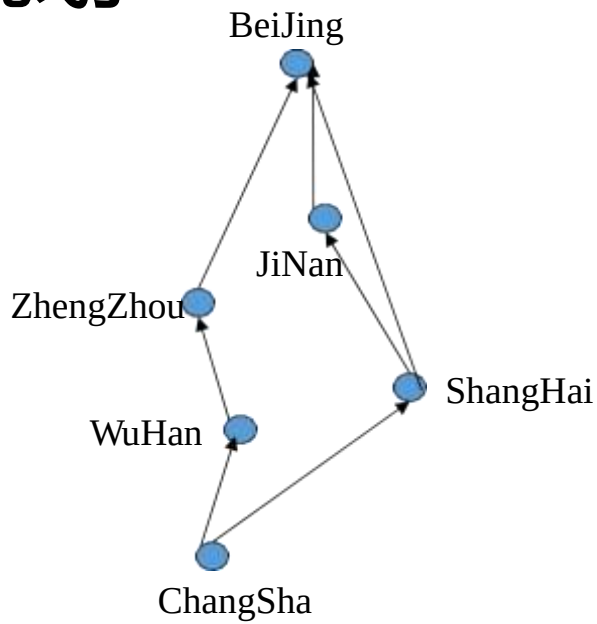


Fig.5 Route Planning

Example 2 – Monkey & Banana
例2 - 猴子与香蕉问题

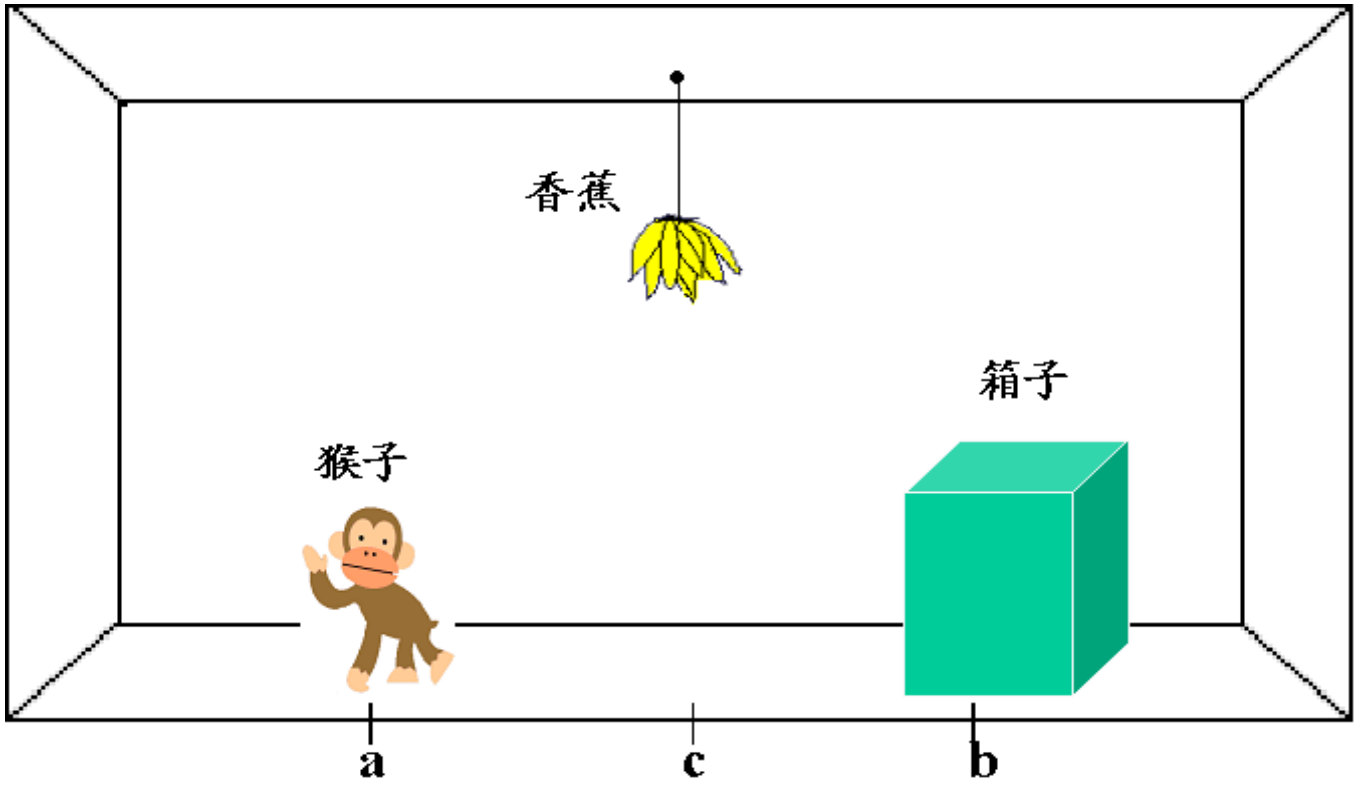


Fig.6 Monkey and Banana problem

Example 2 – Monkey & Banana

- 用一个四元表列**(W,x,Y,z)**来表示这个问题的状态：
  - W 猴子的水平位置
  - X 当猴子在箱子顶上时取x=1；否则取x=0
  - Y 箱子的水平位置
  - z 当猴子摘到香蕉时取z=1；否则取z=0
- 这个问题中的操作(算符):
  - **goto(U)** 表示猴子走到水平位置U

    (**W**，0，Y，z) —— goto(U) ——→ (**U**，0，Y，z)
  - **pushbox(V)** 表示猴子把箱子推到水平位置V:

    (**W**，0，**W**，z) —— pushbox(V) ——→ (**V**，0，**V**，z)

Example 2 – Monkey & Banana

- **climbbox**: 猴子爬上箱顶

  （W，**0**，W，z） —climbbox→ （W，**1**，W，z）

  在应用算符climbbox时也必须注意到，猴子和箱子应当在同一位置上，而且猴子不在箱顶上 。

- **grasp**: 猴子摘到香蕉

  （c，1，c，**0**） grasp （c，1，c，**1**）

  其中，c是香蕉正下方的地板位置，在应用算符grasp时，要求猴子和箱子都在位置c上，并且猴子已在箱子顶上。

Example 2 – Monkey & Banana

- 令初始状态为(a，0，b，0)
- 这时，goto(U)是唯一适用的操作，并导致下一状态(U，0，b，0)
- 现在有3个适用的操作，即**goto(U)**，**pushbox(V)**和**climbbox**(若U=b)
- 从状态空间图可看出，该初始状态变换为目标状态的操作序列为：

  {goto(b), pushbox(c), climbbox, grasp}
- 一种求解方案.

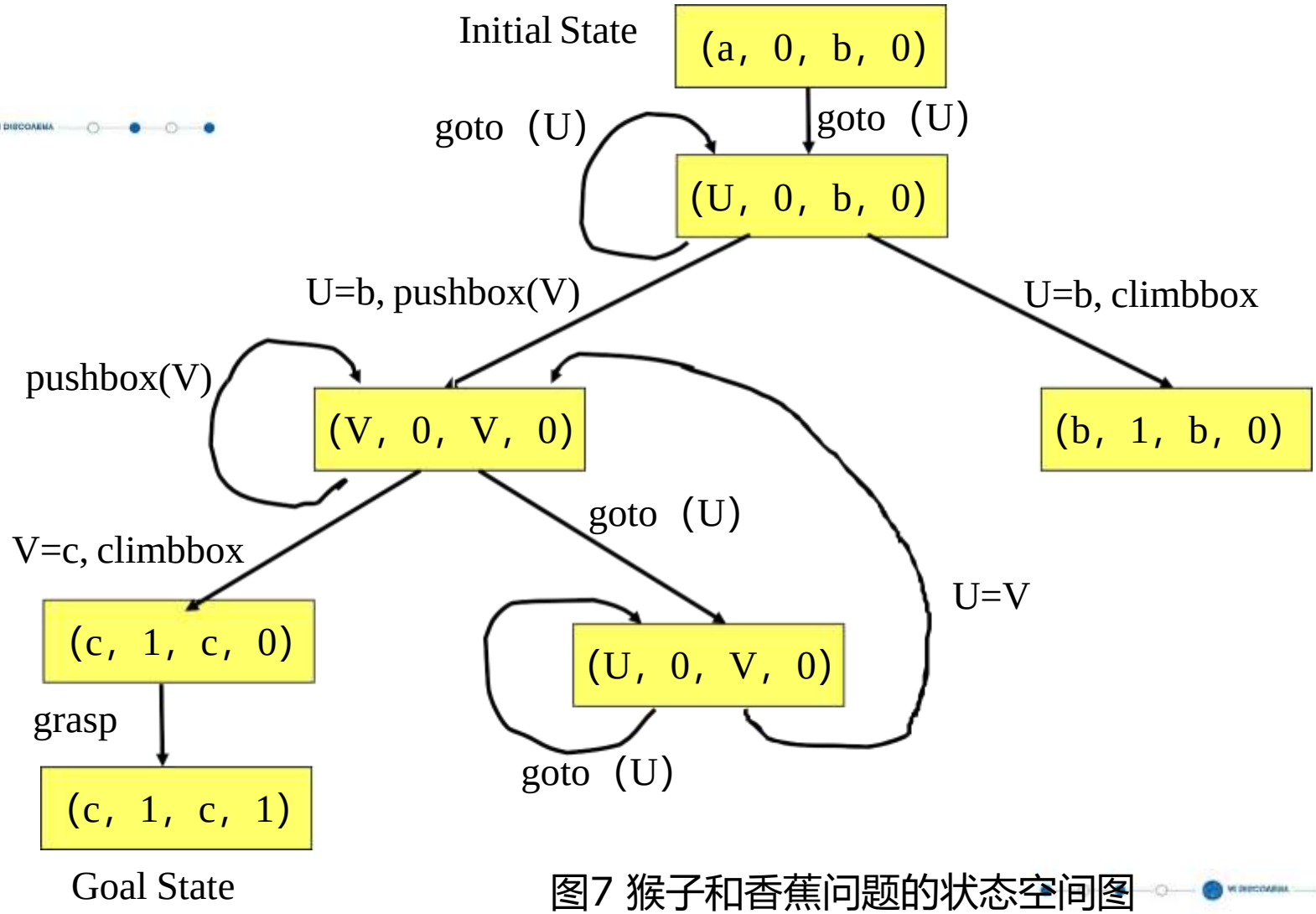
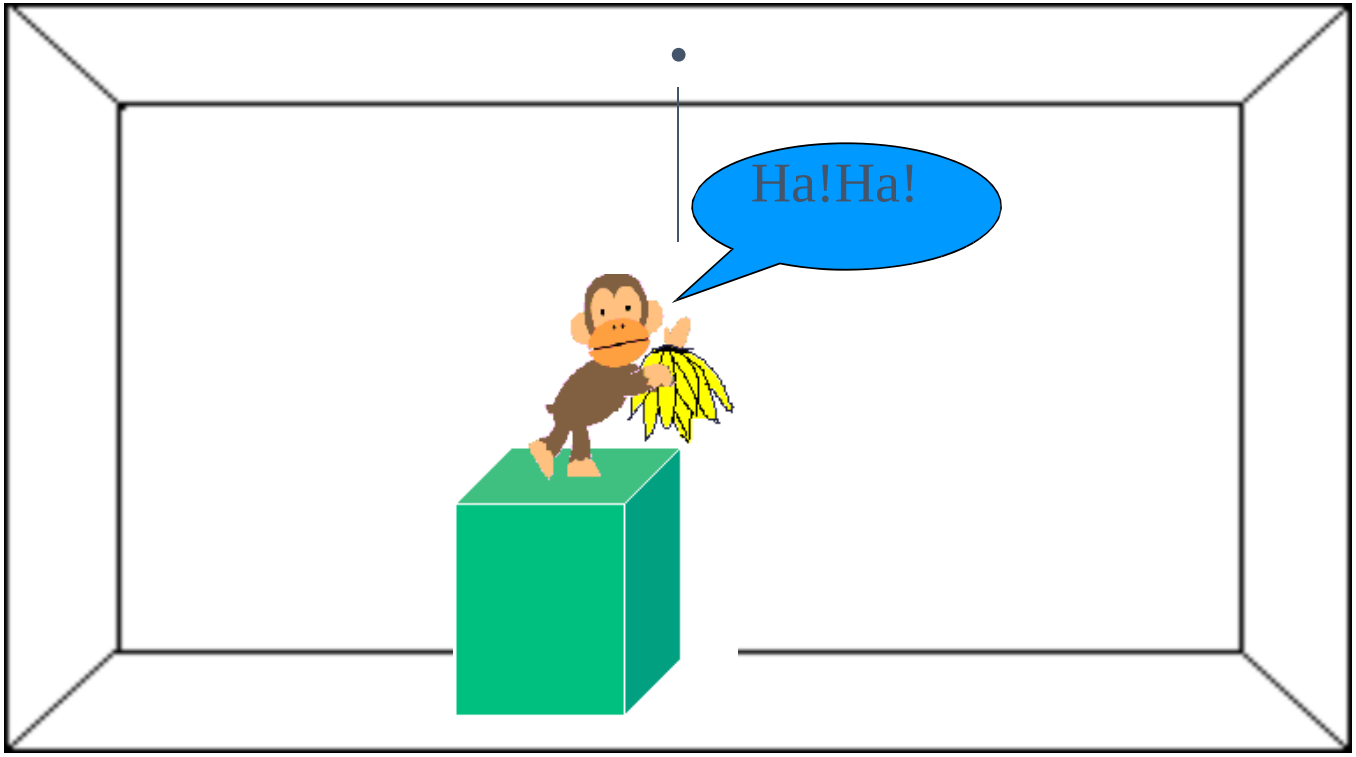


图7 猴子和香蕉问题的状态空间图

Example 2 – Monkey & Banana

The solving process（动画）：



Example 3 – 取名字

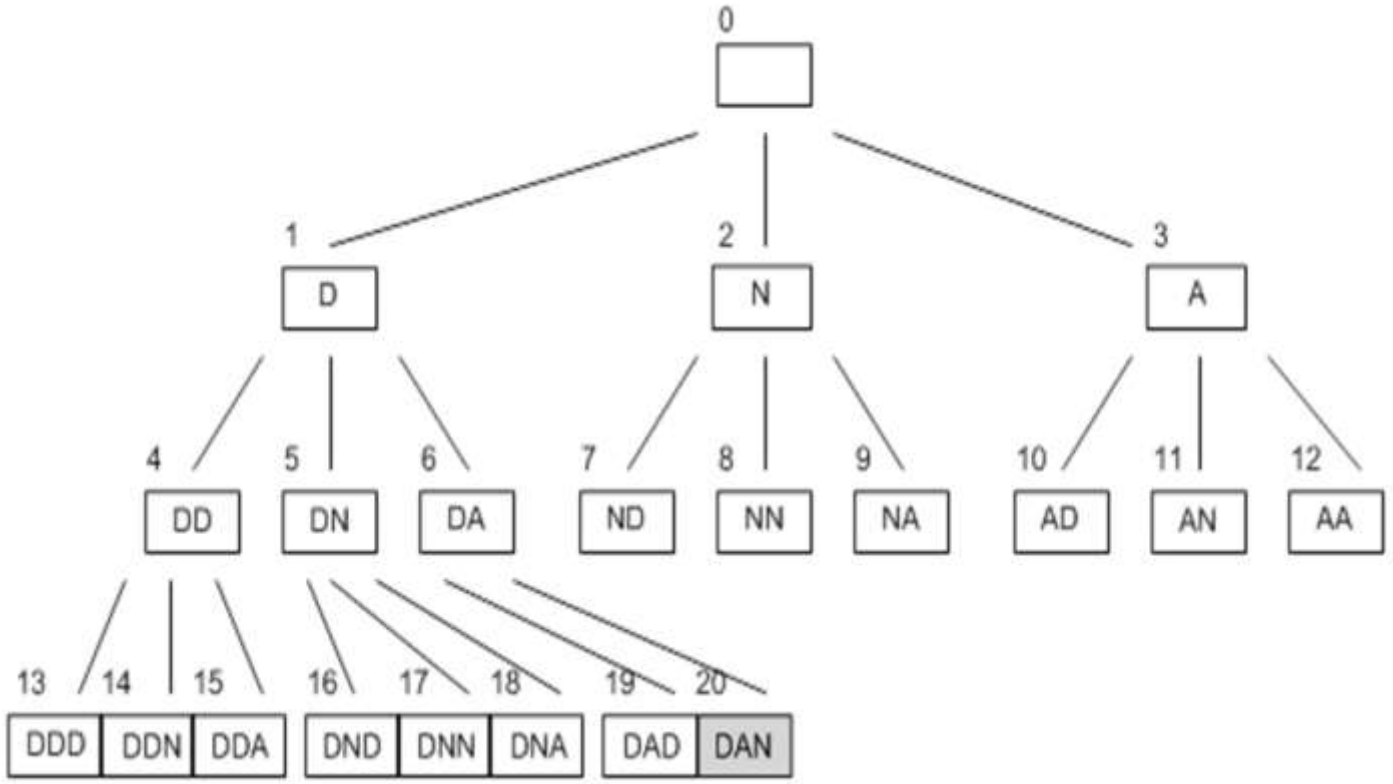
- 一位研究基因的教授准备给他的新生宝宝取名字，希望只用字母D,N & A
- Search by writing down possibilities (states)
  - D,DN,DNNA,NA,AND,DNAN, etc.
  - Operators: add letters on to the end of already known states
    (i) add a 'D' to an existing string
    (ii) add an 'N' to an existing string and
    (iii) add an 'A' to an existing string
  - Initial state is an empty string
- Goal test
  - Look up state in a book of boys names
  - Good solution: DAN

Example 3 – A name of a boy

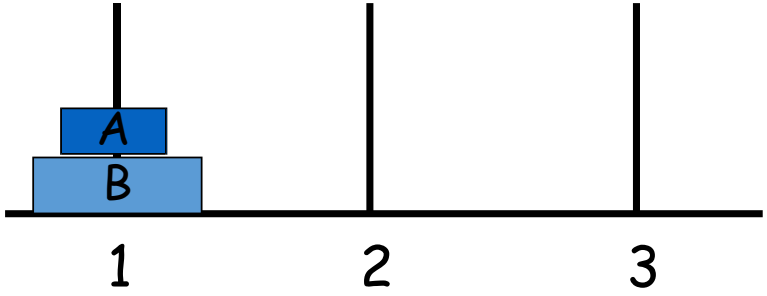
- 问题：
  - 有多少种3个字符串或者更少字母的组合？
  - 能否保证给小男孩用字母D、N和A 找到合适的名字？

  - Answer
    - For strings of length 3, there are three choices for the first letter (D, N and A), three choices for the second and three choices for the third. Hence, there are 3*3*3 = **27 possible three-letter** names with D, N and A in. Similarly, there are 3*3 = **9 possible two-letter names** and **3 possible one letter names**. Adding all the possibilities up, we get: 27 + 9 + 3 = 39 possible names.
    - Yes, all names will eventually **be enumerated** .

Example 3 – A name of a boy



More examples——Hanoi-Tower梵塔难题



- 二阶Hanoi-Tower

  设用Sk={Sk0, Sk1}表示问题的状态，其中，Sk0表示盘A所在的柱子号，Sk1表示盘片B所在的柱子号

1. 共有多少可能状态？分别是什么？
2. 若操作分别用A(i, j)和B(i, j)表示，共有多少种操作，分别怎么表示？
3. 画出该问题的状态空间图。

Answer

1　　共9种状态，分别是：

S_0=(1, 1), S_1=(1, 2), S_2=(1, 3)

S_3=(2, 1),S_4=(2, 2), S_5=(2, 3)

S_6=(3, 1), S_7=(3, 2), S_8=(3, 3)

2　　共12个算子，分别是：

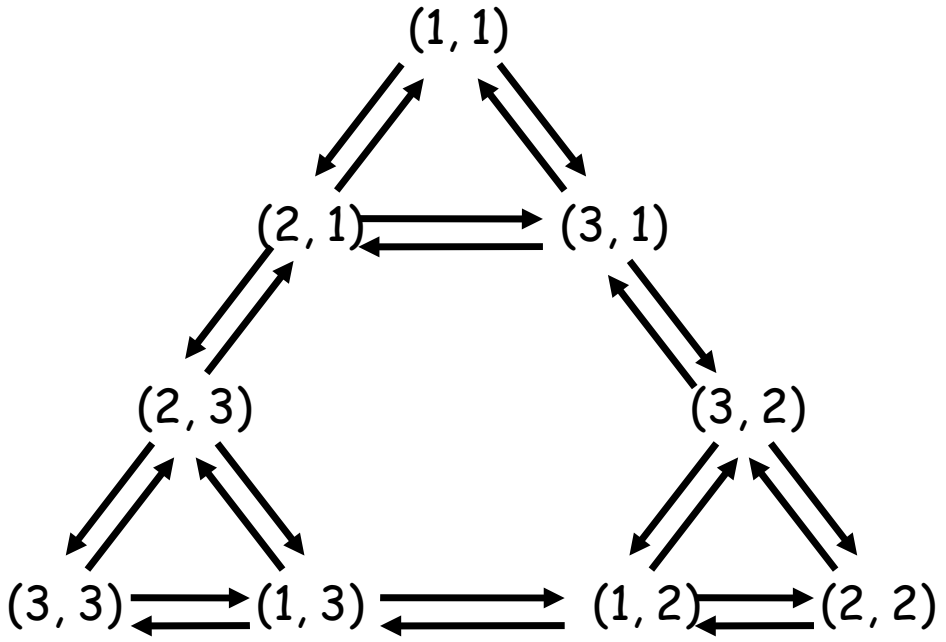
A(1, 2), A(1, 3), A(2, 1),

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

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

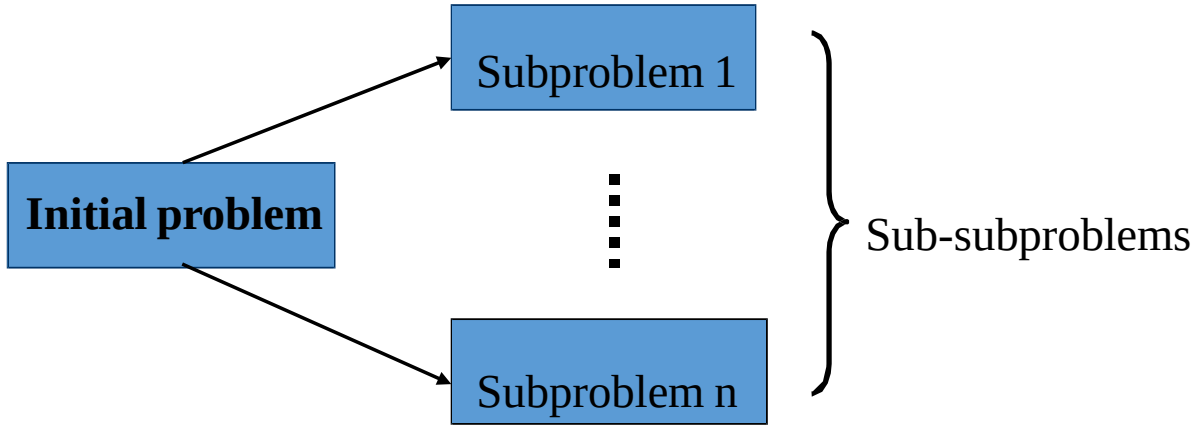
B(2, 3), B(3, 1), B(3, 2)

3　　状态空间图



2.2 Problem Reduction(问题归约法)

- 问题归约描述
  - 问题归约：另一种基于状态空间的问题描述与求解方法。
  - 先把问题分解为子问题和子-子问题，然后解决较小的问题。对该问题的某个具体子集的解答就意味着对原始问题的一个解答。



问题归约

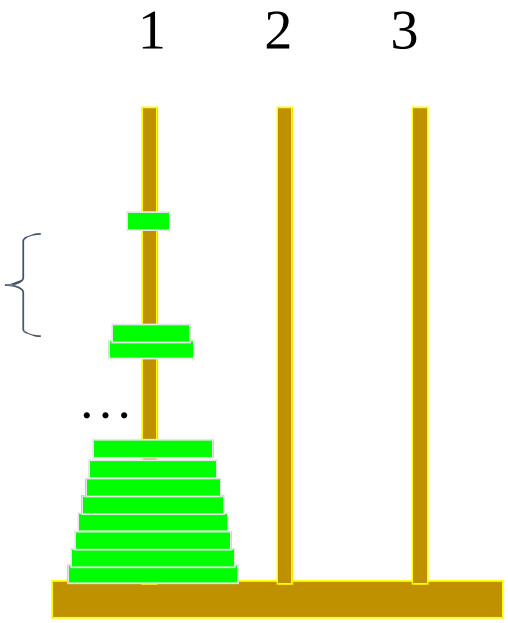
- **问题归约表示的组成部分：**
  - 一个初始问题描述(An initial problem description)
  - 一套把问题变换为子问题的操作符(A set of operators transforming the initial problem into subproblems and sub-subproblems)
  - 一套本原问题描述(A set of description of primitive problems)
- **问题归约的实质：**
  - 从目标(要解决的问题)出发逆向推理，建立子问题以及子问题的子问题，直至最后把初始问题归约为一个平凡的本原问题(可直接求解的问题)集合。

Hanoi-Tower Problem （梵塔难题）

- 3根柱子和**64个盘**

Constrain Conditions（约束条件）

- 每次只许移动一个;
- 只能先搬动柱子顶部的圆盘;
- 不许把尺寸较大的圆盘堆放在尺寸较小的圆盘上



- Moving times:

  64 disks $2^{64}-1 \approx 2^{64} = 10^{19.27}$

If one person move 1 disk in one second, then to finish this problem needs more than 3000 billion years.(30000多亿年)

问题归约示例

- Three disk Hanoi-Tower(3圆盘梵塔难题)
  - 问题：有3个柱子(1，2和3)和3个不同尺寸的圆盘（A，B和C）。在每个圆盘的中心有一个孔，所以圆盘可以堆叠在柱子上。最初，3个圆盘都堆在柱子1上，如何按规则将3个圆盘都堆在柱子3上?

(a) initial configuration　　(b) Goal configuration

Fig.10 The Hanoi Tower problem

Hanoi-Tower Problem

- 问题表示
  - 与状态空间法不同，不能仅表示“状态”，需要表示目前的“问题”
  - 假设用向量（$D_n, D_{n-1}, \ldots, D_1$）表示从大到小的圆盘所在的柱子号，则
    - 初始状态：(1, 1, …, 1)
    - 目标状态：(3, 3, …, 3)
  - 初始问题则为要将初始状态转变为目标状态

$$(1, 1, \ldots, 1) \Rightarrow (3, 3, \ldots, 3)$$

问题归约

- 将原始问题归约为一个较简单问题集合

  移动步骤和约束条件

  - 要把所有圆盘都移至柱子3，我们必须首先把圆盘C移至柱子3；而且在移动圆盘C至柱子3之前，要求柱子3必须是空的.
  - 只有在移开圆盘A和B之后，才能移动圆盘C；而且圆盘A和B最好不要移至柱子3就不能把圆盘C移至柱子3。因此，首先应该把圆盘A和B移到柱子2上.
  - 然后才能够进行关键的一步，把圆盘C从柱子1移至柱子3，并继续解决难题的其余部分。

Problem Reduction

- 归约过程
  - (a) 移动圆盘A和B至柱子2的双圆盘难

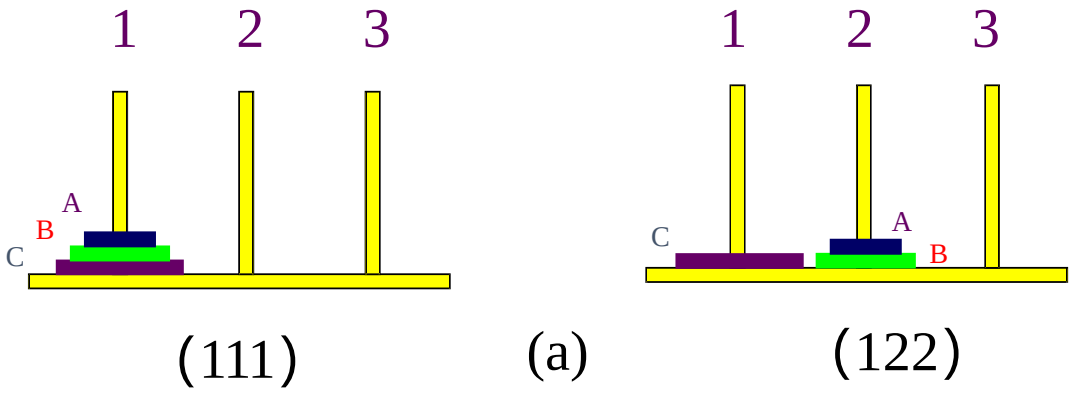


Fig.11 Reduction for the Hanoi Tower problem

Problem Reduction

(122) (b) (322)

- (b) 移动圆盘C至柱子3的单圆盘难题
- (c) 移动圆盘A和B至柱子3的双圆盘难题

(322) (c) (333)

Fig.11 Reduction for the Hanoi Tower problem

Problem Reduction

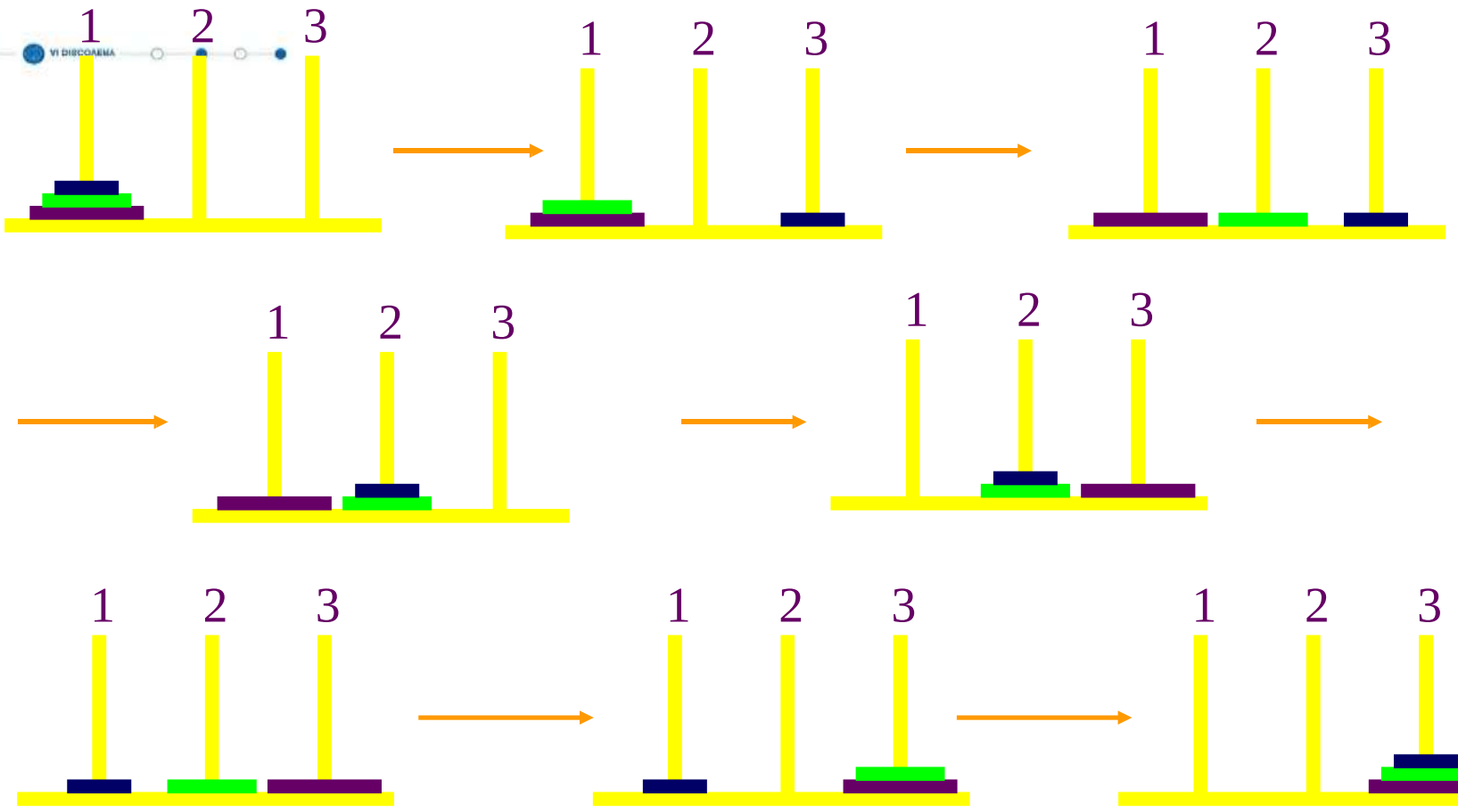
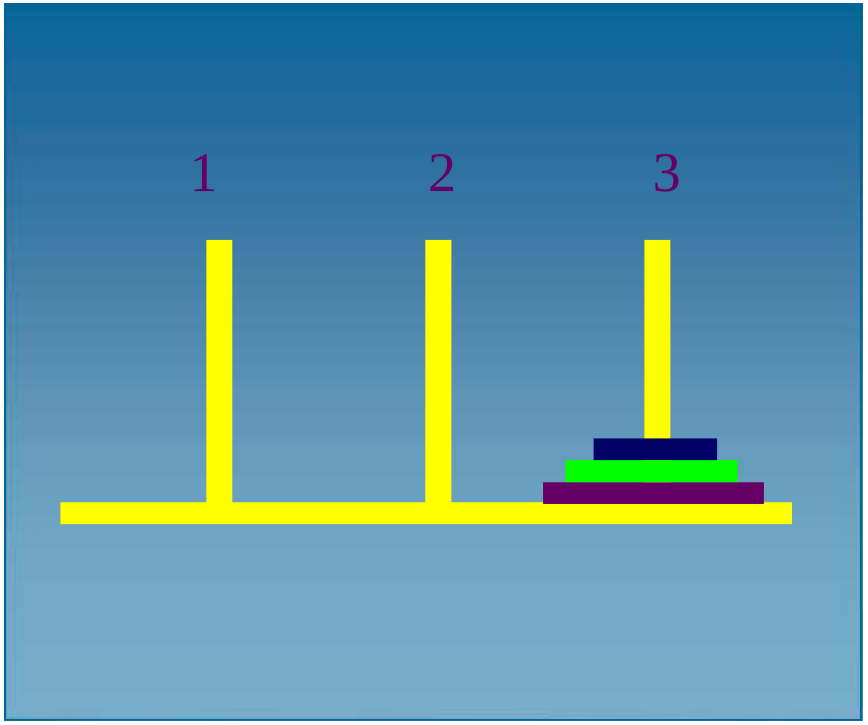


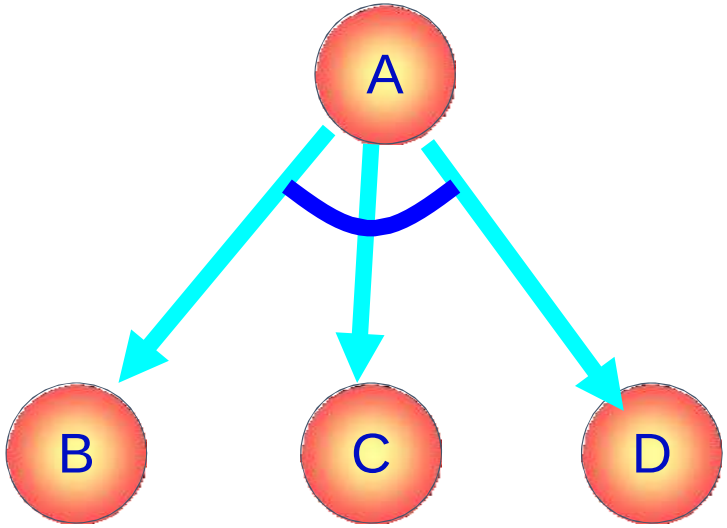
Fig.12 The solving process of Hanoi Tower problem

Three-disk Hanoi Tower

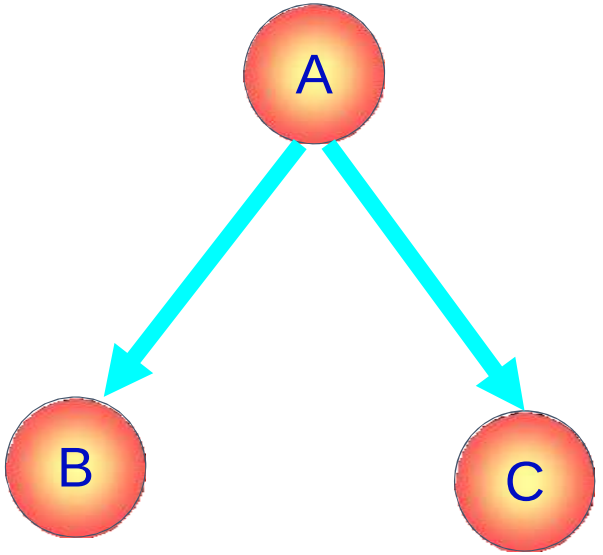


AND/OR Graph Representation
与或图表示

- 与图、或图、与或图



与图



或图

与或图表示

- The Idea of AND/OR Graph
  - 一般地，我们用一个类似**图的结构**来表示把问题归约为**后继问题的替换集合**，这种结构图叫做**问题归约图**，或叫**与或图**。

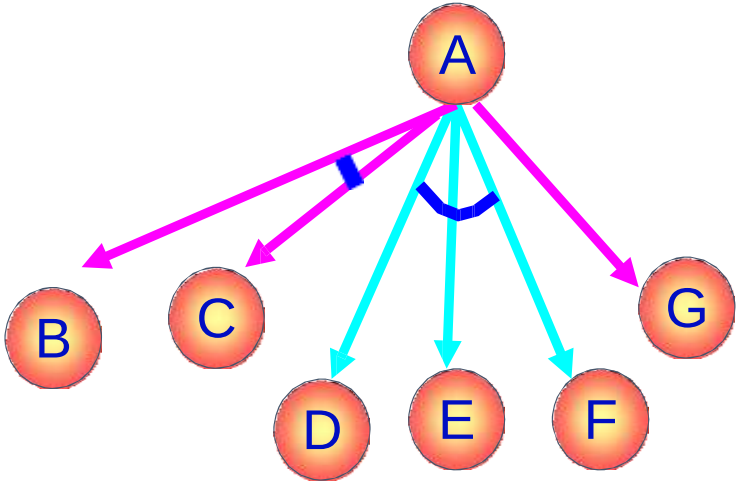


Fig.13 子问题替换集合结构图

与或图表示

- 通过在图中加入附加节点使得对应多个后继问题的每个子集合都有自己的父节点。

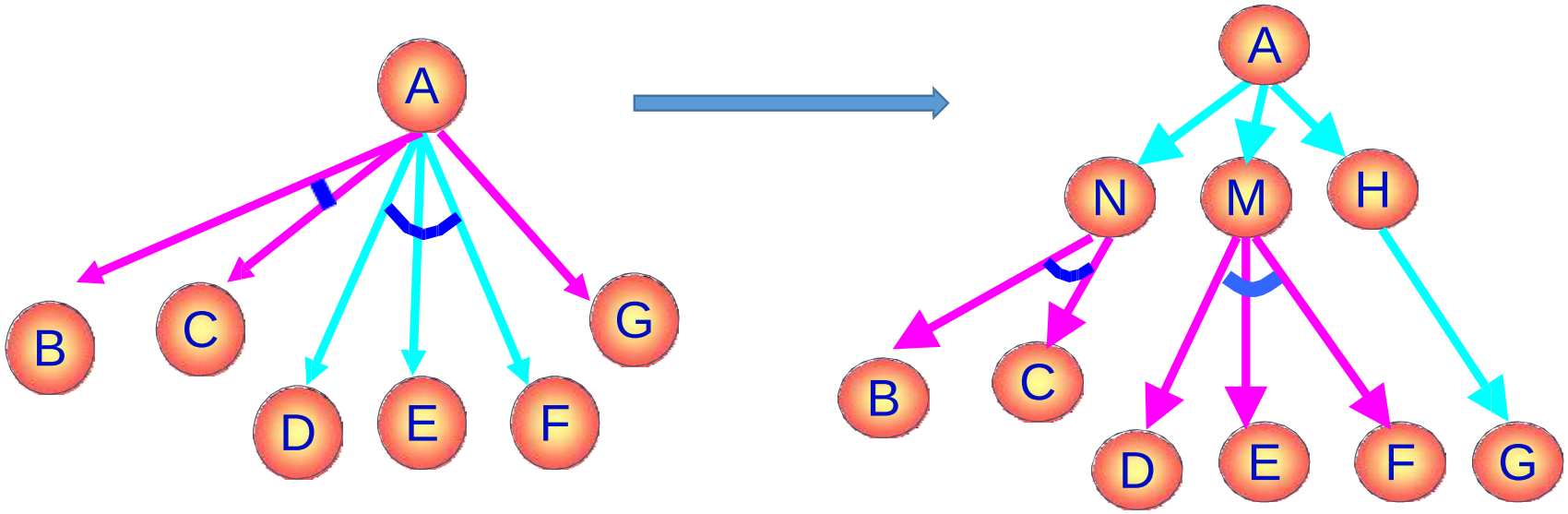


Fig.14 与或图

与或图表示

- 一些关于与或图的术语：
  - 父节点
  - 子（后继）节点
  - 或节点
  - 与节点
  - 弧线
  - 起始节点
  - 终叶节点

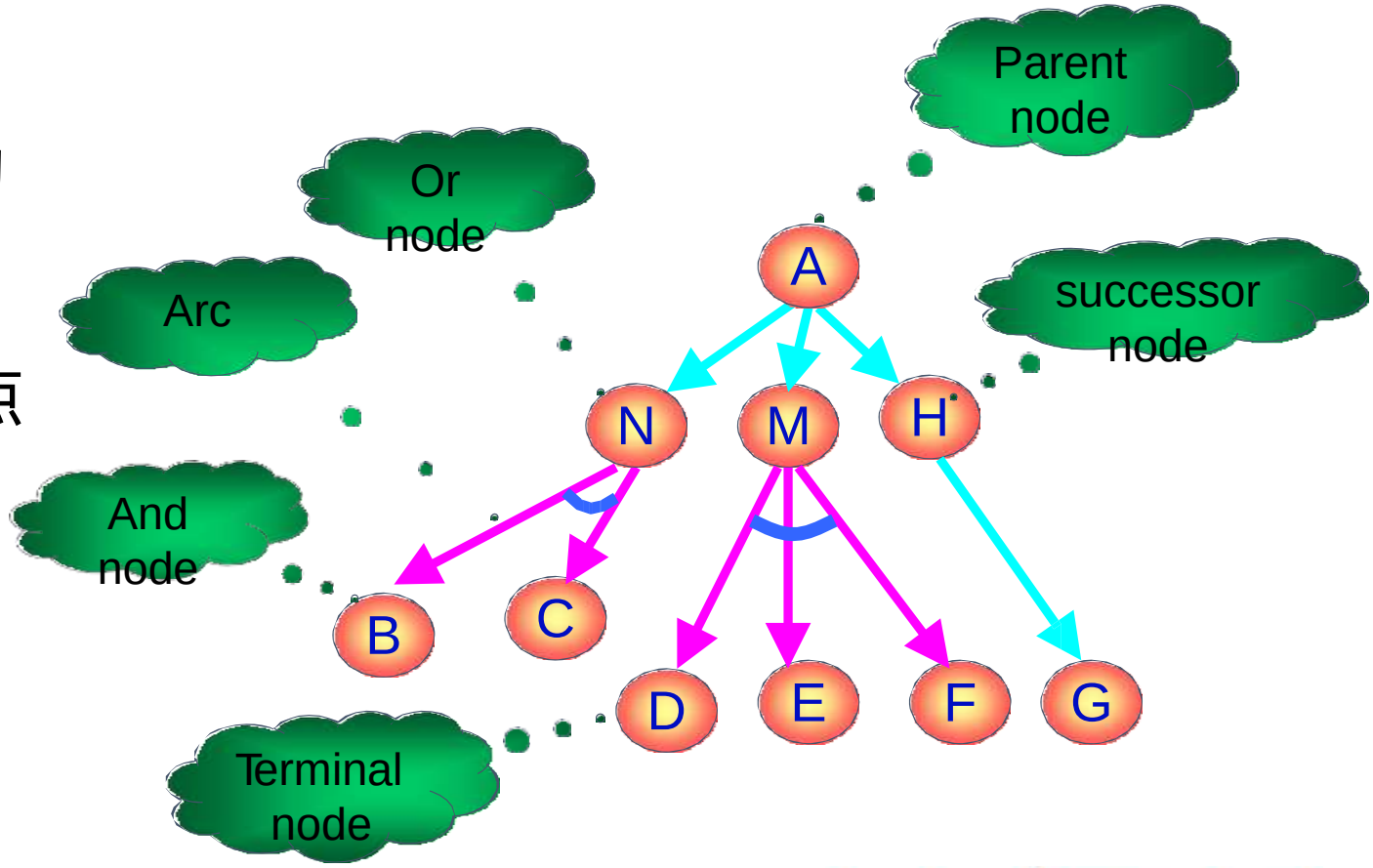
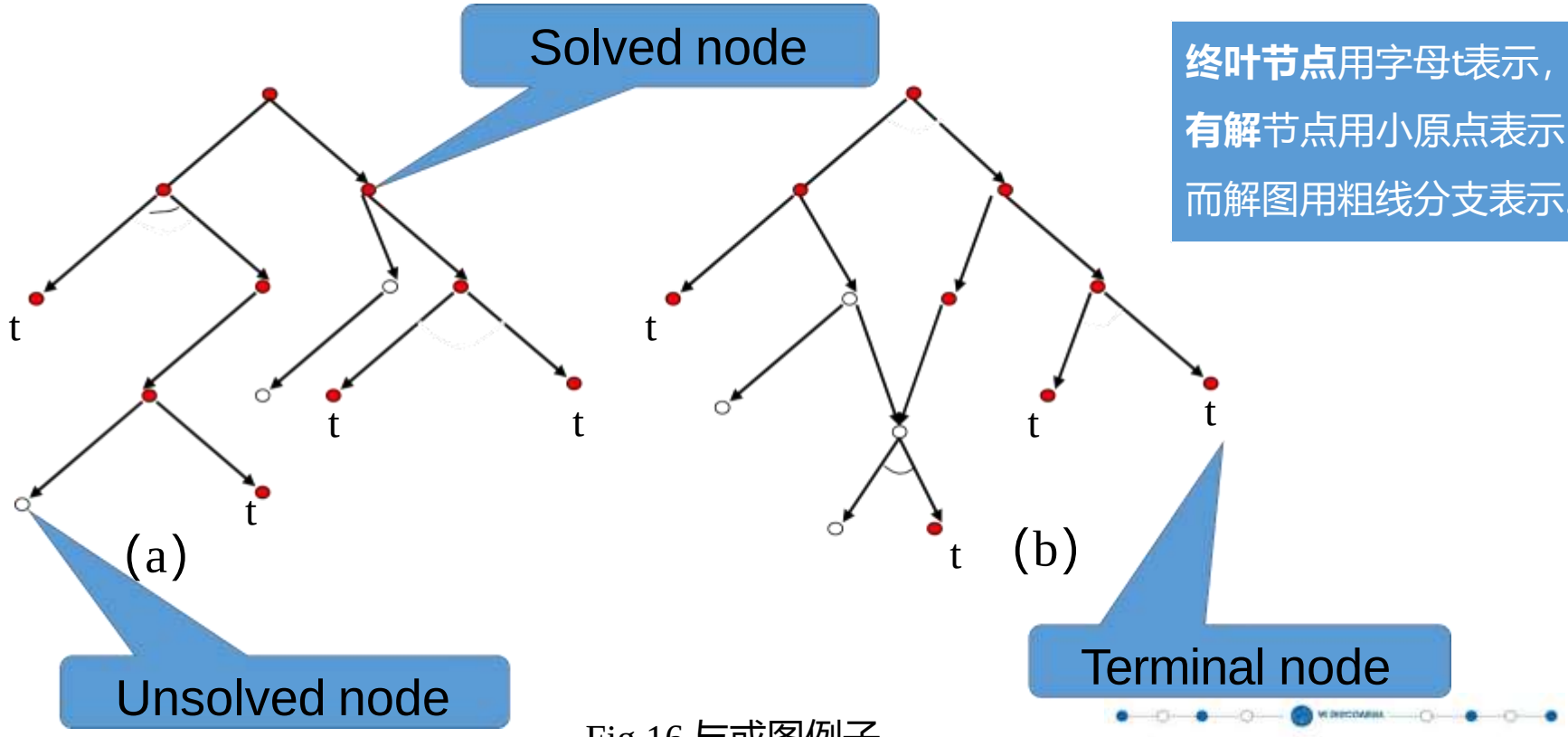


Fig.15 Structure of an AND/OR graph

与或图表示

- 可解节点和不可解节点Solvable node and unsolvable node



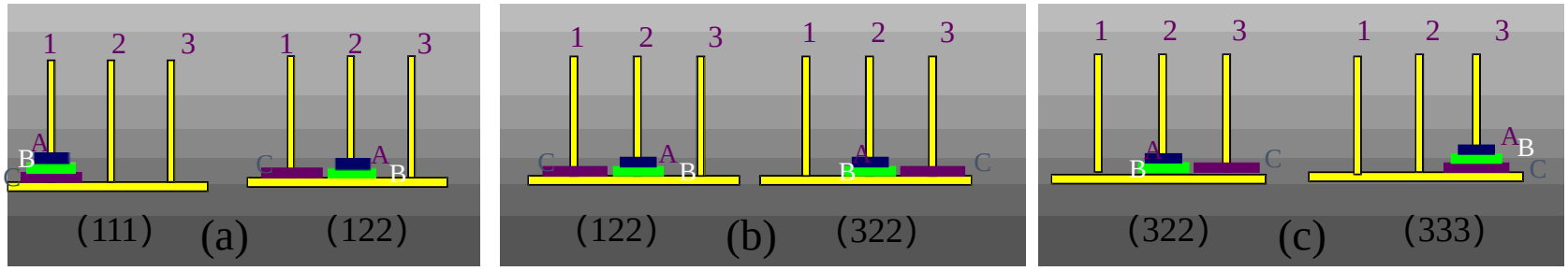
终叶节点用字母t表示，
有解节点用小原点表示，
而解图用粗线分支表示。

Fig.16 与或图例子

与或图表示

- **与或图的构建规则：**
  - **每个节点**对应一个问题或问题集合
  - **终叶节点**对应一个本原问题
  - 通过**算符**将问题转换为一个子问题集合
  - 通过**连接线**将同一个子问题的“与”节点连接起来

与或图表示：3圆盘问题



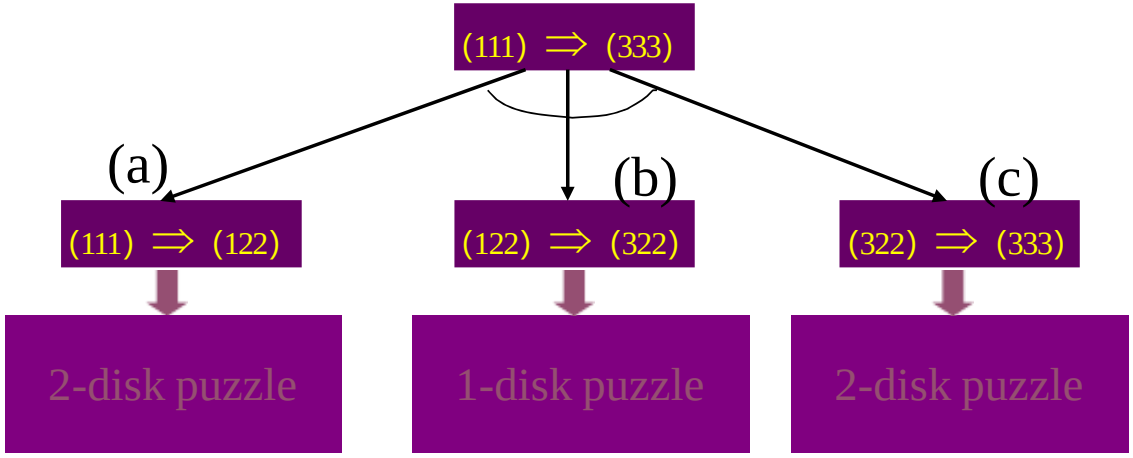


Fig.17 Reduction graph for the 3-disk Hanoi Tower

与或图表示

- Puzzle (b)： a primivtive problem
- Puzzle (a) and (c)： 2-disk puzzle

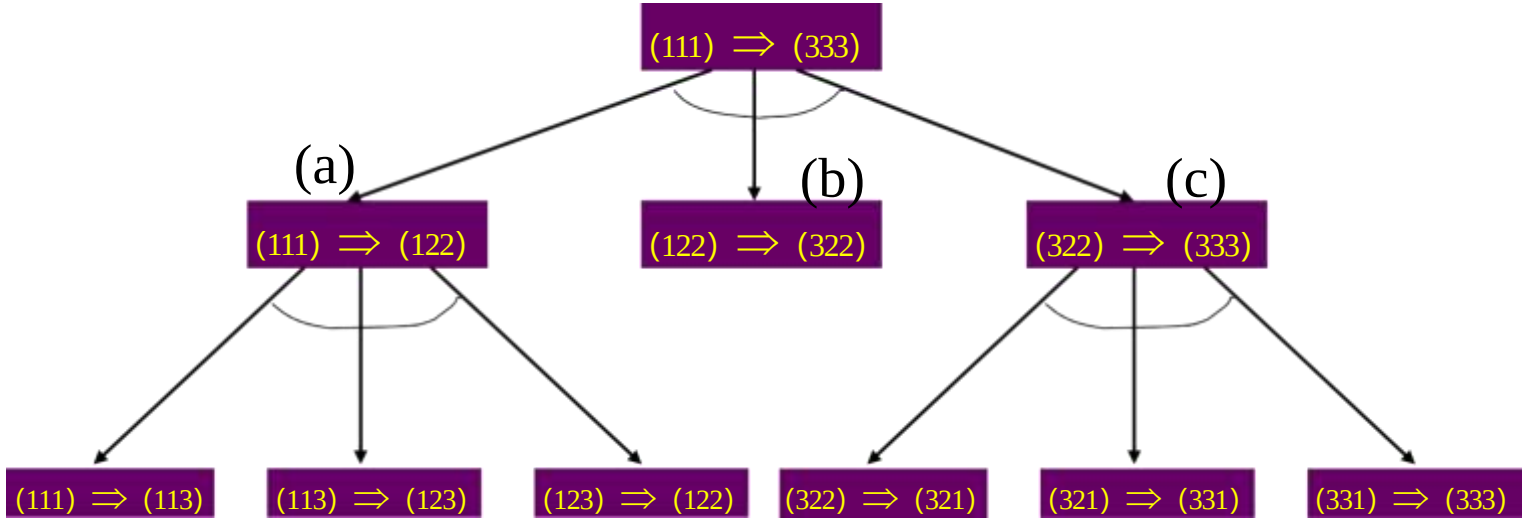
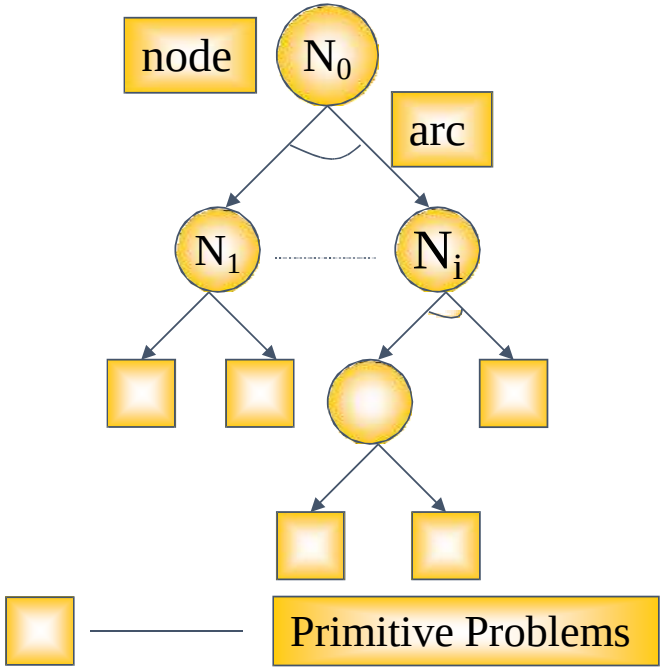


Fig.18 And/Or graph for the 3-disk Hanoi Tower

(注意数据结构)

与或图表示

- 结论
  - 每个节点代表一个明显的问题或者问题集合
  - 除了起始节点之外，每个节点只有一个父辈节点。
  - 实际上，这些图是 与或树



与或图表示4圆盘问题

作业1：试用四元数列结构表示四圆盘梵塔问题，并画出求解该问题的与或图。

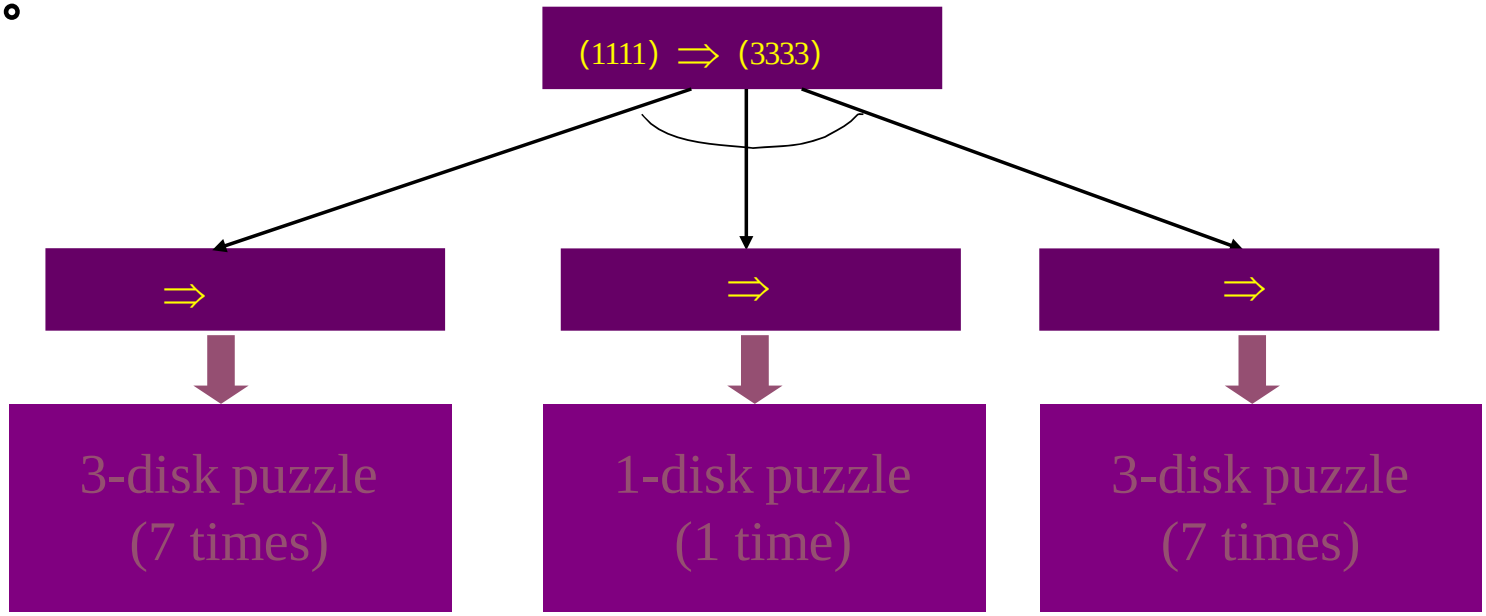
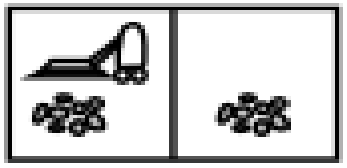


Fig.19 Reduction graph for the 4-disk Hanoi Tower

作业2：求复杂吸尘器世界状态个数。



- 吸尘器电源开关 Power: on/off/sleep
- 传感器 Dirt-sensing camera: on/off
- 刷子高度 Brush-height: 1/2/3/4/5
- 地毯块 Position: 10
- 地毯状态 Rug: dirty/clean

作业3：请写出**传教士野人问题**(Missionaries& Cannibals,MC问题）中的操作符、操作符执行的前提条件和执行后问题空间状态的变化情况.

注意，状态可以有多种表示方式，操作符或算子（可自行定义）。