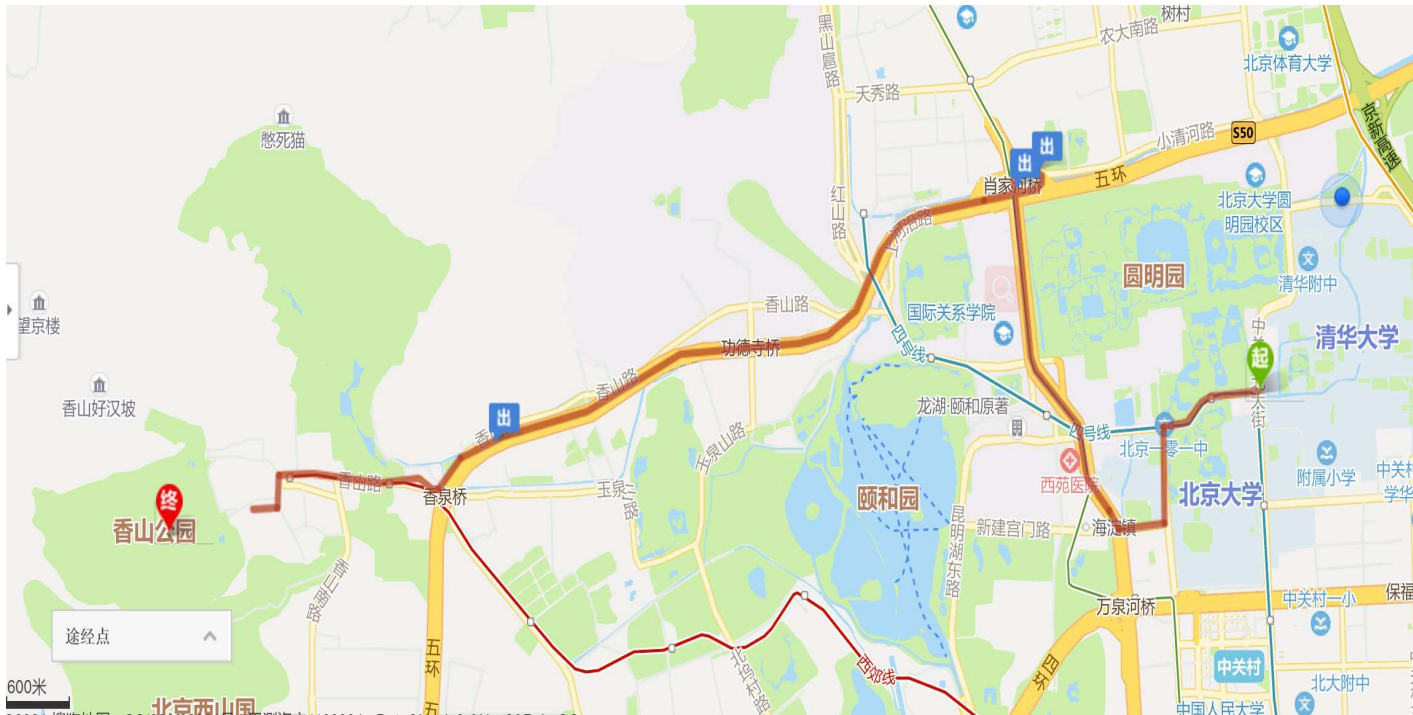


第一章 搜索问题

- ◆ 什么是搜索问题?
- ◆ 从自动导航说起





搜索问题

◆ 内容：

- 状态空间的搜索问题

◆ 搜索方式：

- 盲目搜索
- 启发式搜索

◆ 关键问题：

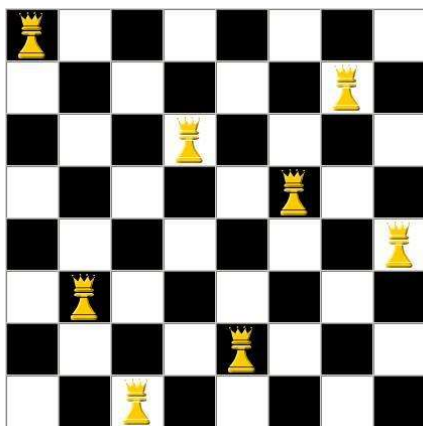
如何利用知识，尽可能有效地找到问题的解（最佳解）。



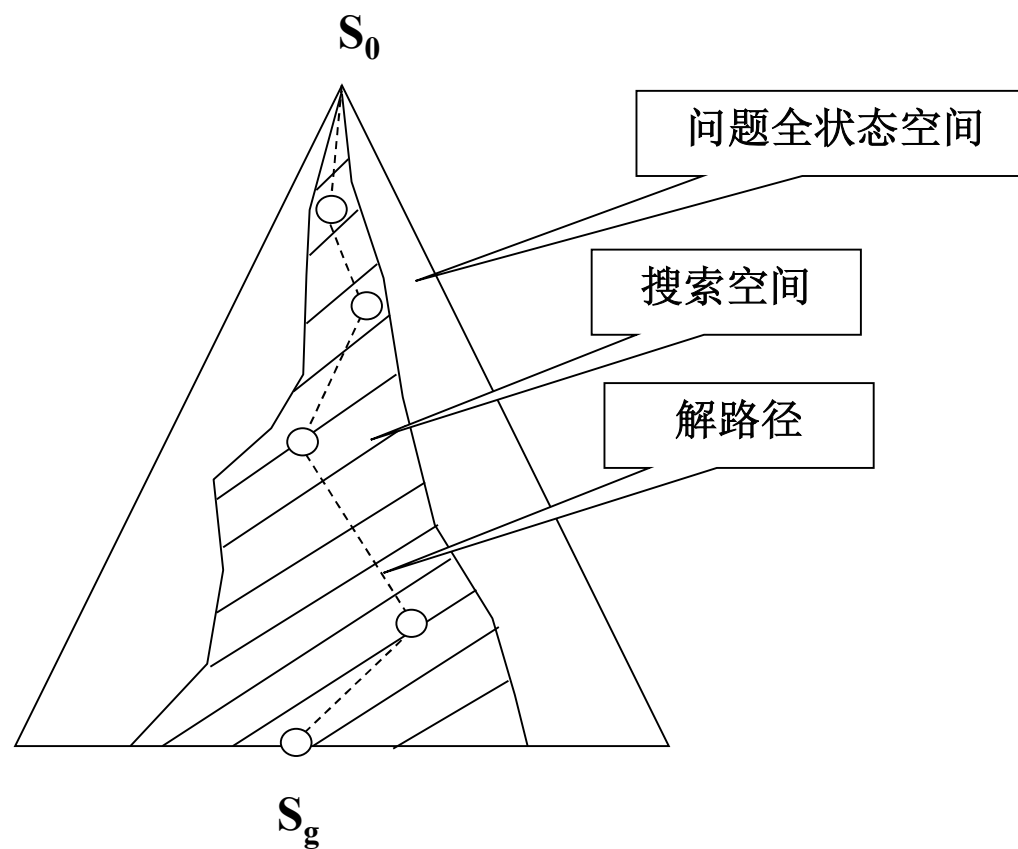
搜索问题（续2）

◆ 问题举例：

- 地图路径
- 传教士和野人问题
- 华容道问题
- 八皇后问题



搜索问题（续3）





搜索问题（续4）

◆讨论的问题：

- 有哪些常用的搜索算法。
- 问题有解时能否找到解。
- 找到的解是最佳的吗？
- 什么情况下可以找到最佳解？
- 求解的效率如何。



盲目搜索与启发式搜索

◆ 盲目搜索：

- 深度优先搜索
- 宽度优先搜索

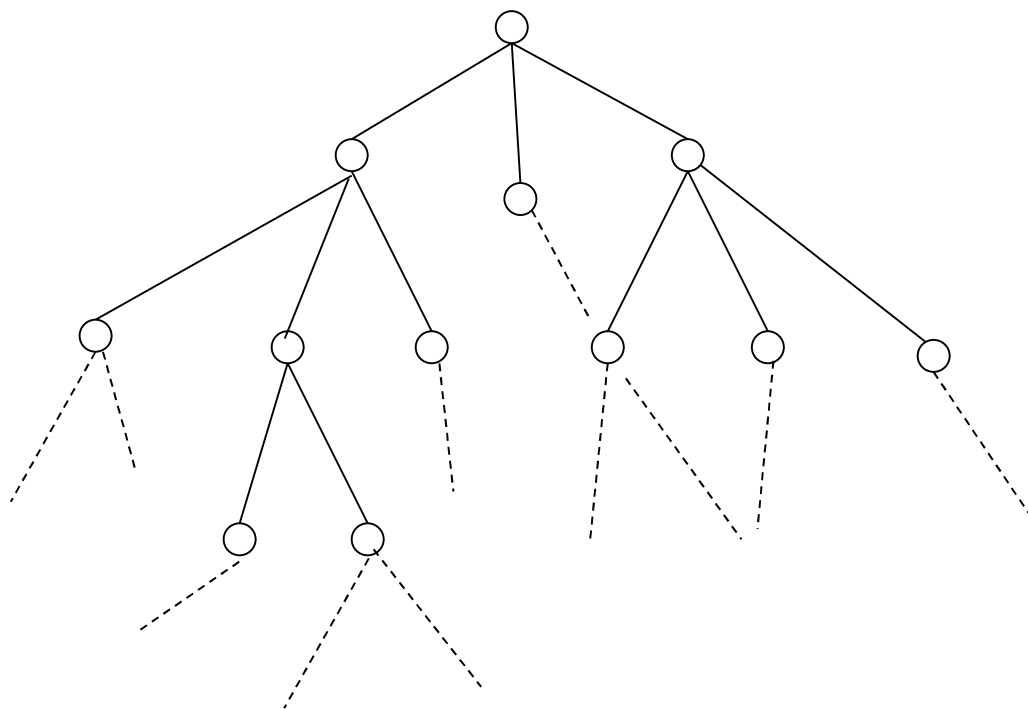
◆ 启发式搜索

- A算法
- A*算法



搜索

◆ 如何选择节点扩展？





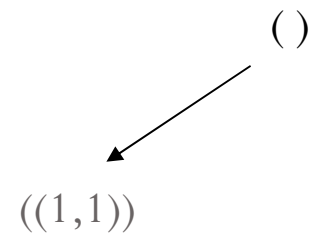
1.1 深度优先搜索

◆ 优先扩展深度深的节点

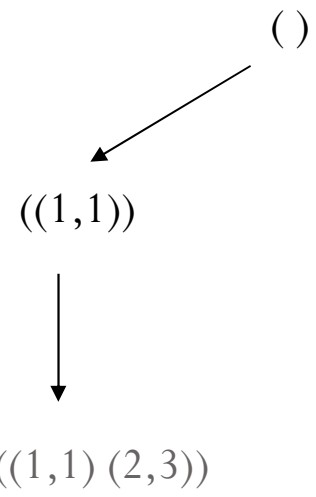
◆ 例：皇后问题

	Q		
			Q
Q			
		Q	

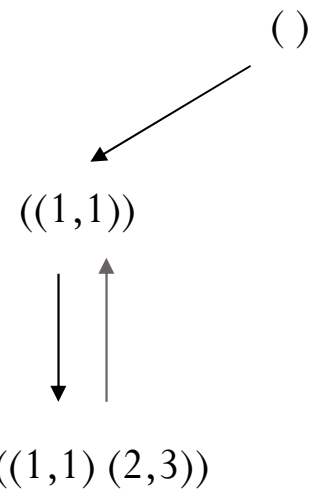
()



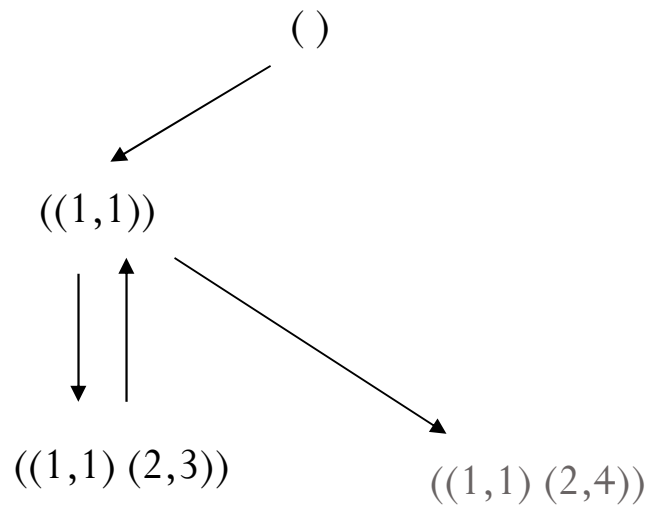
Q			



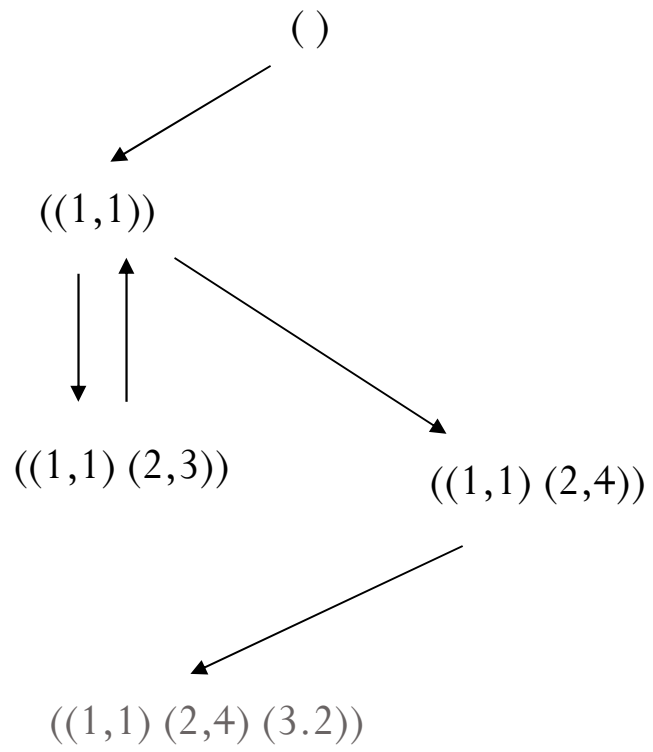
Q			
		Q	



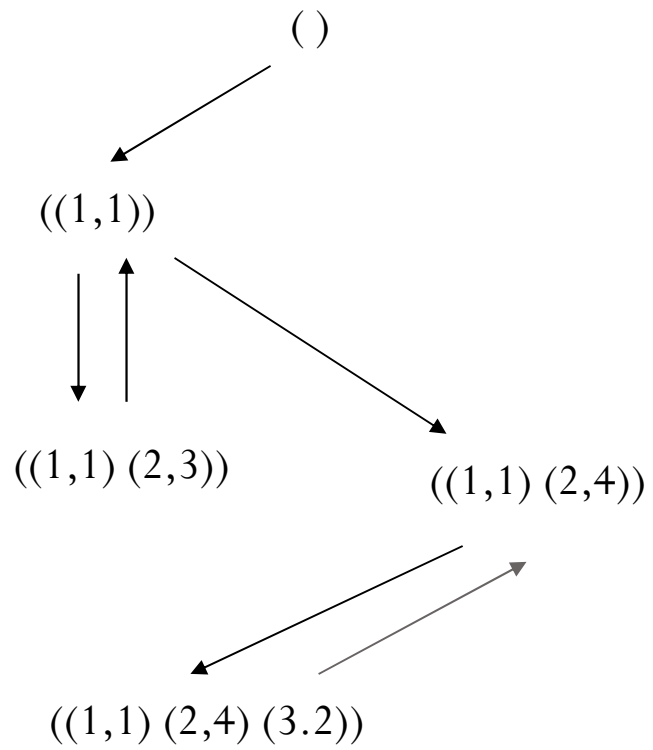
Q			



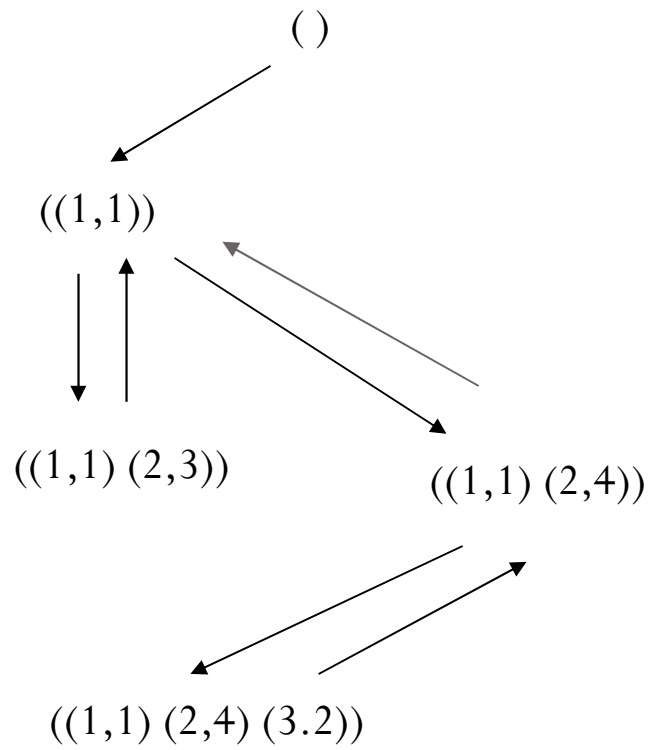
Q			
			Q



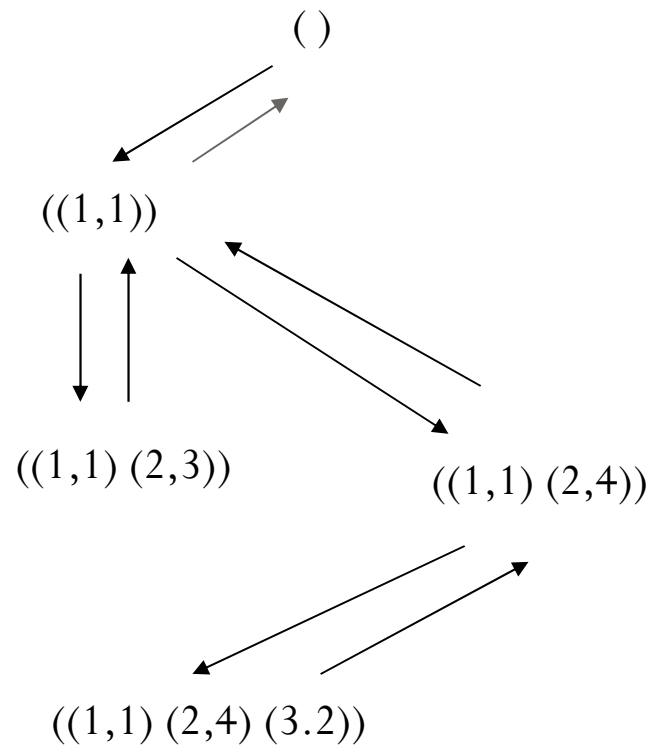
Q			
			Q
	Q		

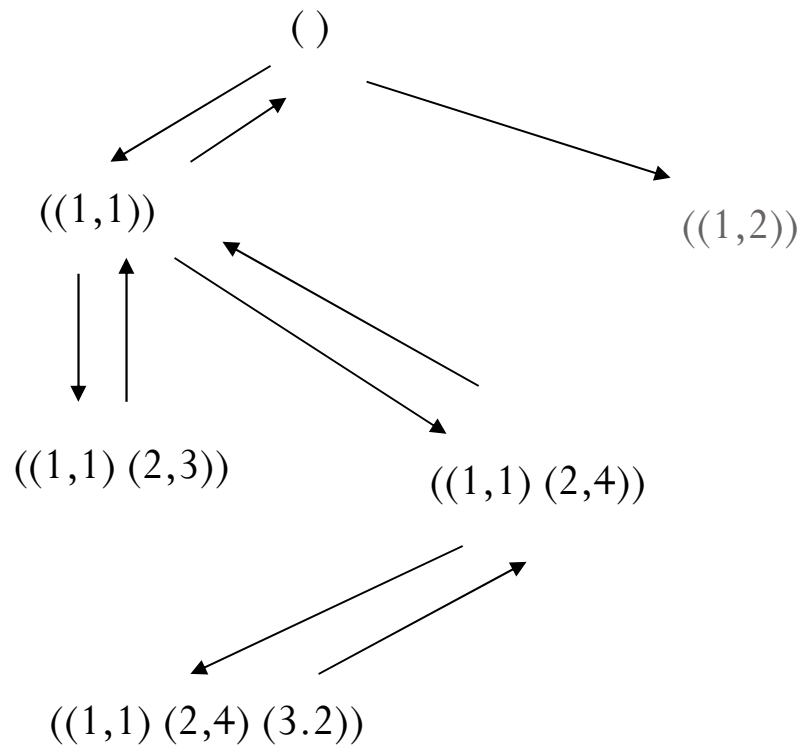


Q			
			Q



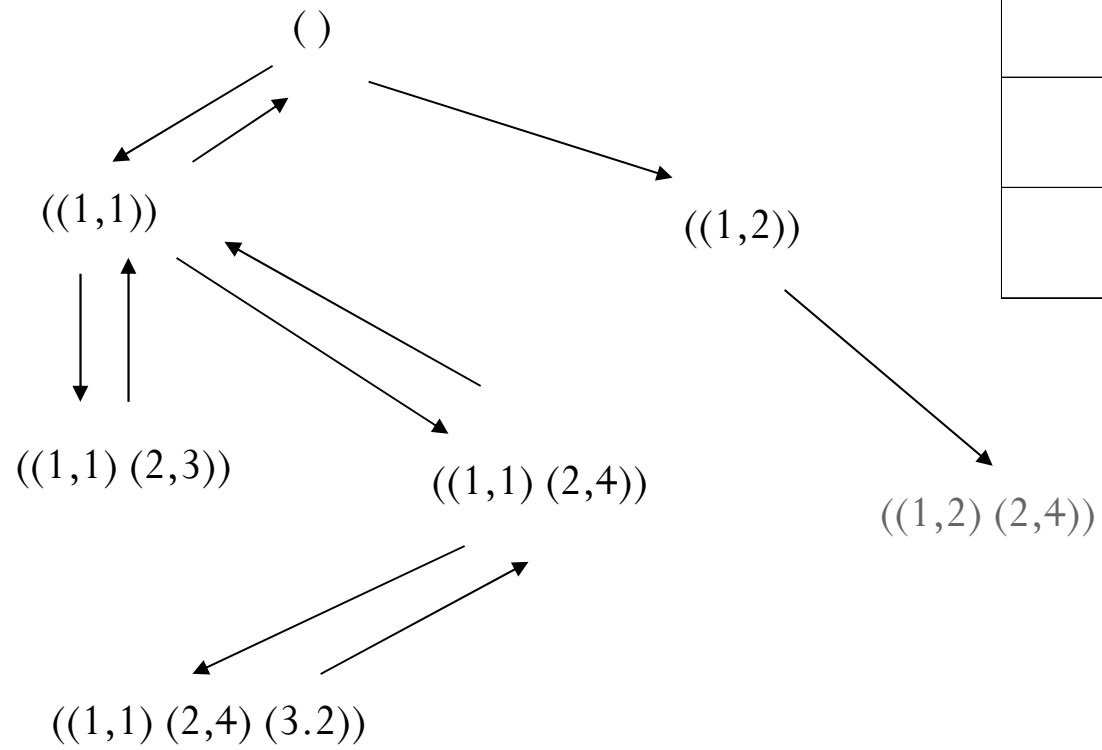
Q			



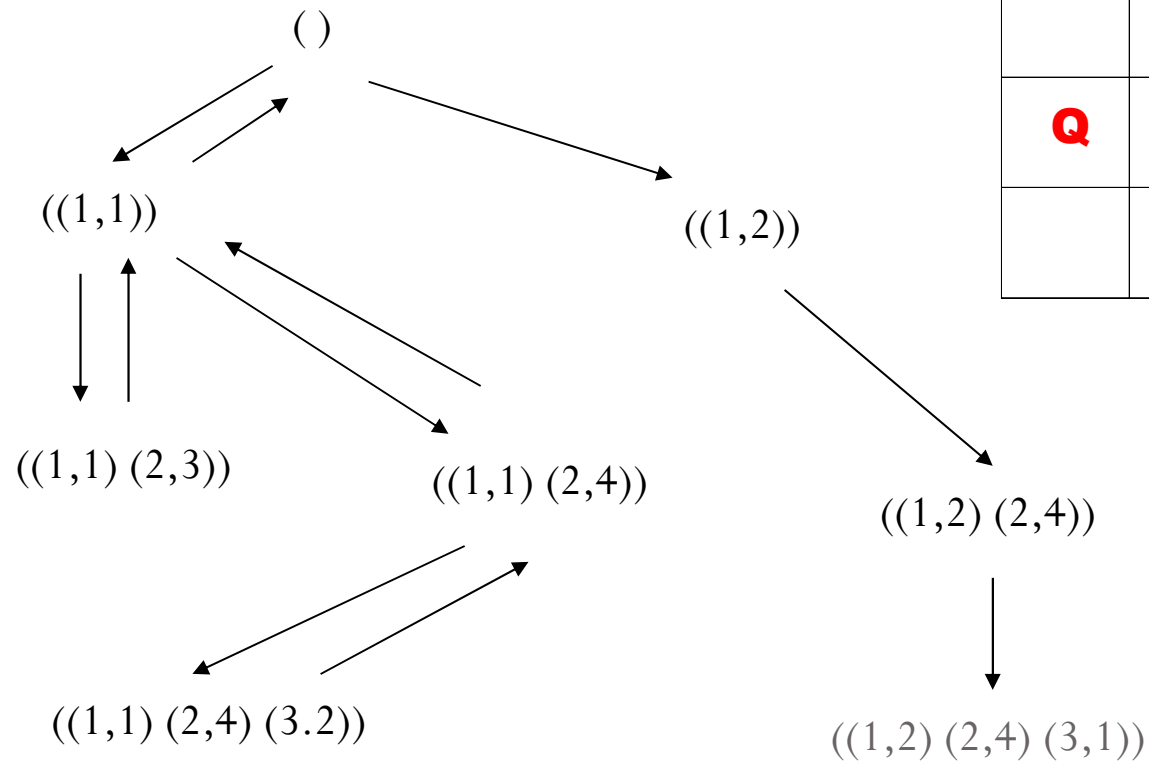


	Q		

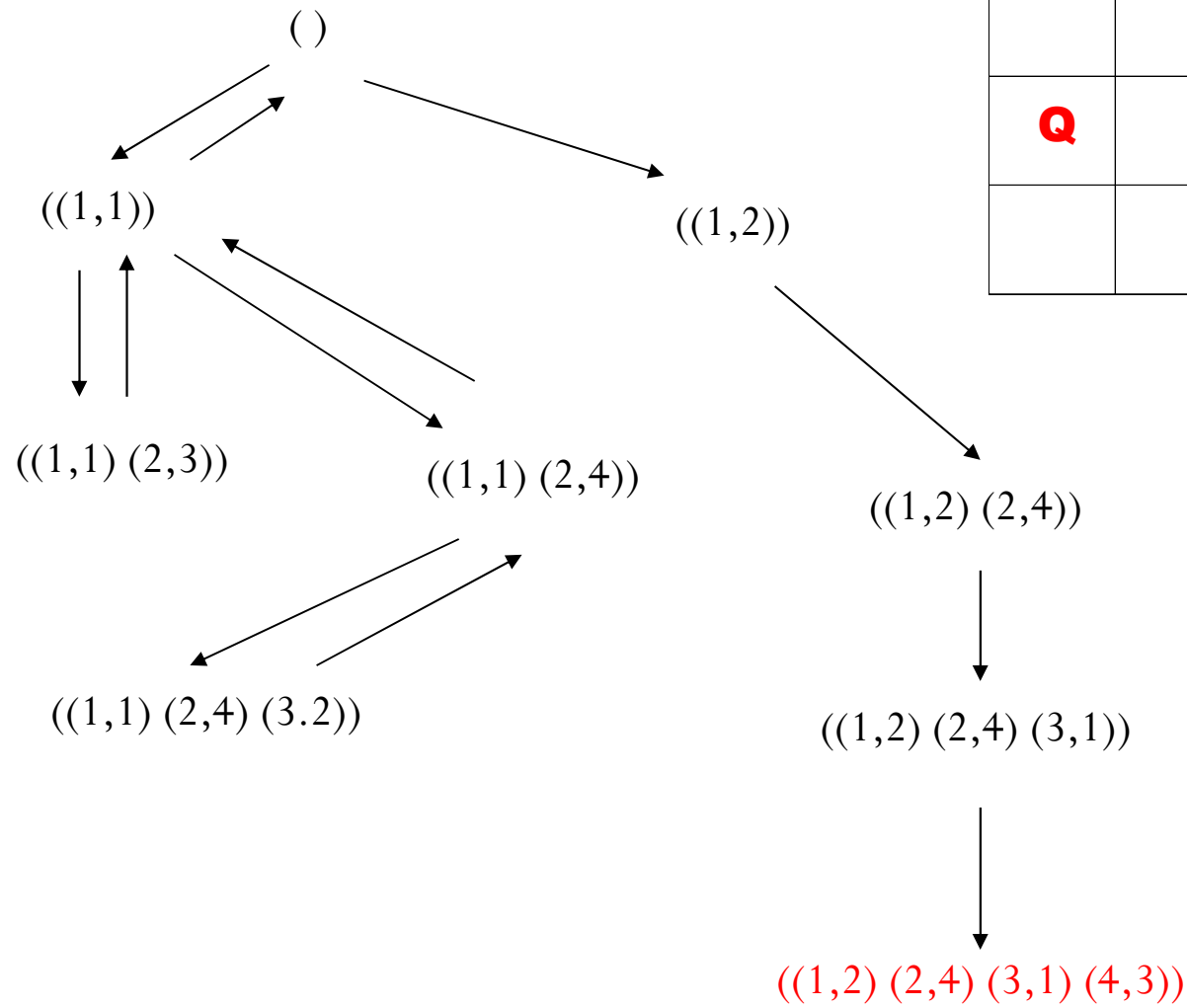
	Q		
			Q



	Q		
			Q
Q			



	Q		
			Q
Q			
		Q	



深度优先搜索的性质

- ◆ 一般不能保证找到最优解
- ◆ 当深度限制不合理时，可能找不到解，
可以将算法改为可变深度限制
- ◆ 最坏情况时，搜索空间等同于穷举
- ◆ 是一个通用的与问题无关的方法
- ◆ 节省内存，只存储从初始节点到当前节点的路径



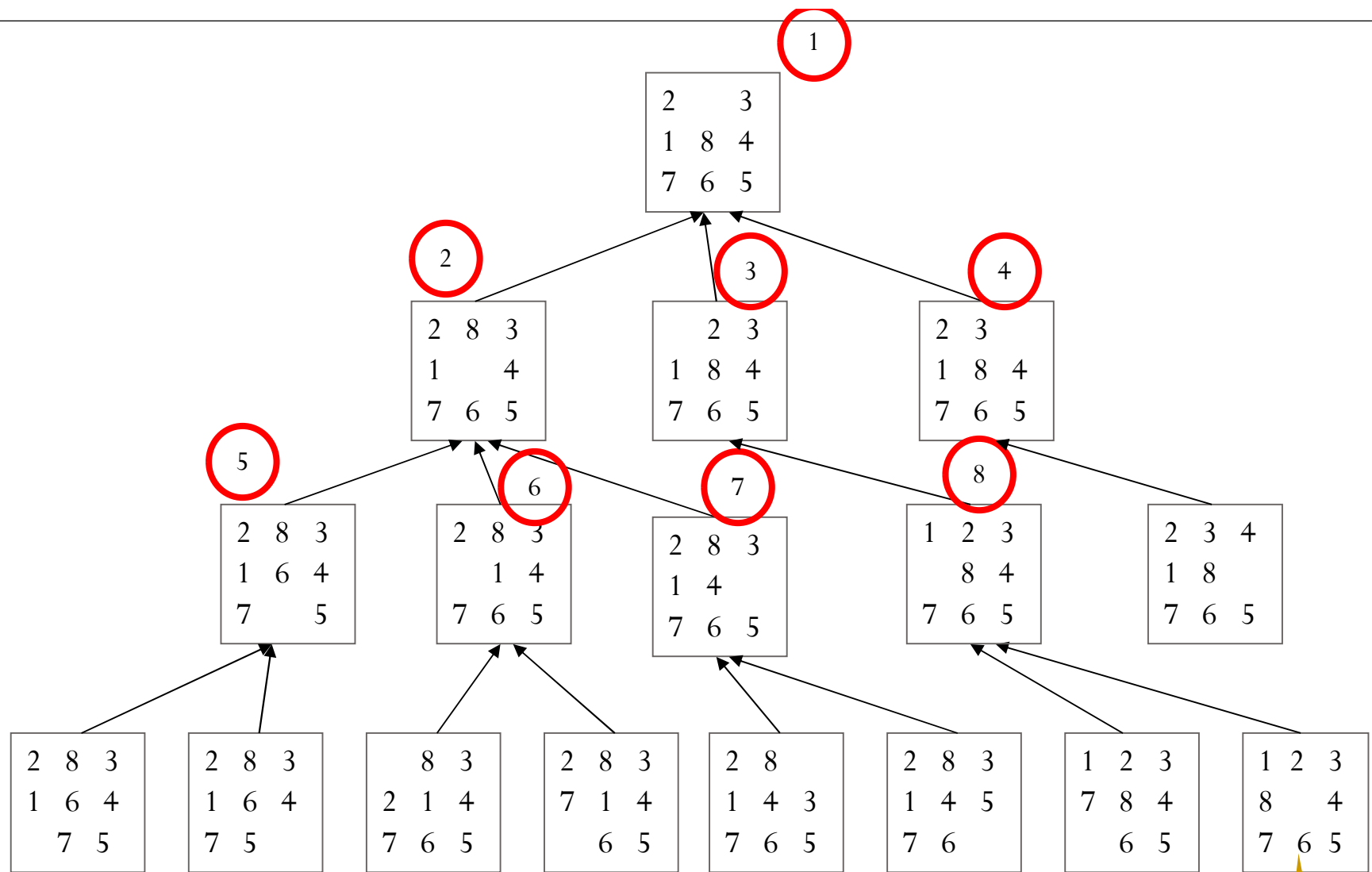
练习题

设有三个没有刻度的杯子，分别可以装8两、8两和3两水。两个8两的杯子装满了水，请问如何在不借助于其他器具的情况下，让4个人每人喝到4两水。

请编程实现。

1.2 宽度优先搜索

◆ 优先扩展深度浅的节点



目标



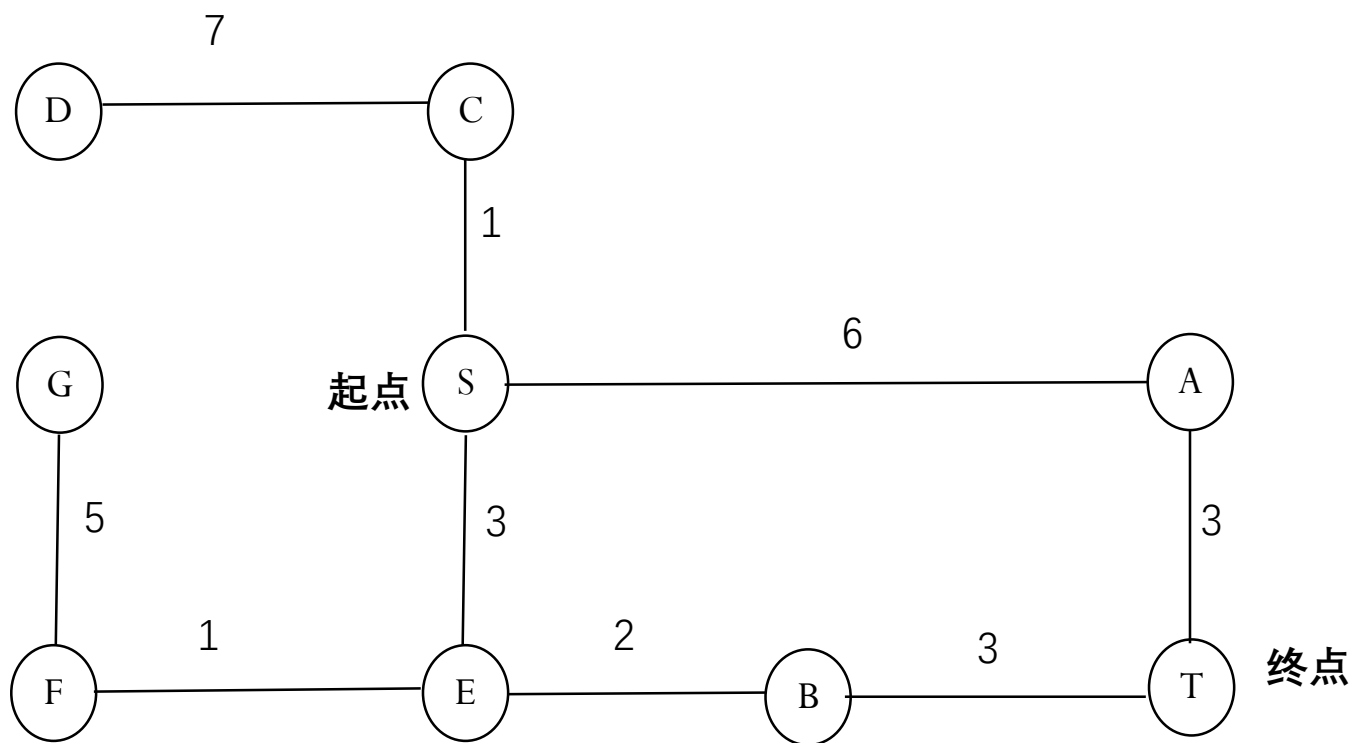
宽度优先搜索的性质

- ◆ 当问题有解时，一定能找到解
- ◆ 当问题为单位耗散值，且问题有解时，一定能找到最优解
- ◆ 方法与问题无关，具有通用性
- ◆ 效率较低
- ◆ 存储量比较大

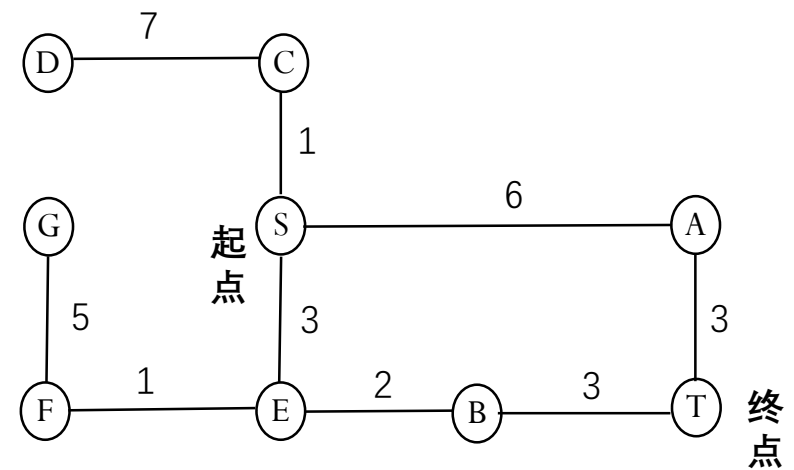
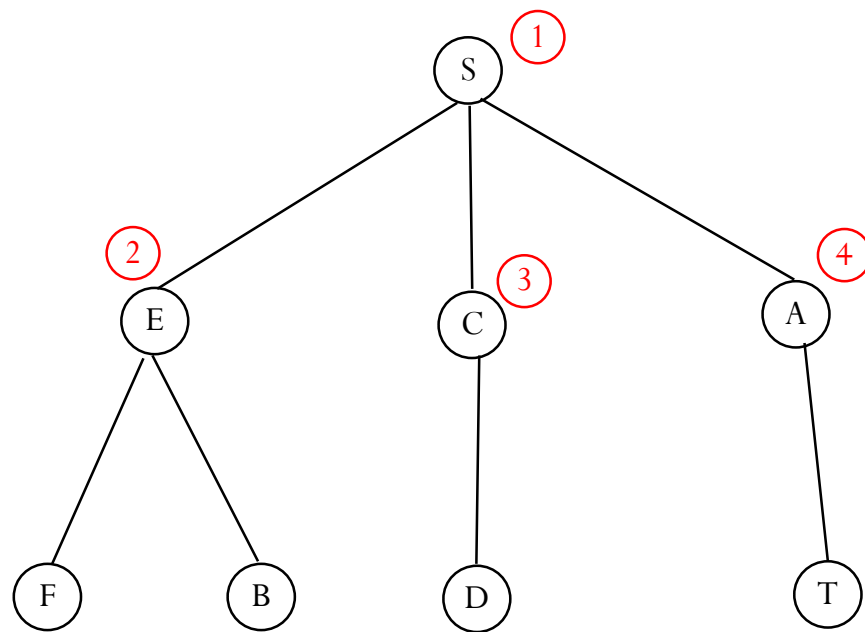


1.3 迪杰斯特拉 (Dijkstra) 算法

◆ 宽度优先算法的不足



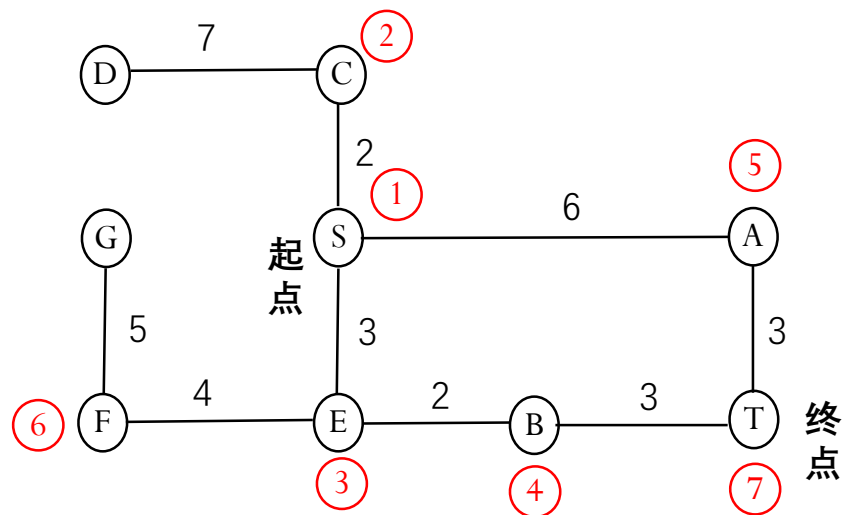
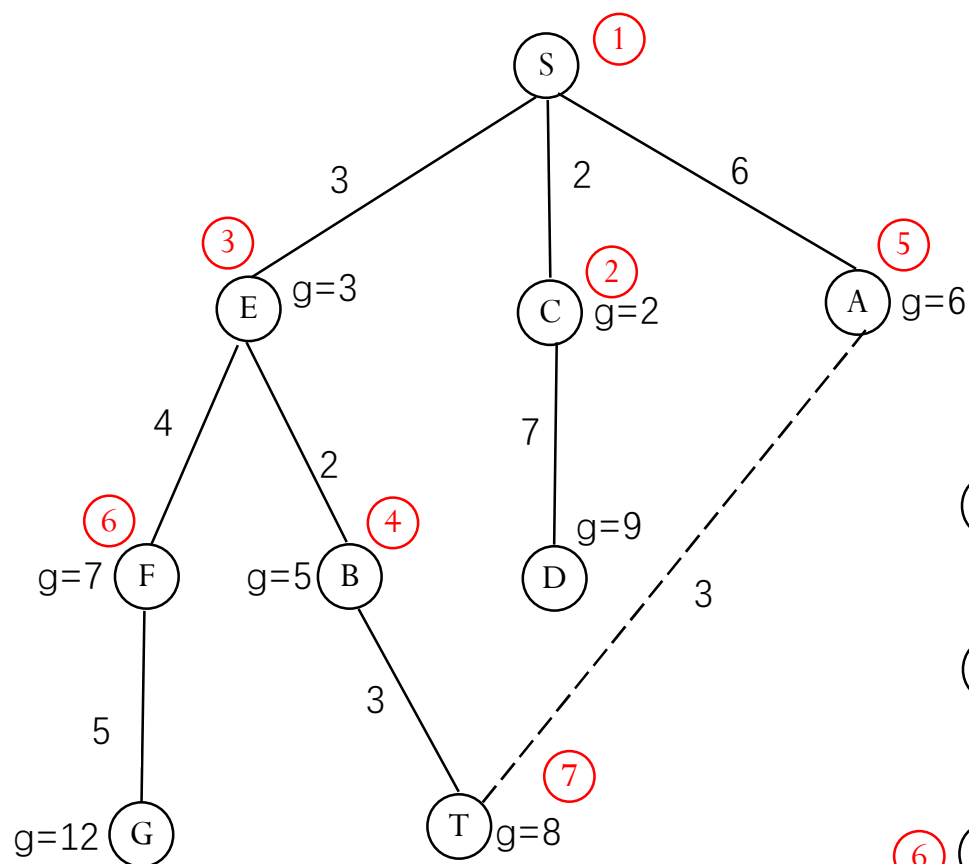
◆ 宽度优先的不足



◆ 宽度优先没有考虑两个节点间的距离

迪杰斯特拉 (Dijkstra) 算法

◆ 优先扩展距离起点最近的节点，直到终点距离最短





迪杰斯特拉 (Dijkstra) 算法

◆ 优点:

- 当问题有解时，可以找到最佳解。

◆ 不足:

- 只考虑了节点距离起点的距离，没有考虑节点到终点的距离



1.4 启发式图搜索

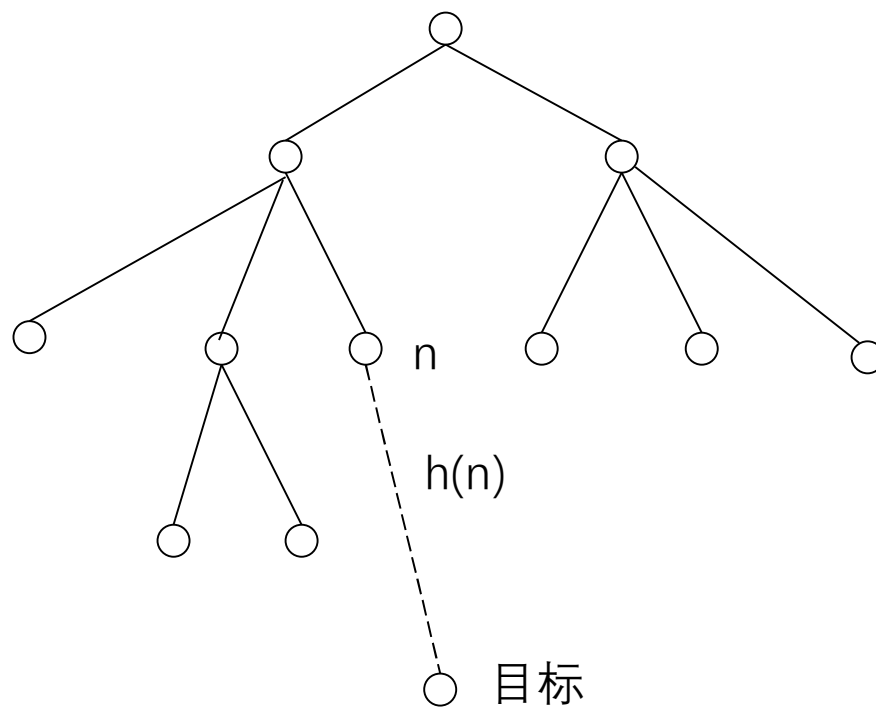
- ◆ 引入启发知识，在保证找到最佳解的情况下，尽可能减少搜索范围，提高搜索效率。



1.4 启发式图搜索

◆ 启发知识:

- 评估节点到达目标的距离



1.4.1 启发式搜索算法A (A算法)

◆ 评价函数的格式:

$$f(n) = g(n) + h(n)$$

$f(n)$: 评价函数

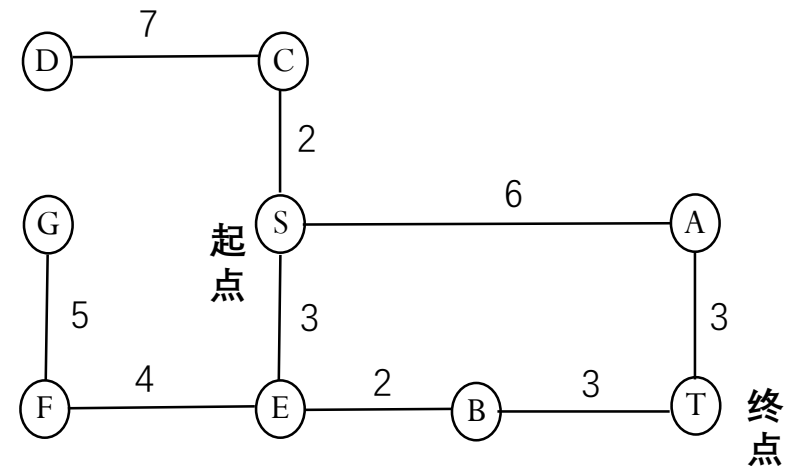
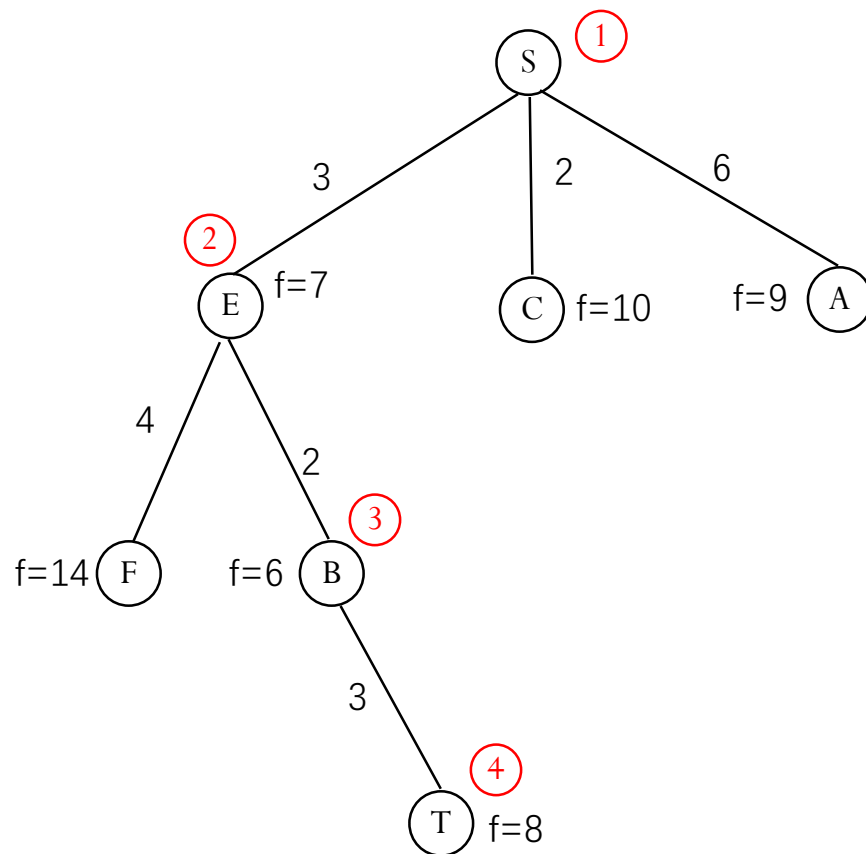
$h(n)$: 启发函数

符号的意义

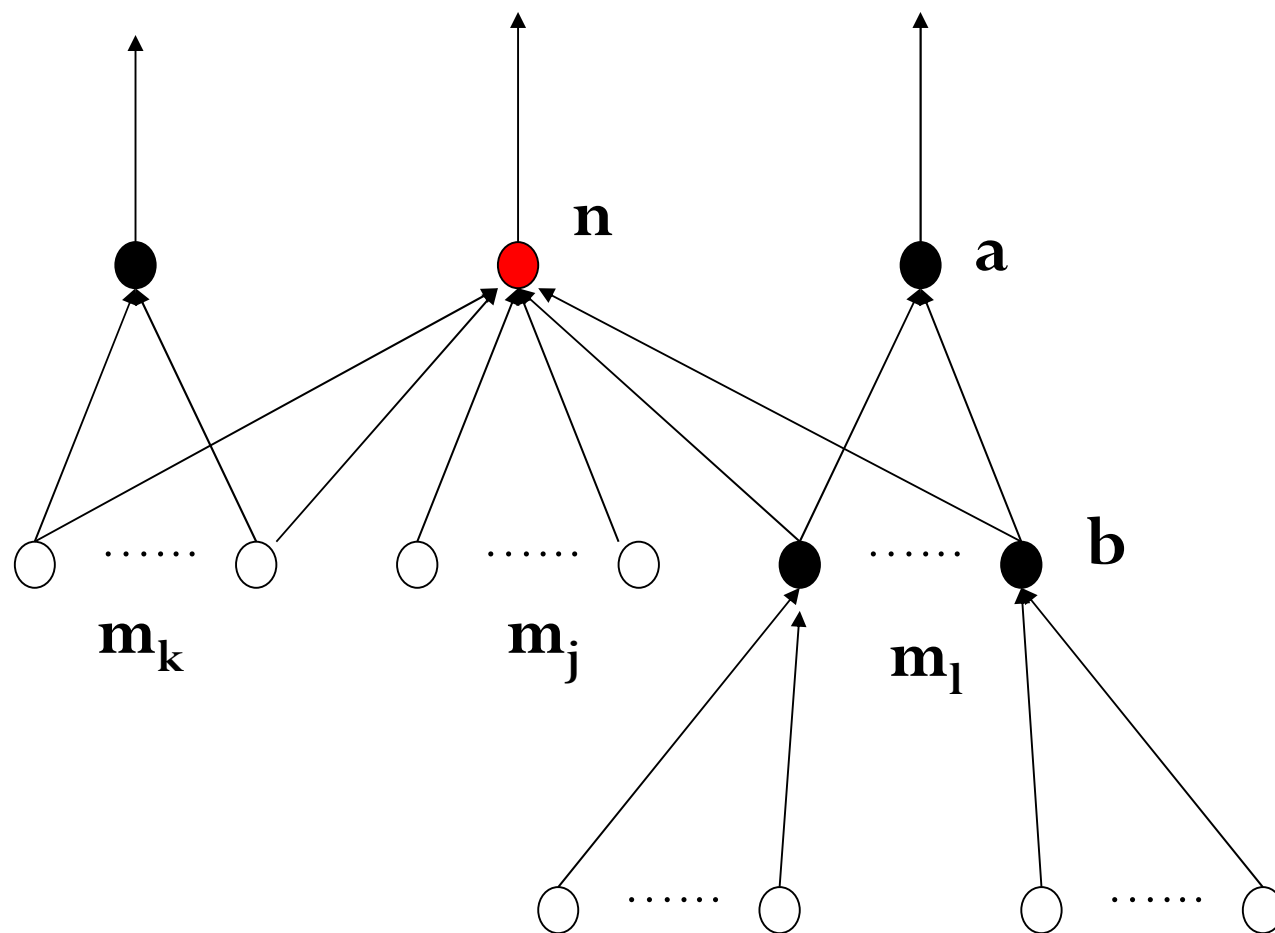
- ◆ $g^*(n)$: 从s到n的最短路径的耗散值
- ◆ $h^*(n)$: 从n到g的最短路径的耗散值
- ◆ $f^*(n) = g^*(n) + h^*(n)$: 从s经过n到g的最短路径的耗散值
- ◆ $g(n)$ 、 $h(n)$ 、 $f(n)$ 分别是 $g^*(n)$ 、 $h^*(n)$ 、 $f^*(n)$ 的估计值
- ◆ 用 $f(n)$ 对待扩展节点进行评价

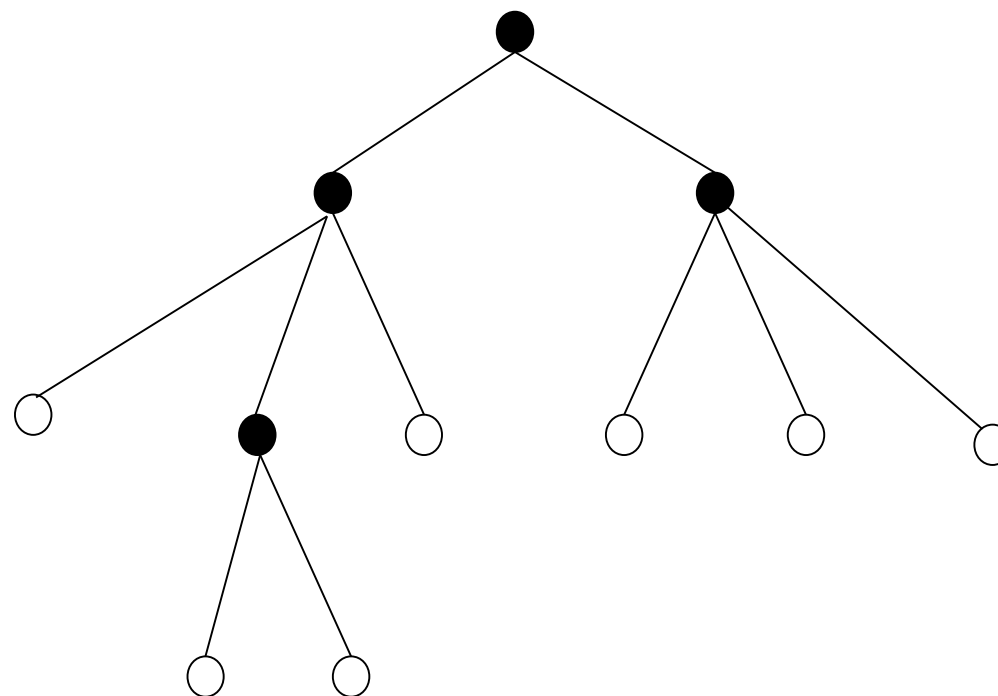
举例

◆ 优先扩展 $f(n)$ 值最小的节点，直到 $f(\text{终点})$ 最小。



$h(A)=3$	$h(E)=4$
$h(B)=1$	$h(F)=7$
$h(C)=8$	$h(G)=12$
$h(D)=13$	$h(T)=0$





● Closed表

○ Open表

A算法



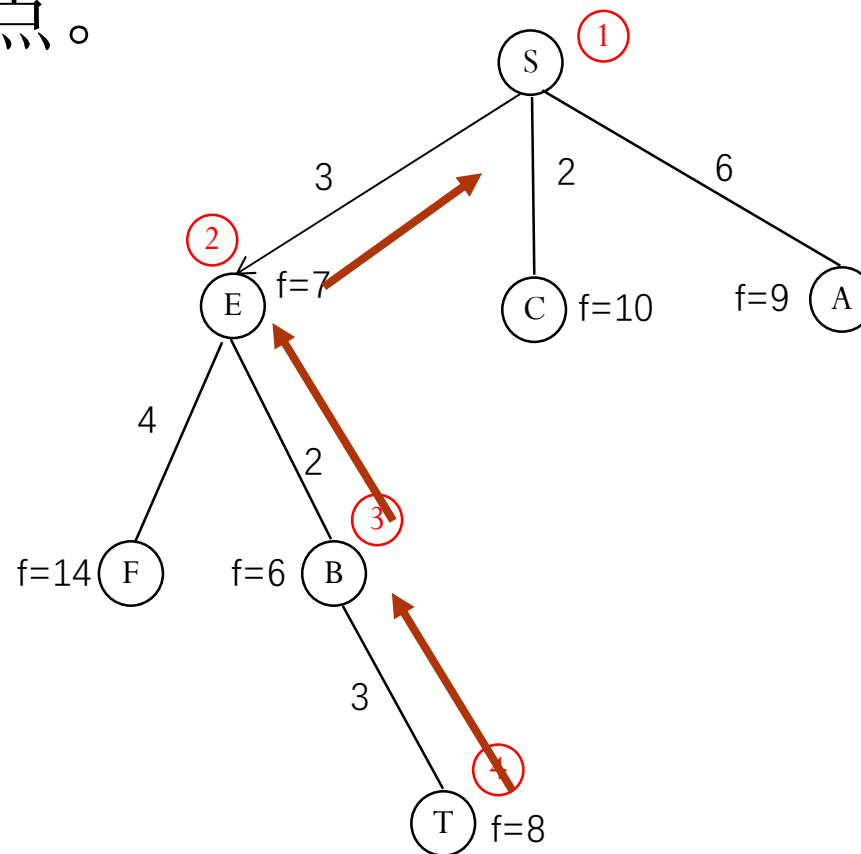
```
A-algorithm (s)  //s为初始节点
OPEN=(s), CLOSED=(), f(s)=g(s)+h(s);
while OPEN不空 do:
  begin
    n=FIRST(OPEN);
    if GOAL(n) THEN return n;
    REMOVE(n, OPEN), ADD(n, CLOSED);
    EXPAND(n)  $\rightarrow \{m_i\}$ ,
    计算f(n, mi)=g(n, mi)+h(mi);
```



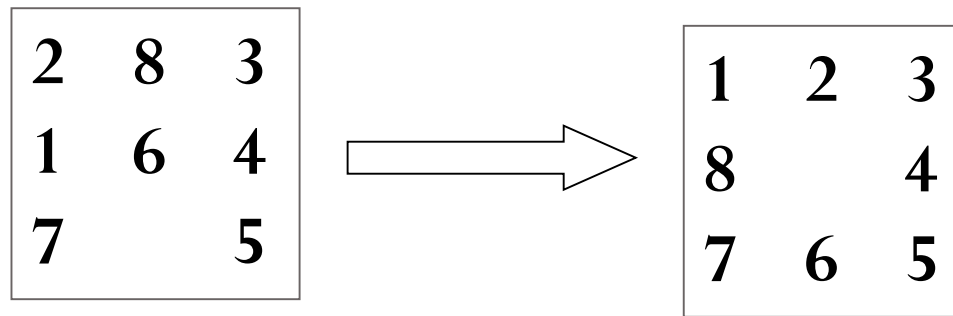
```
ADD( $m_j$ , OPEN), 标记 $m_j$ 到 $n$ 的指针;  
if  $f(n, m_k) < f(m_k)$  then  
     $f(m_k) = f(n, m_k)$ , 标记 $m_k$ 到 $n$ 的指针;  
if  $f(n, m_l) < f(m_l)$  then  
     $f(m_l) = f(n, m_l)$ , 标记 $m_l$ 到 $n$ 的指针,  
    ADD( $m_l$ , OPEN);  
OPEN中的节点按 $f$ 值从小到大排序;  
end while  
return FAIL;
```

如何得到解路径？

◆ 从目标开始，顺序访问父节点，直到初始节点。



一个A算法的例子：八数码问题



定义评价函数：

$$f(n) = g(n) + h(n)$$

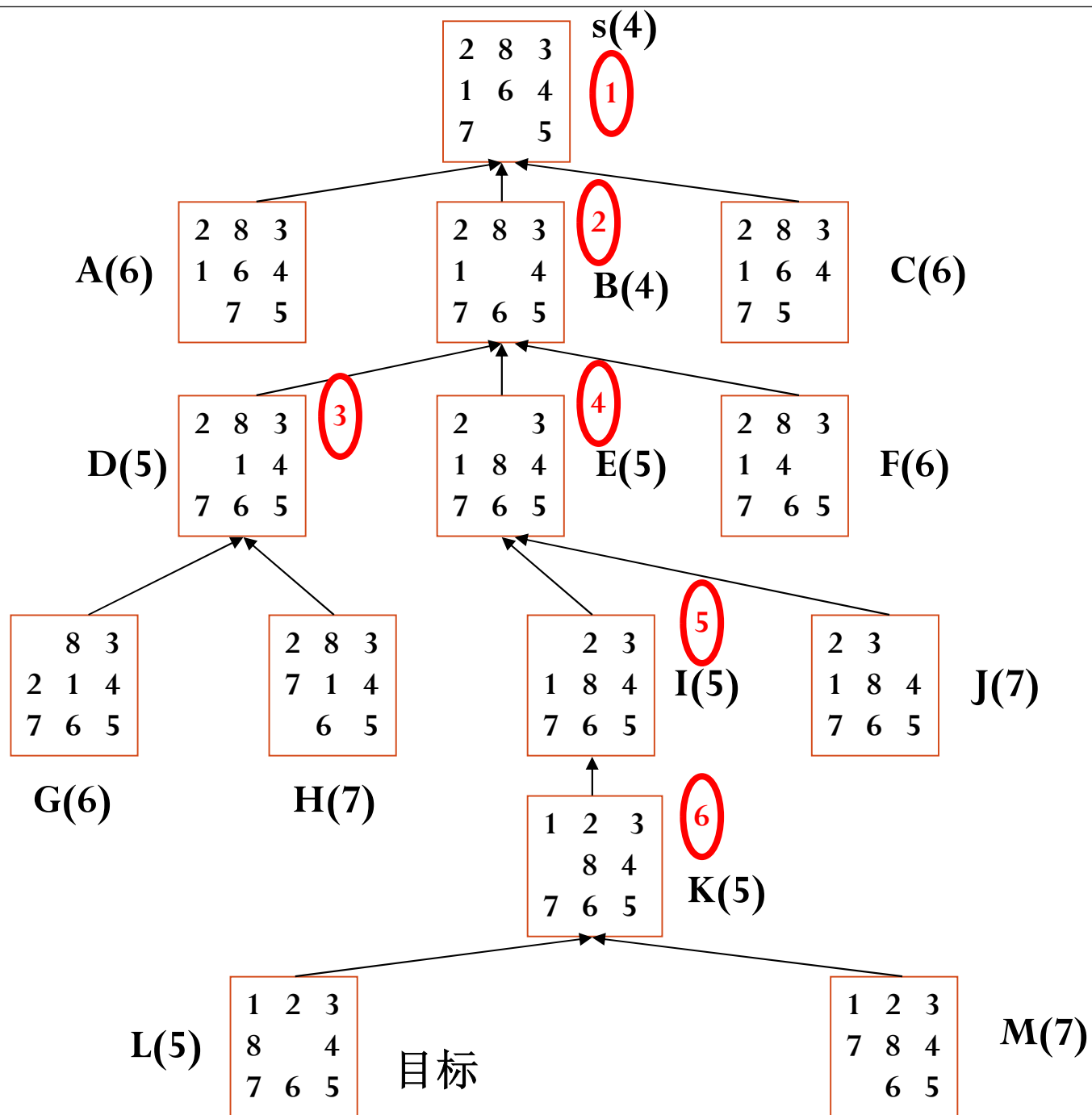
$g(n)$ 为从初始节点到当前节点的耗散值

$h(n)$ 为当前节点“不在位”的将牌数

h 计算举例

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

$$h(n) = 4$$





1.4.2 最佳图搜索算法A* (A*算法)

◆ 在A算法中，如果满足条件：

$$h(n) \leq h^*(n)$$

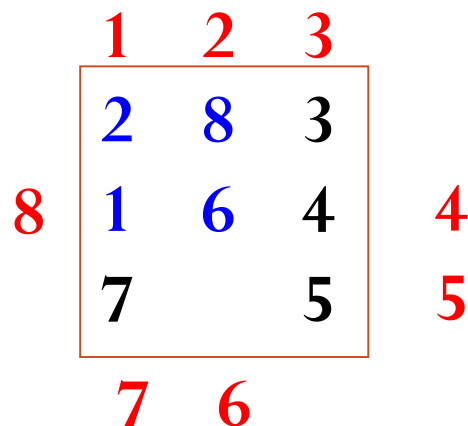
则A算法称为A*算法。



A*条件举例

◆ 8数码问题

- $h_1(n)$ = “不在位” 的将牌数
- $h_2(n)$ = 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

定义 h 函数的一般原则

- ◆ 放宽限制条件，在宽条件下，给出估计函数。

例：传教士与野人问题

- ◆思路：放宽约束条件，在宽约束条件下得到一个估计值。
- ◆假设只有乘船人数的约束没有其他约束
- ◆从左岸到右岸至少需要的摆渡次数：

$$\left\lceil \frac{M + C - 3}{2} \right\rceil \times 2 + 1 \geq M + C - 2$$



- ◆ 从右岸到左岸至少需要的摆渡次数:
- ◆ $M+C$
- ◆ 综合在一起, 所需的最少摆渡次数:
- ◆ $M+C-2b$
- ◆ 以该最小摆渡次数作为启发函数 h , 从推导可知, 该 h 满足 A^* 条件



A*算法的两个主要结论

定理 (可采纳性定理):

若存在从初始节点 s 到目标节点 t 有路径,
则A*必能找到最佳解结束。

定理：设对同一个问题定义了两个A*算法 A_1 和 A_2 ，若 A_2 比 A_1 有较多的启发信息，即对所有非目标节点有 $h_2(n) > h_1(n)$ ，则在具有一条从 s 到 t 的路径的隐含图上，搜索结束时，由 A_2 所扩展的每一个节点，也必定由 A_1 所扩展，即 A_1 扩展的节点数至少和 A_2 一样多。

简写：如果 $h_2(n) > h_1(n)$ (目标节点除外)，
则 A_1 扩展的节点数 $\geq A_2$ 扩展的节点数



◆注意：

上述定理，评价指标是“扩展的节点数”，也就是说，同一个节点无论被扩展多少次，都只计算一次。

思考题

定理（简写）：如果 $h_2(n) > h_1(n)$ (目标节点除外)，则 A_1 扩展的节点数 $\geq A_2$ 扩展的节点数

- 为什么条件不能是 $h_2(n) \geq h_1(n)$ ？什么情况下会出现问题？能否给定理再增加条件，使得定理在 $h_2(n) \geq h_1(n)$ 条件下也成立？
- **提示：**考虑那些 $f(n)=f^*(t)$ 的节点， t 为目标节点。如果不考虑这样的节点，等号可以加上。

对**h**的评价方法

◆ 平均分叉数

设共扩展了**d**层节点，共搜索了**N**个节点，
则：

$$N = \left(1 - b^{*(d+1)}\right) / (1 - b^*)$$

其中，**b***称为平均分叉数。

- ◆ **b***越小，说明**h**效果越好。
- ◆ 实验表明，**b***是一个比较稳定的常数，同一问题基本不随问题规模变化。

对 h 的评价举例

例：8数码问题，随机产生若干初始状态。

◆使用 h_1 :

$d=14$, $N=539$, $b^*=1.44$;

$d=20$, $N=7276$, $b^*=1.47$;

◆使用 h_2 :

$d=14$, $N=113$, $b^*=1.23$;

$d=20$, $N=676$, $b^*=1.27$



练习题

对于8数码问题，假设移动一个将牌的耗散值为将牌号码，请使用A*算法求解该问题。
(手工演算、编程实现)

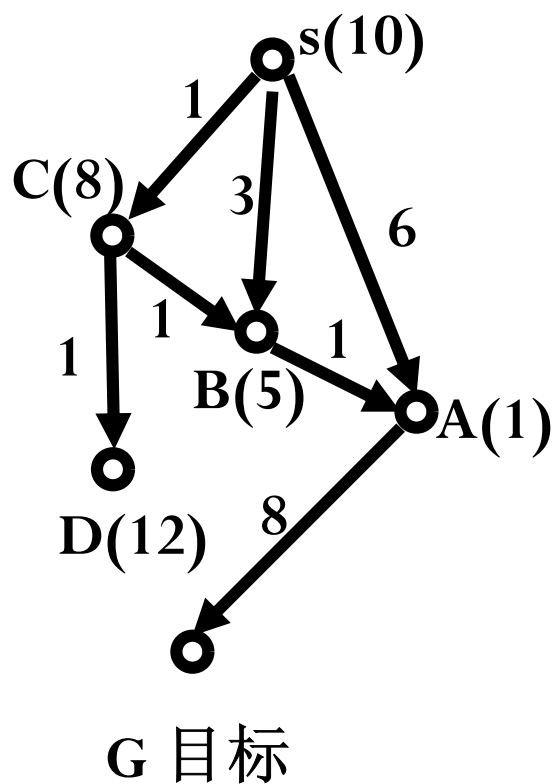


1.4.3 A*算法的改进

◆问题的提出:

因A算法对 m_1 类节点可能要重新放回到OPEN表中, 因此可能会导致多次重复扩展同一个节点, 导致搜索效率下降。

一个例子:



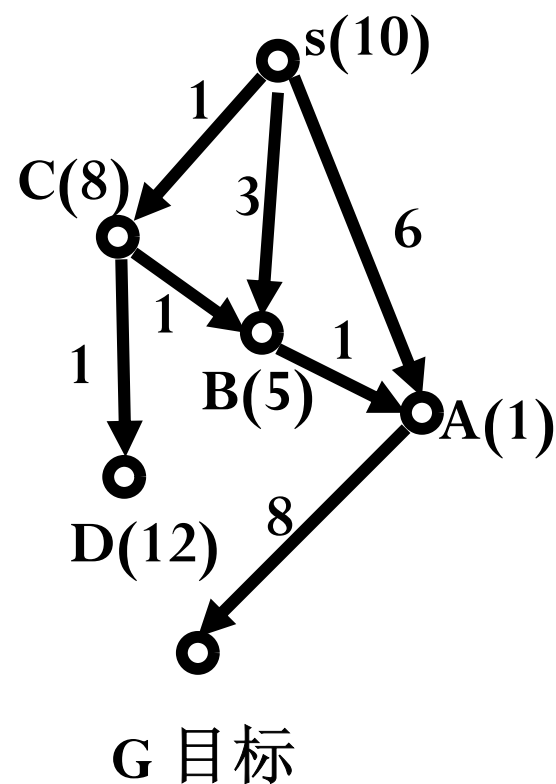
OPEN表	CLOSED表
s(10)	s(10)
<u>A(7)</u> B(8) C(9)	A(7) s(10)
<u>B(8)</u> C(9) G(14)	B(8) s(10)
<u>A(5)</u> C(9) G(14)	A(5) B(8) s(10)
C(9) G(12)	C(9) A(5) s(10)
<u>B(7)</u> G(12) D(14)	B(7) C(9) s(10)
<u>A(4)</u> G(12) D(14)	A(4) B(7) C(9) s(10)
G(11) D(14)	



出现多次扩展节点的原因

◆ 在前面的扩展中，并没有找到从初始节点到当前节点的最短路径，如节点A。

◆ 问题的突破口？





解决的途径

◆ 对 h 加以限制

- 能否对 h 增加适当的限制，使得第一次扩展一个节点时，就找到了从 s 到该节点的最短路径。

◆ 对算法加以改进

- 能否对算法加以改进，避免或减少节点的多次扩展。



改进的条件

- ◆ 可采纳性不变
- ◆ 不多扩展节点
- ◆ 不增加算法的复杂性

对 h 加以限制

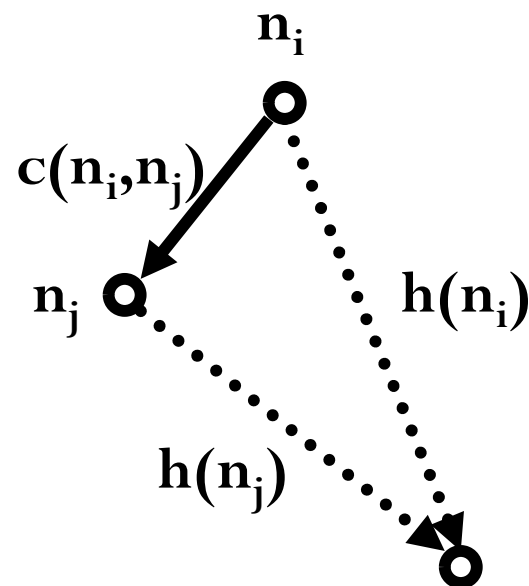
◆ 定义：一个启发函数 h ，如果对所有节点 n_i 和 n_j ，其中 n_j 是 n_i 的子节点，满足

$$\begin{cases} h(n_i) - h(n_j) \leq c(n_i, n_j) \\ h(t) = 0 \end{cases}$$

或

$$\begin{cases} h(n_i) \leq c(n_i, n_j) + h(n_j) \\ h(t) = 0 \end{cases}$$

则称 h 是单调的。



h单调的性质

◆ 定理:

若 $h(n)$ 是单调的, 则A*扩展了节点 n 之后, 就已经找到了到达节点 n 的最佳路径。

即: 当A*选 n 扩展时, 有 $g(n)=g^*(n)$ 。



◆ 思考题:

$h(n)$ 单调与A*条件的关系, 即当 $h(n)$ 满足单调条件时, 是否一定满足A*条件。

◆ 提示:

- 利用单调条件, 从目标及目标的父节点向上归纳
- 结论是满足单调的 h 一定满足A*条件。

h单调的例子

◆ 8数码问题:

h为“不在位”的将牌数

$$h(n_i) - h(n_j) = \begin{cases} 1 \\ 0 \\ -1 \end{cases} \quad (n_j \text{ 为 } n_i \text{ 的后继节点})$$

$$h(t) = 0$$

$$c(n_i, n_j) = 1$$

满足单调的条件。



对算法加以改进

◆ 一些结论：

- OPEN表上任以具有 $f(n) < f^*(s)$ 的节点定会被A*扩展。
- A*选作扩展的任一节点，定有 $f(n) \leq f^*(s)$ 。
- 当 $h(n)$ 恒等于0时，h为单调的。



改进的出发点

$$f^*(s)$$
$$\text{OPEN} = (\dots \dots \mid \dots \dots)$$

f值小于 $f^*(s)$ 的节点

f值大于等于 $f^*(s)$ 的节点

f_m : 到目前为止已扩展节点的最大f值, 用 f_m 代替 $f^*(s)$

修正的A算法

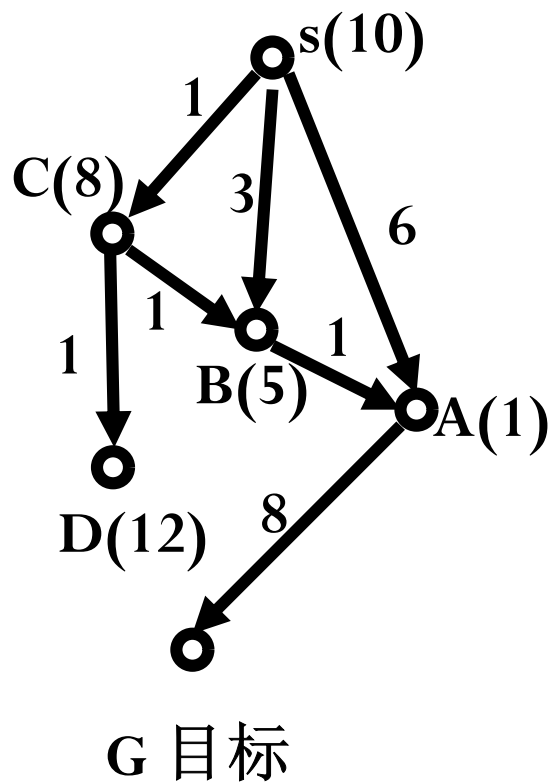


Modified-A-algorithm (s) s为初始节点
OPEN=(s), CLOSED=(), $f(s)=g(s)+h(s)$, $f_m=0$;
while OPEN不空 do:
 begin
 $NEST = \{n_i \mid f(n_i) < f_m, n_i \in OPEN\}$
 if $NEST \neq ()$ then $n = NEST$ 中g最小的节点
 else $n = FIRST(OPEN)$, $f_m = f(n)$;
 if GOAL(n) THEN return n;
 REMOVE(n, OPEN), ADD(n, CLOSED);
 EXPAND(n) $\rightarrow \{m_i\}$,
 计算 $f(n, m_i) = g(n, m_i) + h(m_i)$;



```
ADD( $m_j$ , OPEN), 标记 $m_j$ 到 $n$ 的指针;  
if  $f(n, m_k) < f(m_k)$  then  
     $f(m_k) = f(n, m_k)$ , 标记 $m_k$ 到 $n$ 的指针;  
if  $f(n, m_l) < f(m_l)$  then  
     $f(m_l) = f(n, m_l)$ , 标记 $m_l$ 到 $n$ 的指针,  
    ADD( $m_l$ , OPEN);  
OPEN中的节点按 $f$ 值从小到大排序;  
end while  
return FAIL;
```

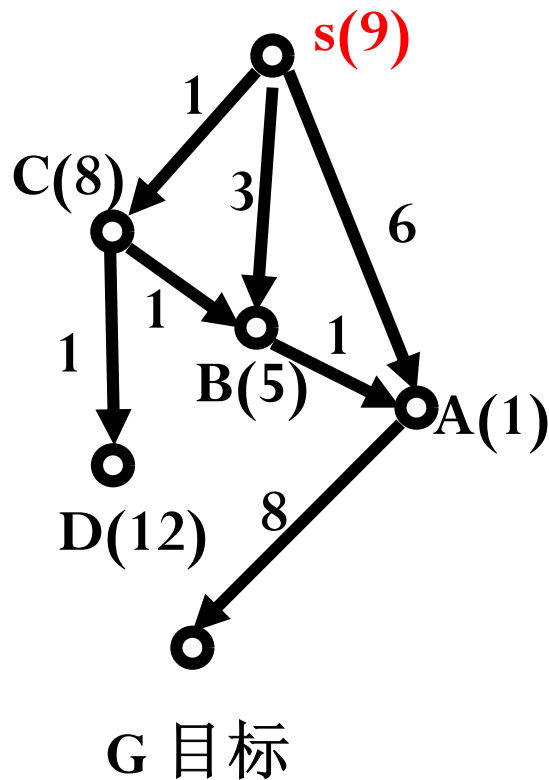
前面的例子：



OPEN表	CLOSED表	f_m
s(0+10)	s(0+10)	10
A(6+1) B(3+5) <u>C(1+8)</u>	s(0+10) C(1+8)	10
A(6+1) <u>B(2+5)</u> D(2+12)	s(0+10) C(1+8) B(2+5)	10
<u>A(3+1)</u> D(2+12)	s(0+10) C(1+8) B(2+5) A(3+1)	10
G(11+0) D(2+12)		

说明：蓝颜色表示在nest中的节点

前面的例子中， $h(s)$ 修改为9:



OPEN表	CLOSED表	f_m
$s(0+9)$	$s(0+9)$	9
$\underline{A(6+1)} \quad \underline{B(3+5)} \quad C(1+8)$	$s(0+10) \quad B(3+5)$	9
$\underline{A(4+1)} \quad C(1+8)$	$s(0+10) \quad B(3+5) \quad A(4+1)$	9
$\underline{C(1+8)} \quad G(12+0)$	$s(0+10) \quad A(4+1) \quad C(1+8)$	9
$\underline{B(2+5)} \quad G(12+0) \quad D(2+12)$	$s(0+10) \quad C(1+8) \quad B(2+5)$	9
$\underline{A(3+1)} \quad G(12+0) \quad D(2+12)$	$s(0+10) \quad C(1+8) \quad B(2+5) \quad A(3+1)$	9
$G(11+0) \quad D(2+12)$		

说明：蓝颜色表示在nest中的节点

思考题

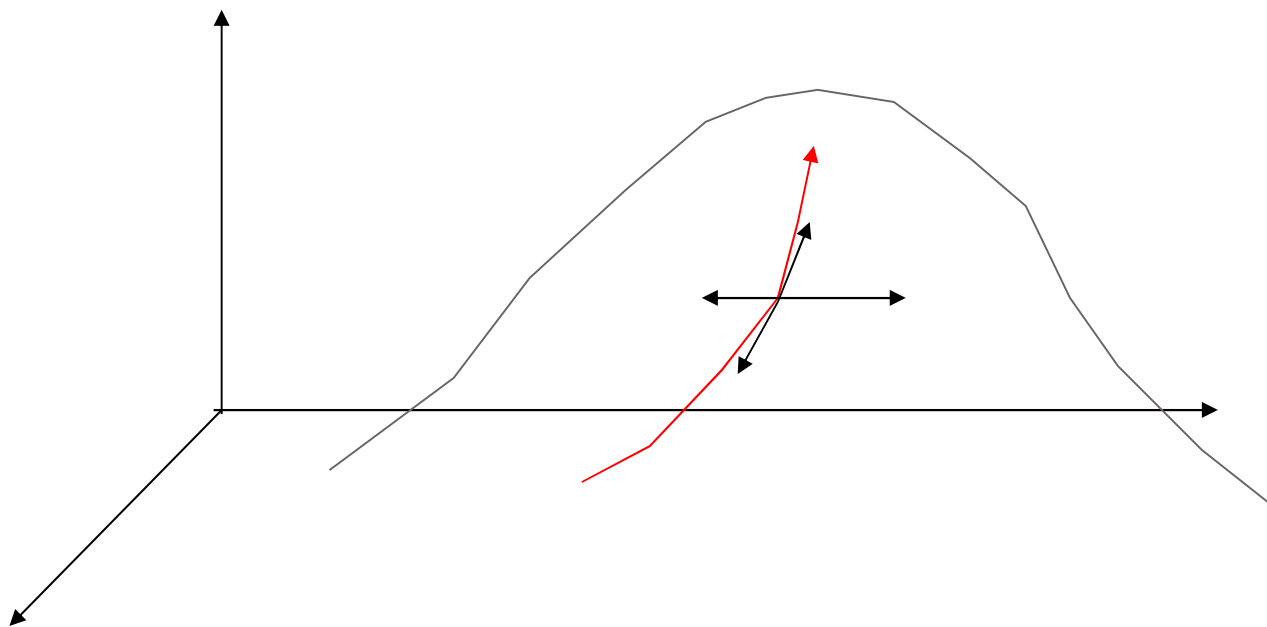
- ◆ 宽度优先算法与A*算法是什么关系？
 - 提示：单位耗散时 $h=0$ ，A*退化为宽度优先

- ◆ 如何修改A*算法，当问题存在多于 n 个解时，算法可以求解前 n 个最好的解。
 - 提示：对于每个节点，当存在多于 n 条路径时，都要保留前 n 条最短距离的路径。



1.5 其他的搜索算法

◆爬山法（局部搜索算法）





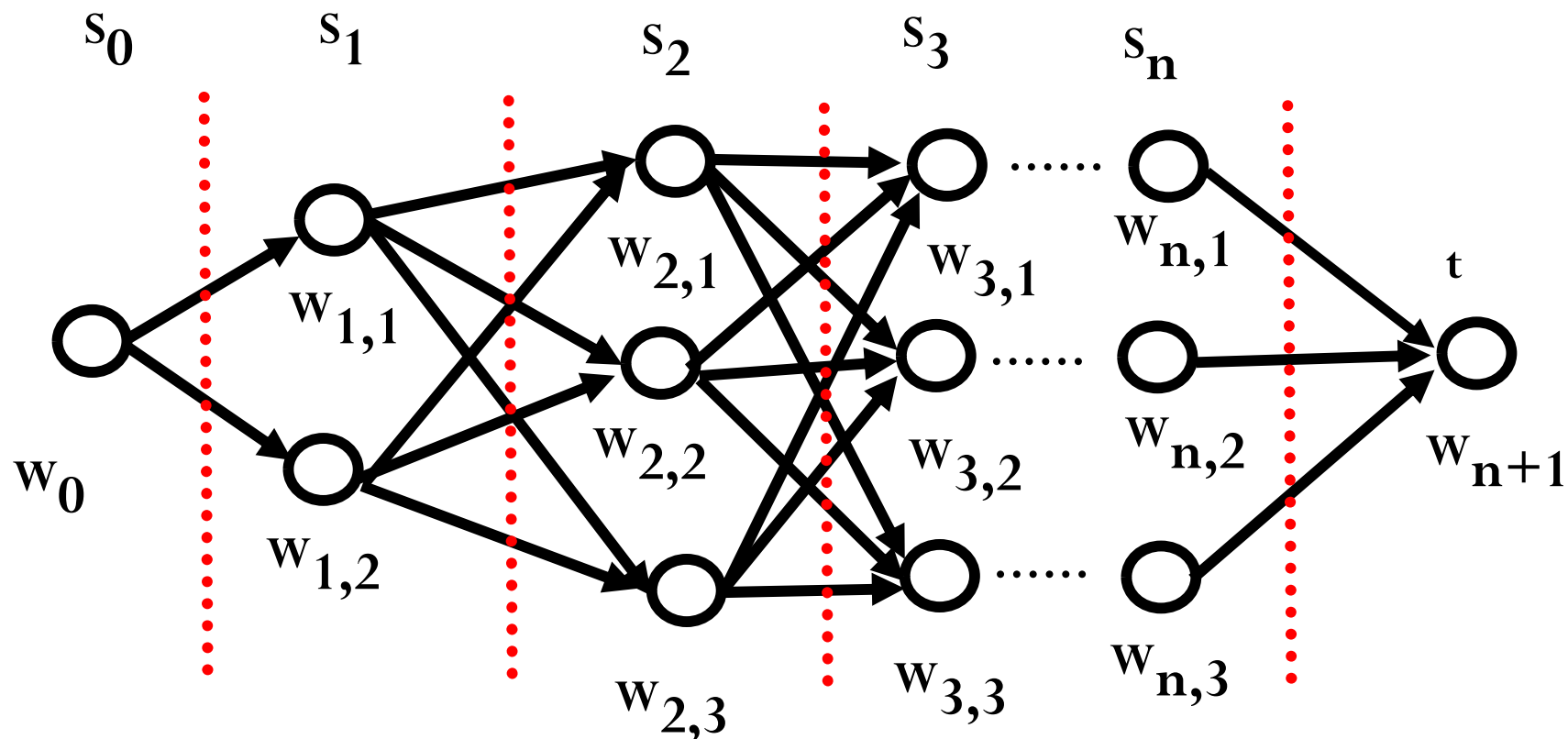
其他的搜索算法（续1）

- ◆ 随机搜索算法

- ◆ 动态规划算法

如果对于任何 n ，当 $h(n)=0$ 时， A^* 算法就成为了动态规划算法。

动态规划：viterbi算法





$$Q(W_{i,j}) = \begin{cases} \min_k (Q(W_{i-1,k}) + D(W_{i-1,k}, W_{i,j})) & i \neq 0 \\ 0 & i = 0 \end{cases}$$

其中： $Q(W_{ij})$ 表示起点到点 W_{ij} 的最佳路径值
 $D(W_{i-1,j}, W_{i,k})$ 表示 $W_{i-1,j}$ 到 $W_{i,k}$ 的距离



1.6 搜索算法实用举例

◆ 汉字识别后处理





汉字识别后处理

◆ 一个例子

我钱线载哦栽哉裁劣绥
优仍们仿伦奶砧犯扔妨
要耍密穷安壁驻努窑垂
扳报叔嵌奴振技寂叙蔽
奋夯杏蚕香脊秀吞吝番
精猜指洁括治捐活治桔
种神衬祥科钟拌样拎补

汉字识别后处理



$$P(S | O) = P(S)P(O | S)/P(O)$$

$$P(S) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1})$$

二元语法时:
$$P(S) = \prod_{i=1}^n P(w_i | w_{i-1})$$

$P(O)$ 为常量

$P(O | S)$ 用识别信度代替

问题变为求
$$\prod_{i=1}^n P(w_i | w_{i-1})CF(w_i)$$
 最大



◆ 概率的计算（语言模型）

$$P(w_i | w_{i-1}) = \frac{w_i w_{i-1} \text{同现的次数}}{w_{i-1} \text{出现的次数}}$$

◆ 平滑

解决 $P(w_i | w_{i-1})$ 可能为0的问题，有很多种方法，其中一种简单的方法：

$$\lambda P(w_i | w_{i-1}) + (1 - \lambda) P(w_i) \Rightarrow P(w_i | w_{i-1})$$



拼音输入法

◆ jī qì xué xī jī qì yīng yòng





拼音输入法

jì qī xué xì jì qī yīng yǒng

及 期 学 系 及 期 应 勇

计 器 雪 习 计 器 英 用

机 其 薛 西 机 其 营 永

... ..



拼音输入法

jì qī xué xì jì qī yīng yǒng

及 期 学 系 及 期 应 勇

计 器 雪 习 计 器 英 用

机 其 薛 西 机 其 营 永

... ..



拼音输入法

$$P(S | O) = P(S)P(O | S)/P(O)$$

$P(O)$ 为常量

$$P(O|S) \approx 1$$

$$P(S) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1})$$

$$\text{求 } P(S) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1}) \text{ 最大}$$



拼音输入法

$$P(S) = \prod_{i=1}^n P(w_i \mid w_1 \dots w_{i-1})$$

二元语法时: $P(S) = \prod_{i=1}^n P(w_i \mid w_{i-1})$

求 $\max \left(\prod_{i=1}^n P(w_i \mid w_{i-1}) \right)$ 所对应的句子

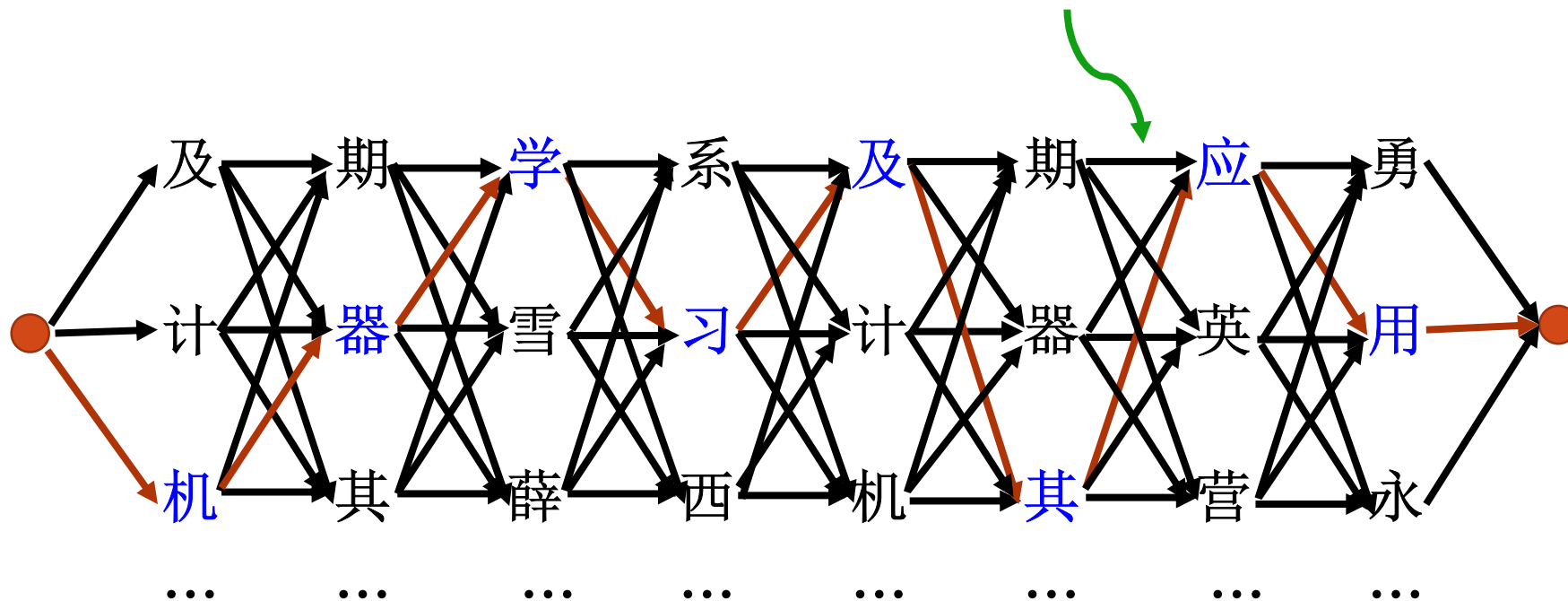
等价于:

求 $\min \left(- \sum_{i=1}^n \log(P(w_i \mid w_{i-1})) \right)$ 所对应的句子



拼音输入法

$$-\log(P(w_i \mid w_{i-1}))$$



$$\text{求} - \sum_{i=1}^n \log(P(w_i \mid w_{i-1})) \text{ 最小}$$

编程作业1

◆编程实现拼音输入法，具体见网络学堂。



小结

◆ 盲目搜索

- 深度优先
- 宽度优先

◆ 启发式搜索

- A算法
- A*算法
- 改进的A*算法

◆ 动态规划

- viterbi算法