

## 1.4 启发式图搜索

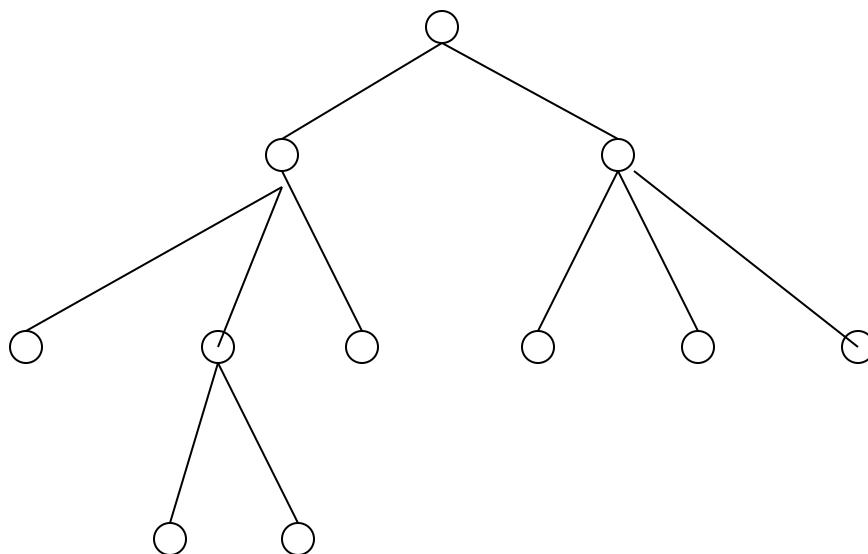
- 利用知识来引导搜索，达到减少搜索范围，降低问题复杂度的目的。
- 启发信息的强度
  - 强：降低搜索工作量，但可能导致找不到最优解
  - 弱：一般导致工作量加大，极限情况下变为盲目搜索，但有可能找到最优解

# 希望：

- 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。

# 基本思想

- 定义一个评价函数 $f$ ，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。



# 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的最小耗散值
- $h^*(n)$ : 从n到g的最小耗散值
- $f^*(n)=g^*(n)+h^*(n)$ : 从s经过n到g的最小耗散值
- $g(n)$ 、 $h(n)$ 、 $f(n)$ 分别是 $g^*(n)$ 、 $h^*(n)$ 、 $f^*(n)$ 的估计值

# A算法

- 1, OPEN:=(s),  $f(s):=g(s)+h(s)$ ;
- 2, LOOP: IF OPEN=( ) THEN EXIT(FAIL);
- 3,  $n:=\text{FIRST}(\text{OPEN})$ ;
- 4, IF GOAL(n) THEN EXIT(SUCCESS);
- 5, REMOVE(n, OPEN), ADD(n, CLOSED);
- 6, EXPAND(n)  $\rightarrow \{m_i\}$ ,  
    计算  $f(n, m_i):=g(n, m_i)+h(m_i)$ ;

# A算法（续）

ADD( $m_j$ , OPEN), 标记 $m_j$ 到 $n$ 的指针;

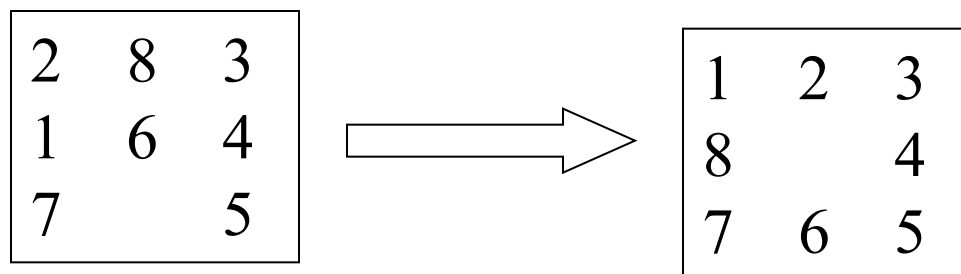
IF  $f(n, m_k) < f(m_k)$  THEN  $f(m_k) := f(n, m_k)$ ,  
标记 $m_k$ 到 $n$ 的指针;

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);

7, OPEN中的节点按 $f$ 值从小到大排序;

8, GO LOOP;

# 一个A算法的例子



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

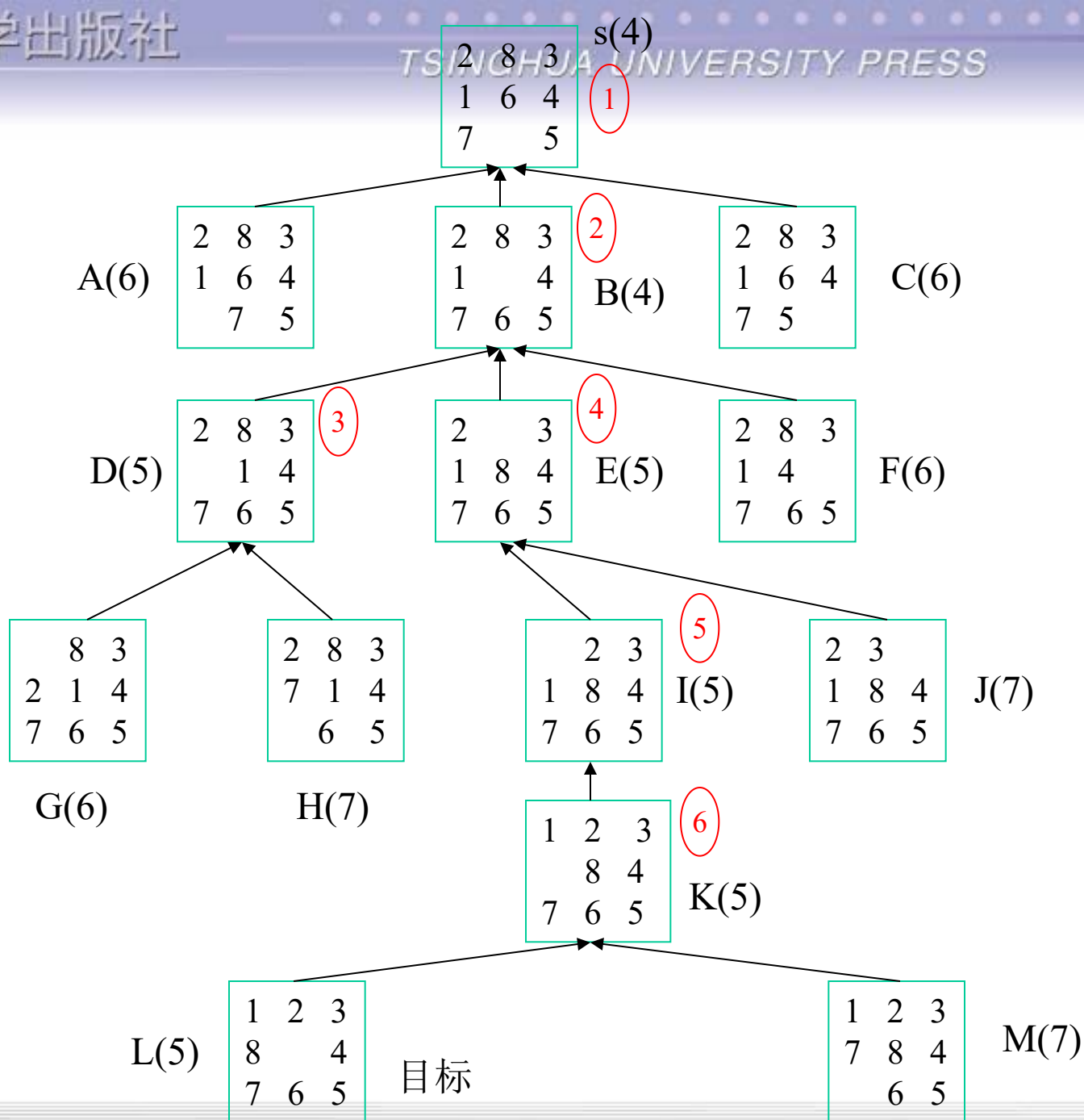
$h(n)$ 为当前节点“不在位”的将牌数



# h计算举例

	<b>1</b>	<b>2</b>	<b>3</b>	
	2	8	3	
<b>8</b>	1	6	4	<b>4</b>
	7		5	<b>5</b>
	<b>7</b>	<b>6</b>		

$$h(n) = 4$$



# 最佳图搜索算法A\* (A\*算法)

- 在A算法中, 如果满足条件:

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

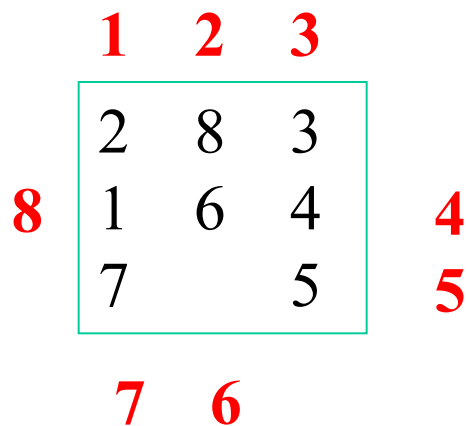
则A算法称为A\*算法。

# A\*条件举例

- 8数码问题

- $h1(n)$  = “不在位” 的将牌数

- $h2(n)$  = 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

# A\*算法的性质

- 当问题有解时，A\*算法一定能找到最佳路径。
- 极端情况下，若 $h(n) \equiv 0$ ，一定能找到最佳路径，此时，若 $g \equiv d$ ，则A\*算法等同于宽度优先算法。
- 几个等式：

$$f^*(s) = f^*(t) = h^*(s) = g^*(t) = f^*(n)$$

其中s是初始节点，t是目标节点，n是s到t的最佳路径上的节点。

# A\*算法的性质（续1）

定理1.1:

对有限图，如果从初始节点s到目标节点t有路径存在，则**算法A一定成功结束**。

## A\*算法的性质（续2）

引理1.1：

对无限图，若有从初始节点s到目标节点t的路径，则A\*不结束时，在OPEN表中即使最小的一个f值也将增到任意大，或有 $f(n) > f^*(s)$ 。

# A\*算法的性质（续3）

引理1.2:

A\*结束前，OPEN表中必存在 $f(n) \leq f^*(s)$ 的结点（n是在最佳路径上的结点）。

存在一个节点n，n在最佳路径上。

$$\begin{aligned} f(n) &= g(n) + h(n) \\ &= g^*(n) + h(n) \\ &\leq g^*(n) + h^*(n) \\ &= f^*(n) \\ &= f^*(s) \end{aligned}$$



# A\*算法的性质（续3）

定理1.2:

对无限图，若从初始节点s到目标节点t有路径存在，则A\*一定成功结束。

引理1.1: A\*如果不结束，则OPEN中所有的n有  
 $f(n) > f^*(s)$

引理1.2: 在A\*结束前，必存在节点n，使得  
 $f(n) \leq f^*(s)$

所以，如果A\*不结束，将导致矛盾。

## A\*算法的性质（续4）

推论1.1:

OPEN表上任一具有 $f(n) < f^*(s)$ 的节点 $n$ ,  
最终都将被A\*选作扩展的节点。

由定理1.2, 知A\*一定结束, 由A\*的结束条件, OPEN表中 $f(t)$ 最小时才结束。而

$$f(t) \geq f^*(t) = f^*(s)$$

所以 $f(n) < f^*(s)$ 的 $n$ , 均被扩展。得证。

# A\*算法的性质（续5）

定理1.3 (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。

# 可采纳性的证明

- 由定理1.1、1.2知A\*一定找到一条路径结束
- 设找到的路径 $s \rightarrow t$ 不是最佳的（ $t$ 为目标）  
则：  $f(t) = g(t) > f^*(s)$
- 由引理1.2知结束前OPEN中存在 $f(n) \leq f^*(s)$ 的节点 $n$ ，所以
$$f(n) \leq f^*(s) < f(t)$$
- 因此A\*应选择 $n$ 扩展，而不是 $t$ 。与假设A\*选择 $t$ 结束矛盾。得证。
- 注意：A\*的结束条件

# A\*算法的性质（续6）

推论1.2:

A\*选作扩展的任一节点 $n$ ，有 $f(n) \leq f^*(s)$ 。

- 由引理1.2知在A\*结束前，OPEN中存在节点 $n'$ ， $f(n') \leq f^*(s)$
- 设此时A\*选择 $n$ 扩展。
- 如果 $n = n'$ ，则 $f(n) \leq f^*(s)$ ，得证。
- 如果 $n \neq n'$ ，由于A\*选择 $n$ 扩展，而不是 $n'$ ，所以有 $f(n) \leq f(n') \leq f^*(s)$ 。得证。

## A\*算法的性质（续7）

定理1.4： 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

## A\*算法的性质（续7）

- 注意：

在定理1.4中，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。

# 定理1.4的证明

- 使用数学归纳法，对节点的深度进行归纳
- （1）当 $d(n)=0$ 时，即只有一个节点，显然定理成立。
- （2）设 $d(n)\leq k$ 时定理成立。（归纳假设）
- （3）当 $d(n)=k+1$ 时，用反证法。
- 设存在一个深度为 $k+1$ 的节点 $n$ ，被 $A_2$ 扩展，但没有被 $A_1$ 扩展。而由假设， $A_1$ 扩展了 $n$ 的父节点，即 $n$ 已经被生成了。因此当 $A_1$ 结束时， $n$ 将被保留在OPEN中。



# 定理1.4的证明（续1）

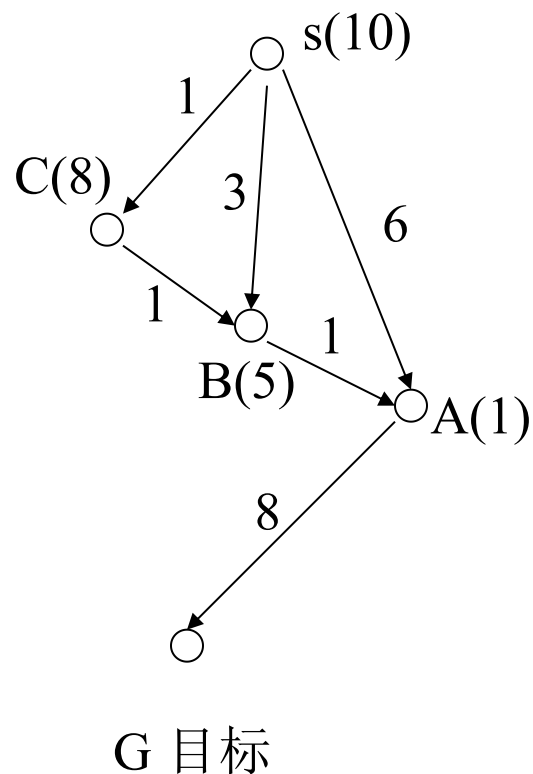
- 所以有：  $f_1(n) \geq f^*(s)$
- 即：  $g_1(n) + h_1(n) \geq f^*(s)$
- 所以：  $h_1(n) \geq f^*(s) - g_1(n)$
- 另一方面，由于  $A_2$  扩展了  $n$ ，有  $f_2(n) \leq f^*(s)$
- 即：  $h_2(n) \leq f^*(s) - g_2(n)$  (A)
- 由于  $d(n)=k$  时，  $A_2$  扩展的节点  $A_1$  一定扩展，有  
 $g_1(n) \leq g_2(n)$  （因为  $A_2$  的路  $A_1$  均走到了）
- 所以：  $h_1(n) \geq f^*(s) - g_1(n) \geq f^*(s) - g_2(n)$  (B)
- 比较A、B两式，有  $h_1(n) \geq h_2(n)$ ，与定理条件矛盾。故定理得证。

# A\*算法的改进

- 问题的提出:

因A算法第6步对 $m_1$ 类节点可能要重新放回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

s(10)

A(7) B(8) C(9)

B(8) C(9) G(14)

A(5) C(9) G(14)

C(9) G(12)

B(7) G(12)

A(4) G(12)

G(11)

CLOSED表

s(10)

A(7) s(10)

B(8) s(10)

A(5) B(8) s(10)

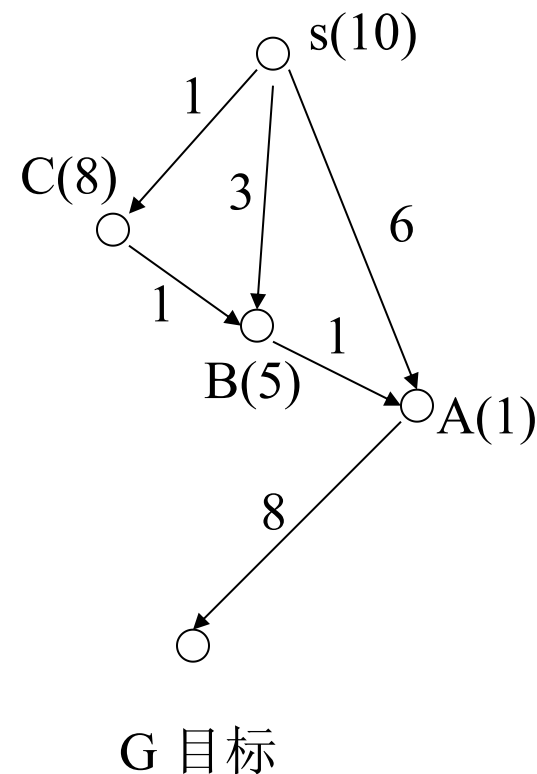
C(9) A(5) s(10)

B(7) C(9) s(10)

A(4) B(7) C(9) s(10)

# 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。



# 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

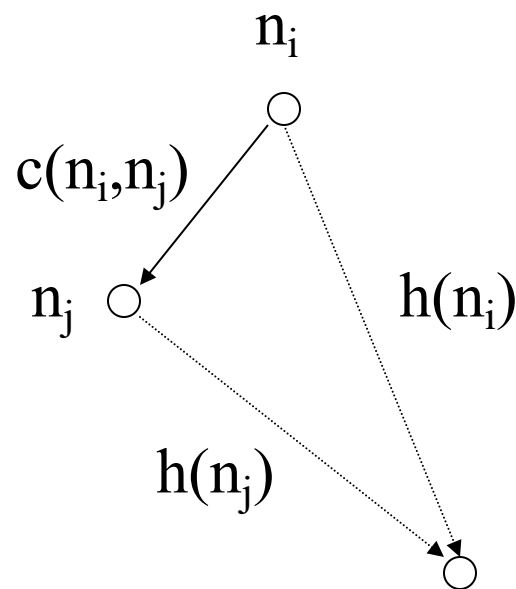
- 定义：一个启发函数h，如果对所有节点  $n_i$  和  $n_j$ ，其中  $n_j$  是  $n_i$  的子节点，满足

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

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1.5:  
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 $g(n)=g^*(n)$ 。



# 定理1.5的证明

- 设 $n$ 是 $A^*$ 扩展的任一节点。当 $n=s$ 时，定理显然成立。下面考察 $n \neq s$ 的情况。
- 设 $P=(n_0=s, n_1, n_2, \dots, n_k=n)$ 是 $s$ 到 $n$ 的最佳路径
- $P$ 中一定有节点在CLOSED中，设 $P$ 中最后一个出现在CLOSED中的节点为 $n_j$ ，则 $n_{j+1}$ 在OPEN中。

# 定理1.5的证明（续1）

- 由单调限制条件，对P中任意节点 $n_i$ 有：

$$h(n_i) \leq C(n_i, n_{i+1}) + h(n_{i+1})$$

$$g^*(n_i) + h(n_i) \leq g^*(n_i) + C(n_i, n_{i+1}) + h(n_{i+1})$$

- 由于 $n_i$ 、 $n_{i+1}$ 在最佳路径上，所以：

$$g^*(n_{i+1}) = g^*(n_i) + C(n_i, n_{i+1})$$

- 带入上式有：

$$g^*(n_i) + h(n_i) \leq g^*(n_{i+1}) + h(n_{i+1})$$

- 从 $i=j$ 到 $i=k-1$ 应用上不等式，有：

$$g^*(n_{j+1}) + h(n_{j+1}) \leq g^*(n_k) + h(n_k)$$

- 即： $f(n_{j+1}) \leq g^*(n) + h(n)$

注意： $(n_j$ 在CLOSED中， $n_{j+1}$ 在OPEN中)

## 定理1.5的证明（续2）

- 重写上式： $f(n_{j+1}) \leq g^*(n) + h(n)$
- 另一方面， $A^*$ 选 $n$ 扩展，必有：

$$f(n) = g(n) + h(n) \leq f(n_{j+1})$$

- 比较两式，有：

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

- 但已知 $g^*(n)$ 是最佳路径的耗散值，所以只有： $g(n) = g^*(n)$ 。得证。

## h单调的性质（续）

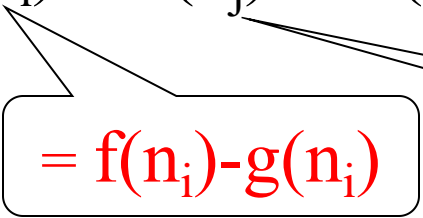
- 定理1.6:

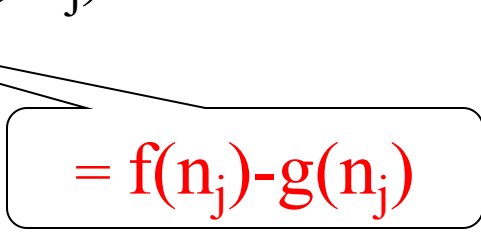
若 $h(n)$ 是单调的，则由 $A^*$ 所扩展的节点序列其 $f$ 值是非递减的。即 $f(n_i) \leq f(n_j)$ 。

# 定理1.6的证明

- 由单调限制条件，有：

$$h(n_i) - h(n_j) \leq C(n_i, n_j)$$


$$= f(n_i) - g(n_i)$$


$$= f(n_j) - g(n_j)$$

$$f(n_i) - g(n_i) - f(n_j) + g(n_j) \leq C(n_i, n_j)$$


$$= g(n_i) + C(n_i, n_j)$$

$$f(n_i) - g(n_i) - f(n_j) + g(n_i) + C(n_i, n_j) \leq C(n_i, n_j)$$

$$f(n_i) - f(n_j) \leq 0, \text{ 得证。}$$

# h单调的例子

- 8数码问题：
  - h为“不在位”的将牌数（ $n_j$ 是 $n_i$ 的子节点）

$$h(n_i) - h(n_j) = \begin{cases} 1 & (\text{不在位} \rightarrow \text{在位}) \\ 0 & (\text{不在位} \rightarrow \text{不在位}) \\ -1 & (\text{在位} \rightarrow \text{不在位}) \end{cases}$$

$$h(t) = 0$$

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

满足单调的条件。

# 对算法加以改进

- 一些结论：
  - OPEN表上任一具有 $f(n) < f^*(s)$ 的节点定会被扩展。（推论1.1）
  - A\*选作扩展的任一节点，定有 $f(n) \leq f^*(s)$ 。（推论1.2）

# 改进的出发点

 $f^*(s)$ 

OPEN = ( ... .. )

f值小于 $f^*(s)$ 的节点  
(NEST)

f值大于等于 $f^*(s)$ 的节点

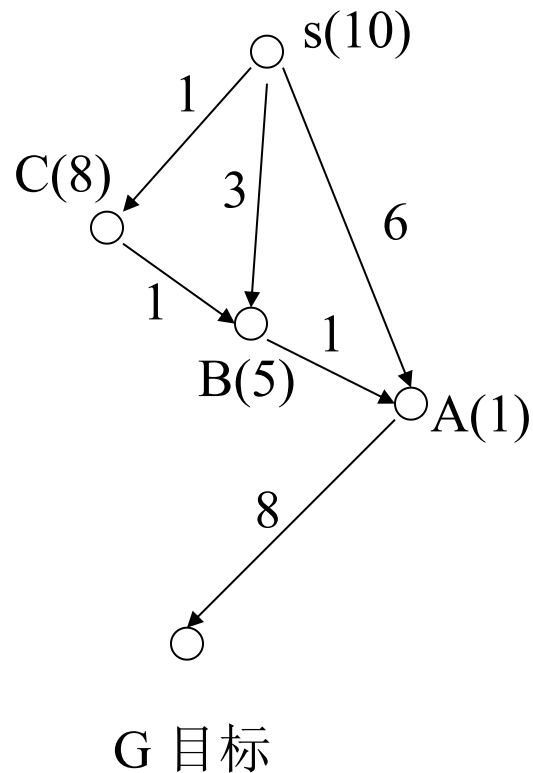
$f_m$ : 到目前为止已扩展节点的最大f值, 用 $f_m$ 代替 $f^*(s)$



# 修正过程A

- 1, OPEN:=(s),  $f(s)=g(s)+h(s)$ ,  $f_m:=0$ ;
- 2, LOOP: IF OPEN=( ) THEN EXIT(FAIL);
- 3, NEST:={ $n_i | f(n_i) < f_m$ }
- IF NEST  $\neq$  ( ) THEN  $n:=$ NEST中 $g$ 最小的节点(?)
- ELSE  $n:=$ FIRST(OPEN),  $f_m:=f(n)$ ;
- 4, ..., 8: 同过程A。

前面的例子：



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