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，但可能可以找到最优解

启发式搜索概述

- 设计一个与问题相关的估价函数，用于评价各节点的重要程度
- 按照节点的重要性次序，亦即估价函数从小到大的顺序扩展节点
- 估价函数的形式：$f(n)=\alpha g(n)+\beta h(n)$

  $g(n)$－从S_0到节点n已付出的代价；

  $h(n)$－n到S_g最优路径的估计代价；

启发式搜索概述（续）

- g(x)比例越大，越倾向广度优先搜索，完备性较好而效率可能受影响；
- h(x)比例越大，越倾向深度优先搜索，效率可能较好；
- 加权α,β调整比例,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。
- 不同的启发式函数构成了算法的不同性质

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

- g*(n)：从S0到n的最短路径的耗散值
- h*(n)：从n到Sg的最短路径的耗散值
- f*(n)=g*(n)+h*(n)：从S0经过n到Sg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
- 恒有： $g(n) \geq g^*(n)$ 且 g(n)不增

A算法

1, OPEN:=(s), f(s):=g(s)+h(s);

2, LOOP: IF OPEN=( ) THEN EXIT(FAIL);

3, n:=FIRST(OPEN);

4, IF GOAL(n) THEN EXIT(SUCCESS);

5, REMOVE(n,OPEN), ADD(n,CLOSED);

6, EXPAND(n) $\rightarrow\{m_i\}$,

计算$f(n,m_i):=g(n,m_i)+h(m_i)$;

$f(n,m_i)$:经过节点n的m_i的评价函数

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

$f(m_l):=f(n,m_l)$，

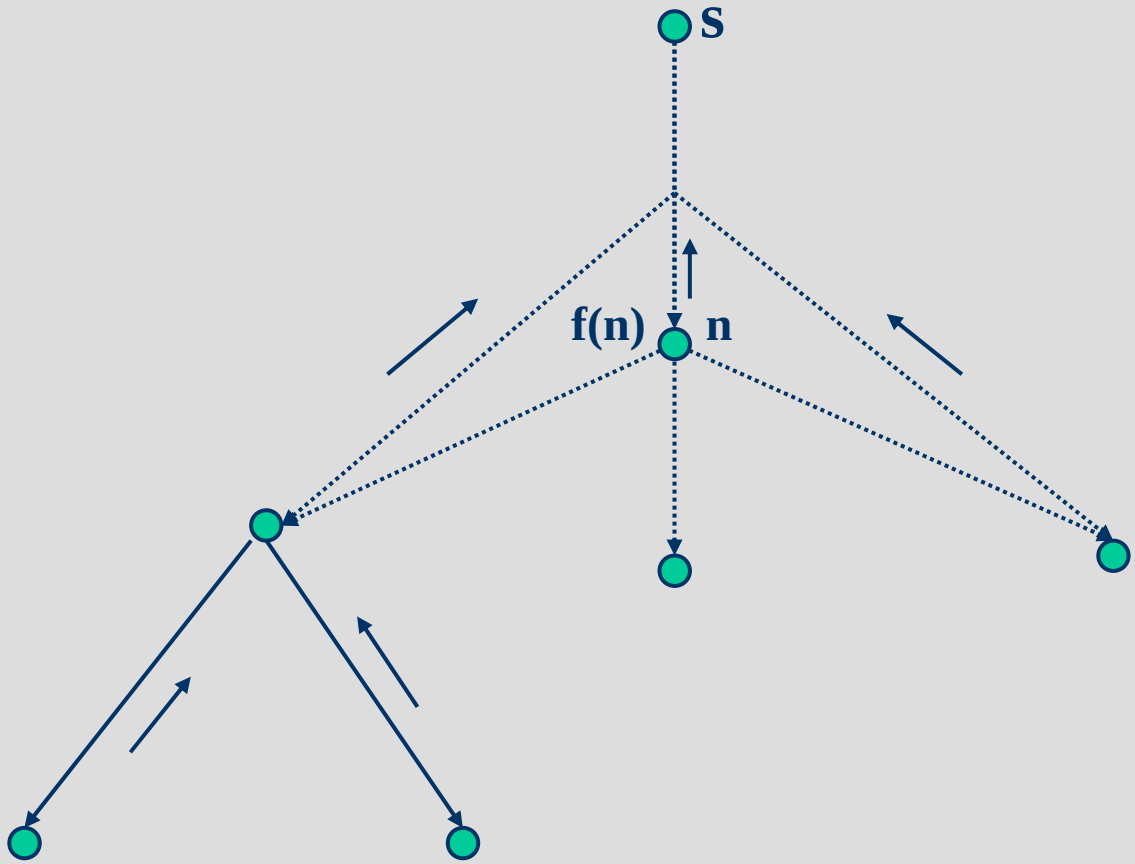
标记m_l到n的指针，

- m_j：扩展的新节点
- m_k：已生成未被扩展的节点
- m_1：已生成并被扩展的节点

ADD(m_l,OPEN)；（不变后代指针，m_l从closed重进open表）

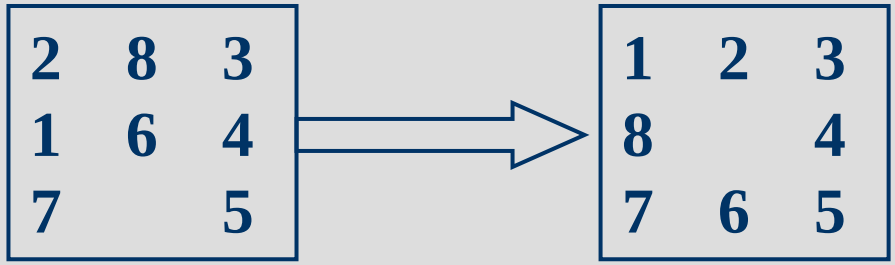
7，OPEN表节点按f值从小到大排序；

8，GO LOOP；



对有限图，如果从初始点到目标点有路径存在，则算法A一定成功结束（完备性）

例：九宫重排问题



定义评价函数1：

$f_1(n) = g(n) + h_1(n)$

g(n)为从初始节点到当前节点的耗散值（深度）

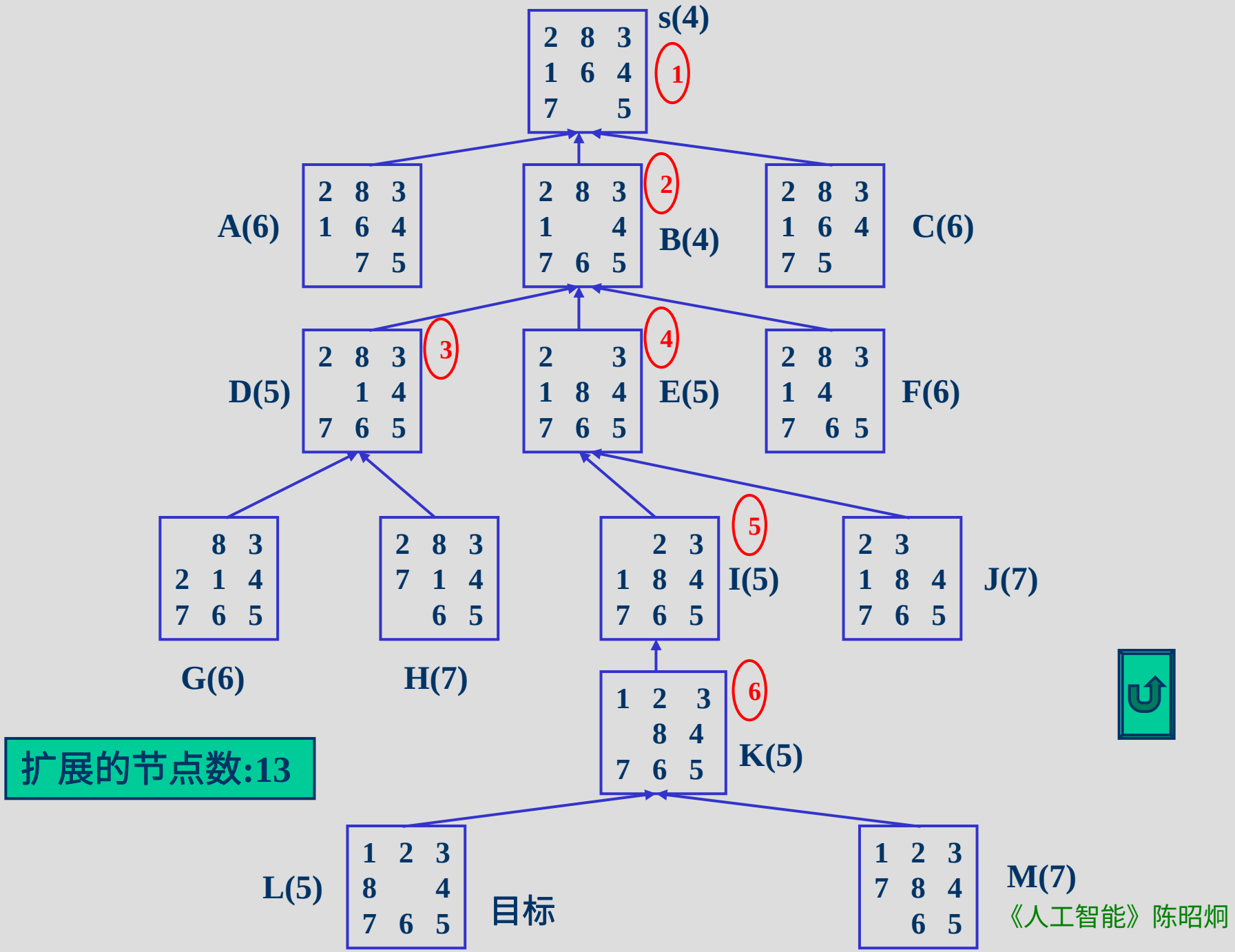
$h_1(n)$为当前节点“不在位”的将牌数

1 2 3

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

7 6

$h_1(n) = 4$


s(4)
2 8 3
1 6 4
7 5
1
A(6)
2 8 3
1 6 4
7 5
2
2 8 3
1 4
7 6 5
B(4)
2 8 3
1 6 4
7 5
C(6)
D(5)
2 8 3
1 4
7 6 5
3
2 3
1 8 4
7 6 5
4
E(5)
2 8 3
1 4
7 6 5
F(6)
8 3
2 1 4
7 6 5
G(6)
2 8 3
7 1 4
6 5
H(7)
2 3
1 8 4
7 6 5
5
I(5)
2 3
1 8 4
7 6 5
J(7)
1 2 3
8 4
7 6 5
6
K(5)
L(5)
1 2 3
8 4
7 6 5
目标
1 2 3
7 8 4
6 5
M(7)
扩展的节点数:13
《人工智能》陈昭炯

定义评价函数2：

$f_2(n) = g(n) + h_1(n)$

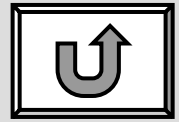
g(n)为从初始节点到当前节点的耗散值（深度）

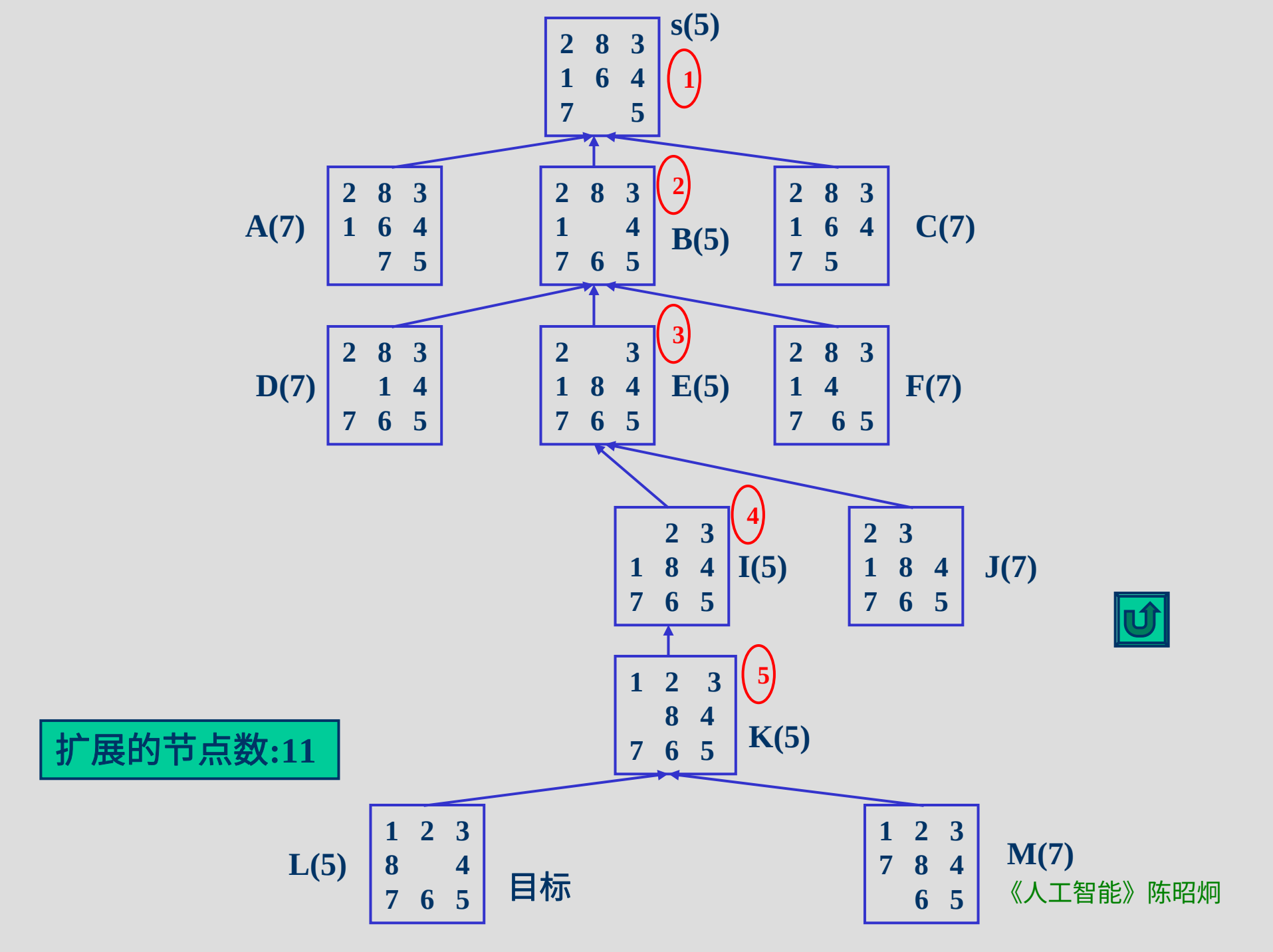
$h_2(n)$为数字移到目标处的距离总和

2	8	3
1	6	4
7		5

$h_2(n)=1+1+1+2=5$

1	2	3
8		4
7	6	5

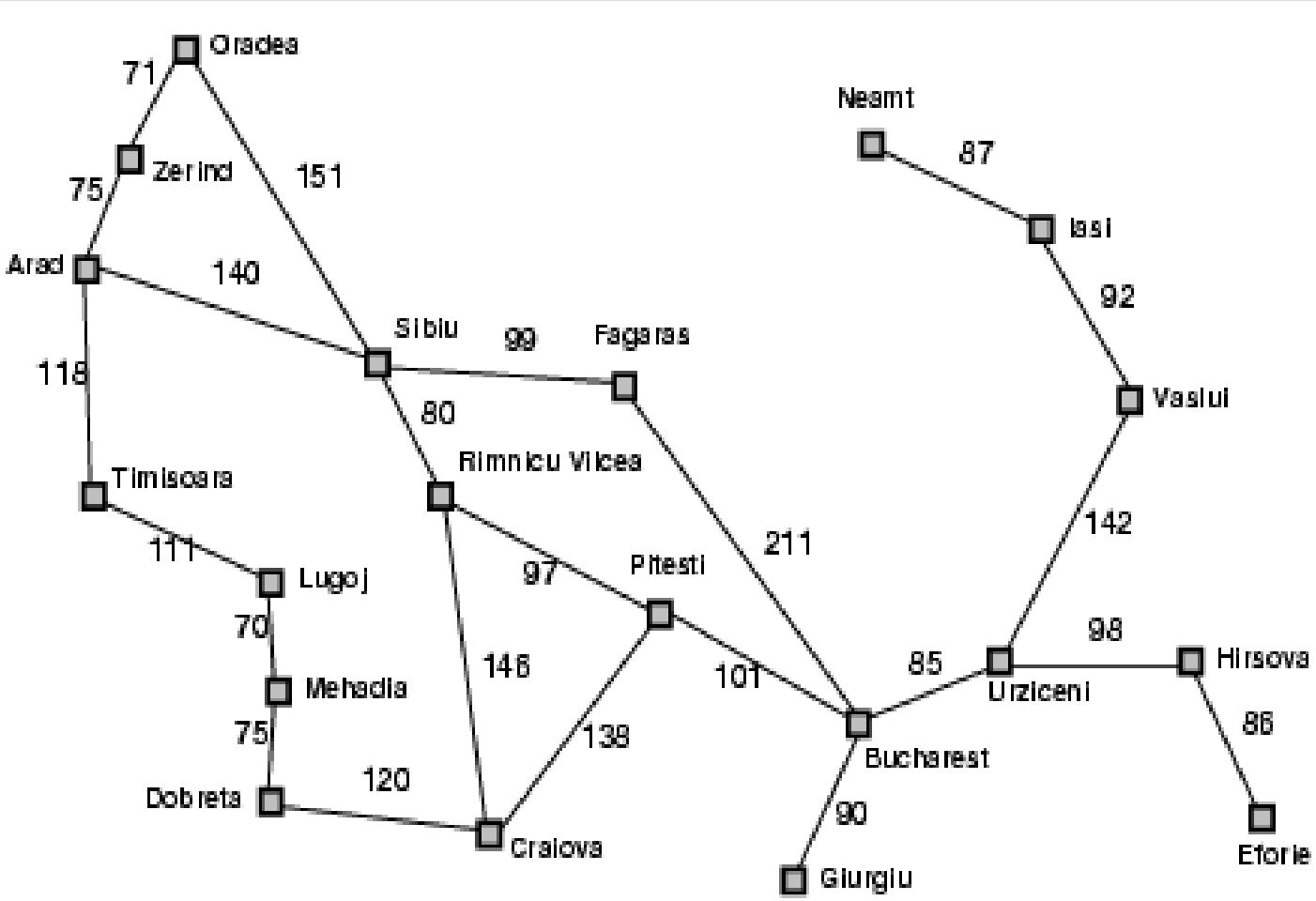



s(5)
2 8 3
1 6 4
7 5
1
A(7)
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
2
B(5)
2 8 3
1 6 4
7 5
C(7)
D(7)
2 8 3
1 4
7 6 5
2 3
1 8 4
7 6 5
3
E(5)
2 8 3
1 4
7 6 5
F(7)
2 3
1 8 4
7 6 5
4
I(5)
2 3
1 8 4
7 6 5
J(7)
1 2 3
8 4
7 6 5
5
K(5)
扩展的节点数:11
L(5)
1 2 3
8 4
7 6 5
目标
1 2 3
7 8 4
6 5
M(7)
《人工智能》陈昭炯

Greedy best-first search

- **Evaluation function *f(n)* = *h(n)* (heuristic)**
- **Greedy best-first search expands the node that appears to be closest to goal**

Romania with step costs in km

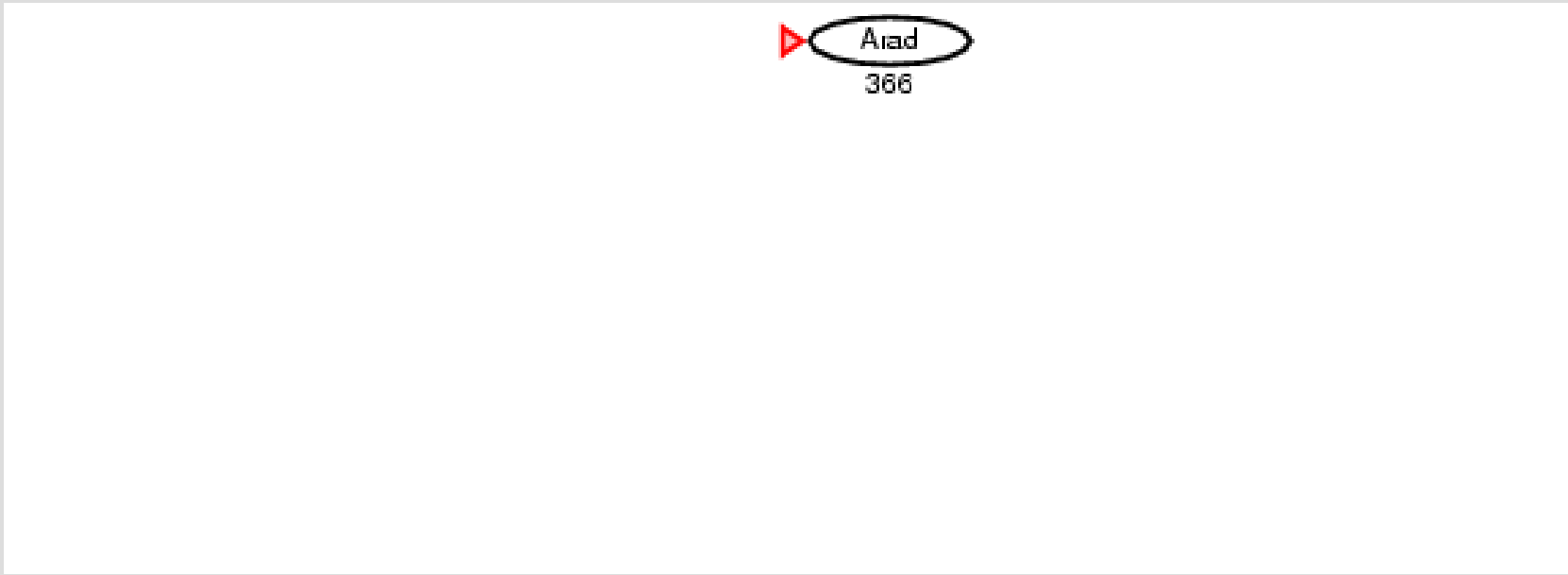


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

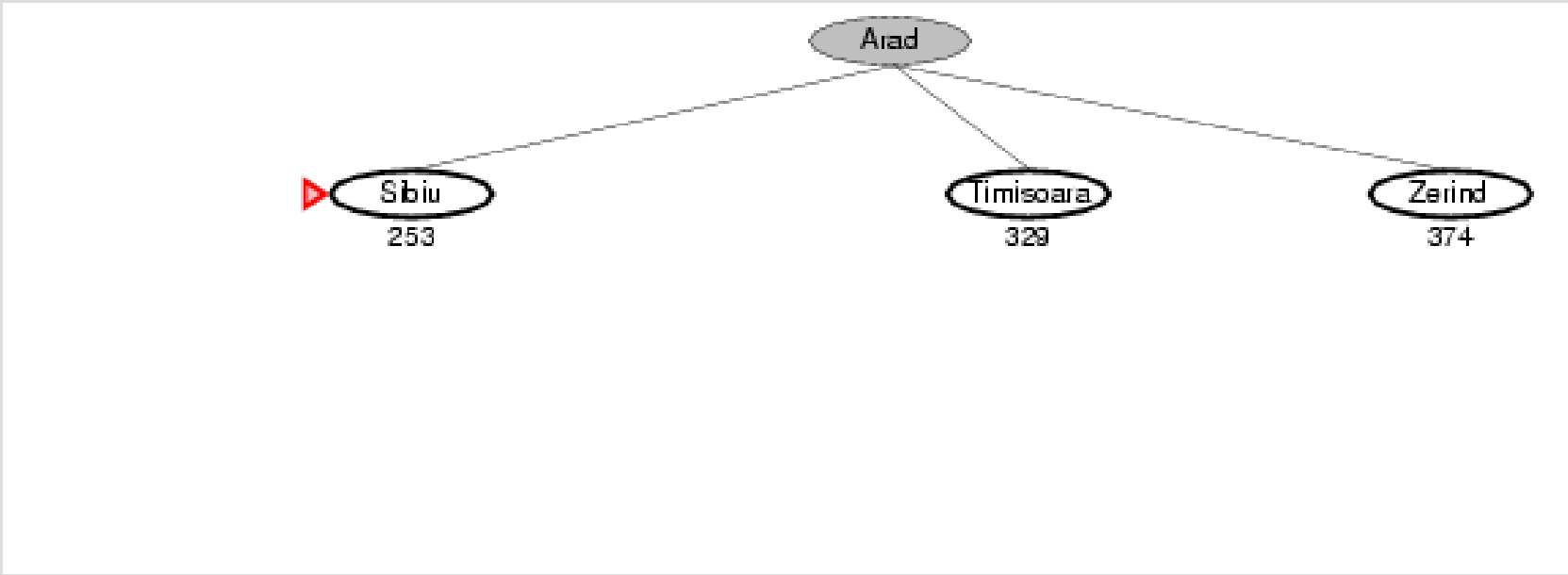
***h(n)* = straight-line distance from *n* to Bucharest**

《人工智能》陈昭炯

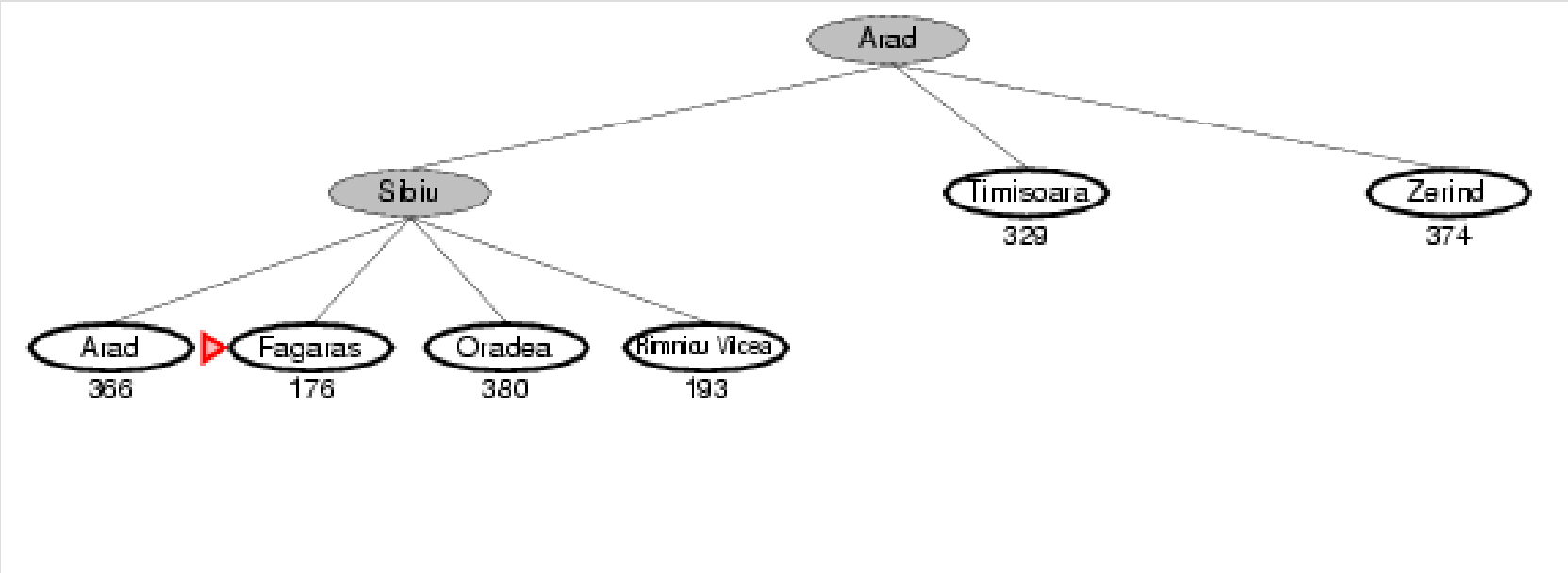
Greedy best-first search example



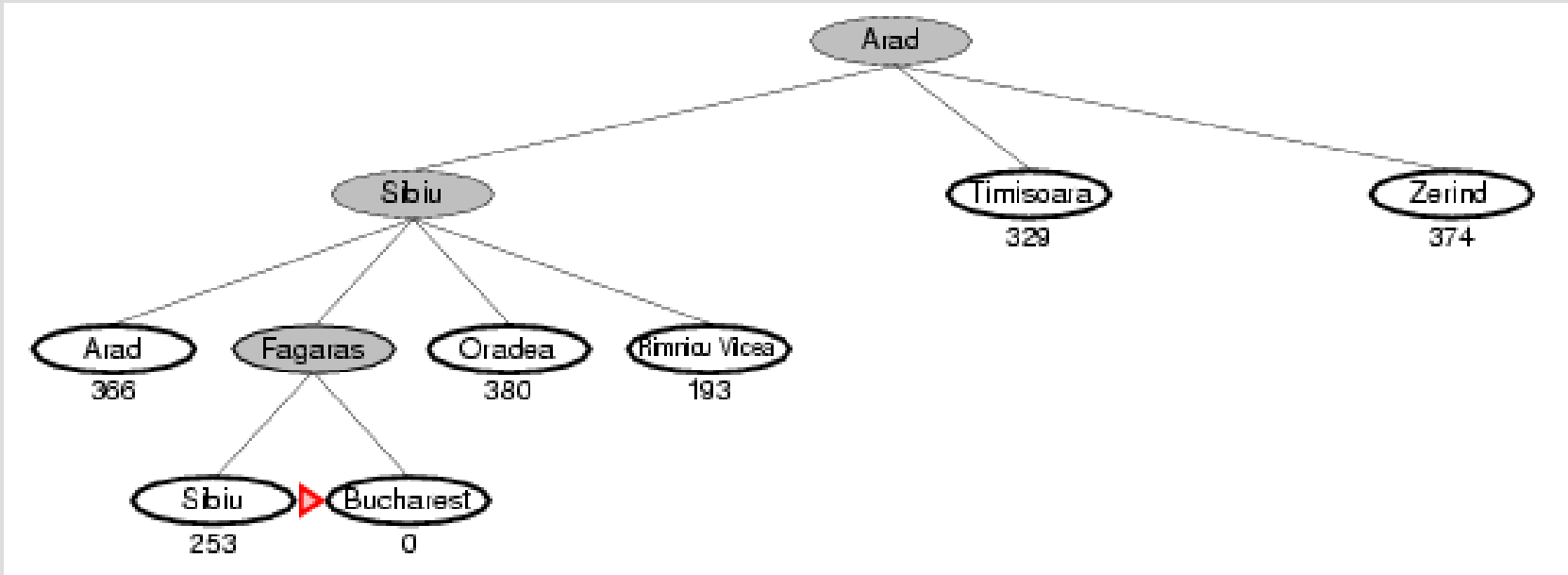
Greedy best-first search example

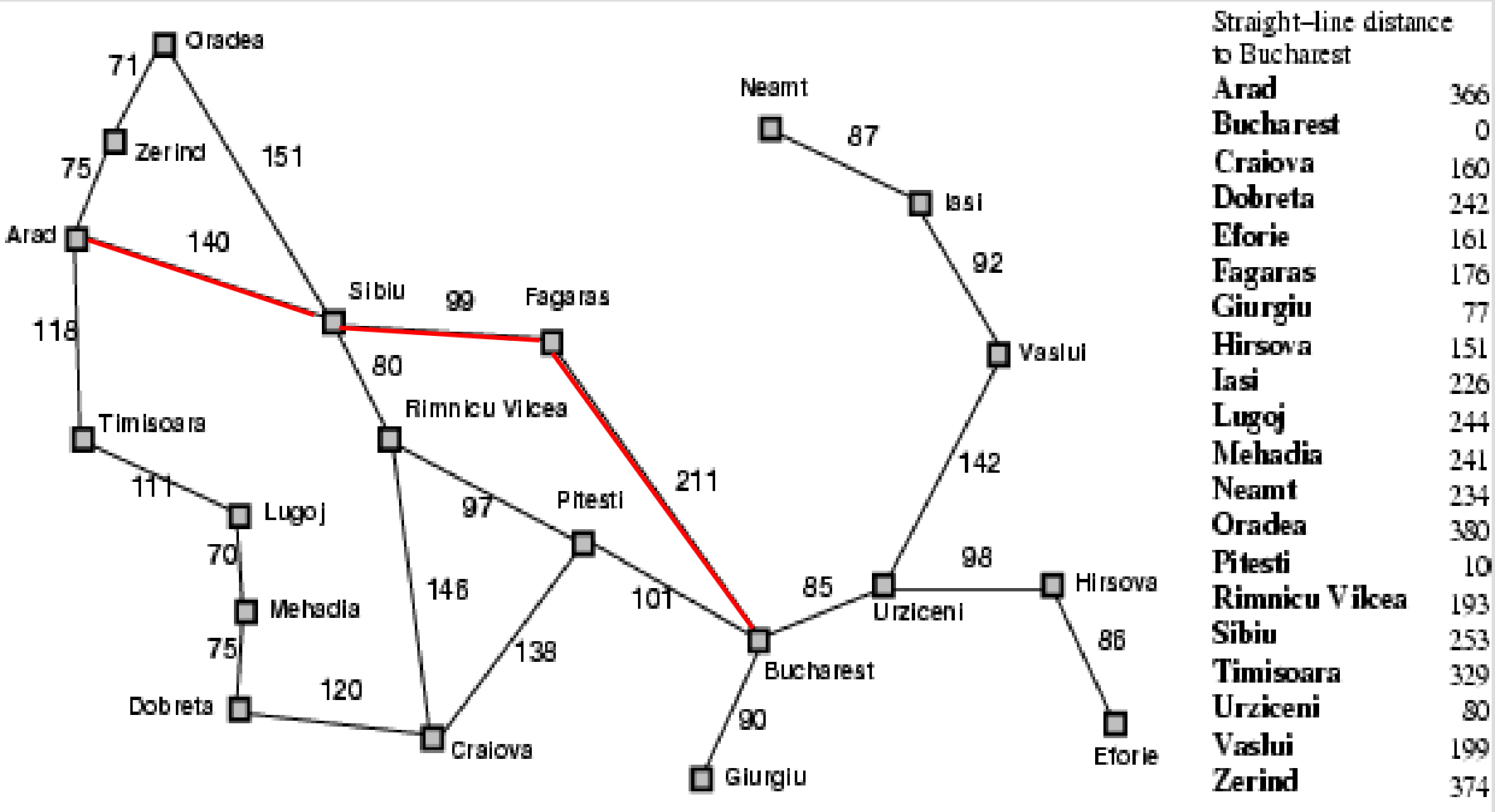


Greedy best-first search example



Greedy best-first search example





Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

《人工智能》陈昭炯

Properties of greedy best-first search

- **Complete? No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt →**
- **Time? $O(b^m)$, but a good heuristic can give dramatic improvement**
- **Space? $O(b^m)$ -- keeps all nodes in memory**
- **Optimal? No**

爬山法（Hill-climbing）

n:=s

LOOP:IF GOAL(n) THEN EXIT(SUCCESS);

EXPAND(n)→{m_i},计算h(m_i),nextn:=m(min h(m_i)的节点)；

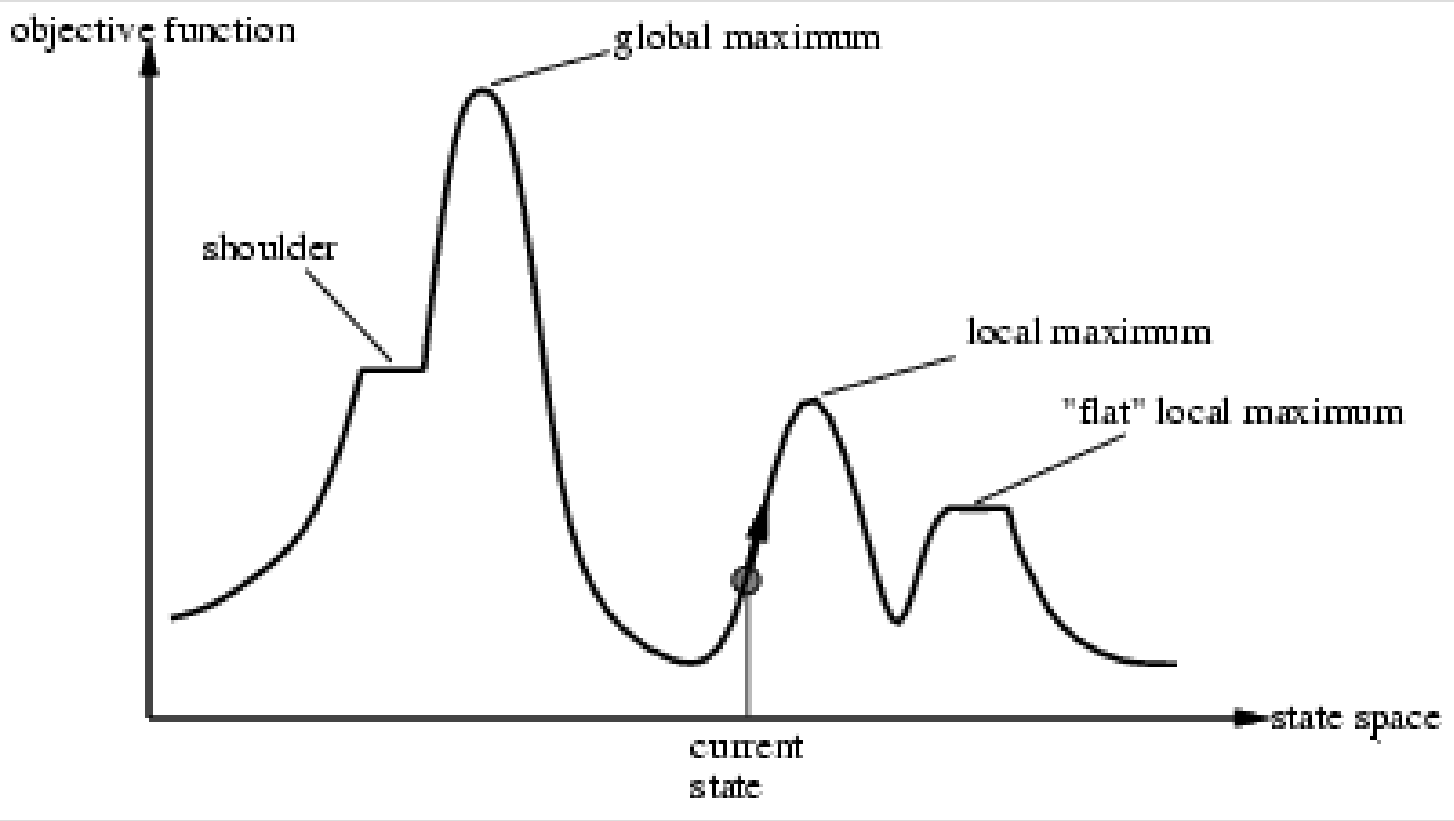
IF h(n)<h(nextn) THEN EXIT(FAIL);

N:=nextn;

GO LOOP;

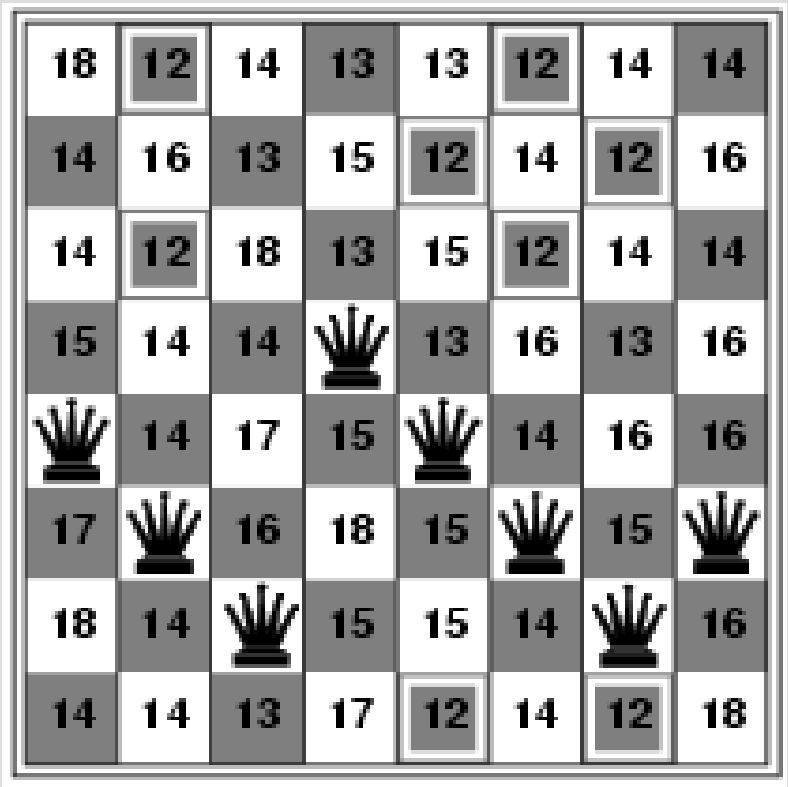
Hill-climbing search

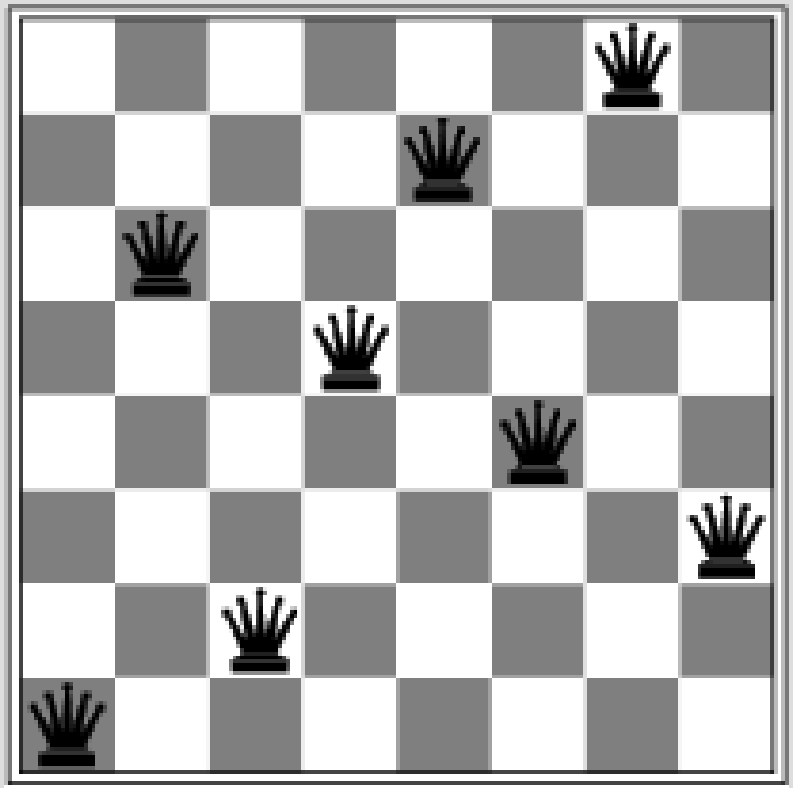
- "Like climbing Everest in thick fog with amnesia"
- Problem: depending on initial state, can get stuck in local maxima



《人工智能》陈昭炯

Hill-climbing search: 8-queens problem





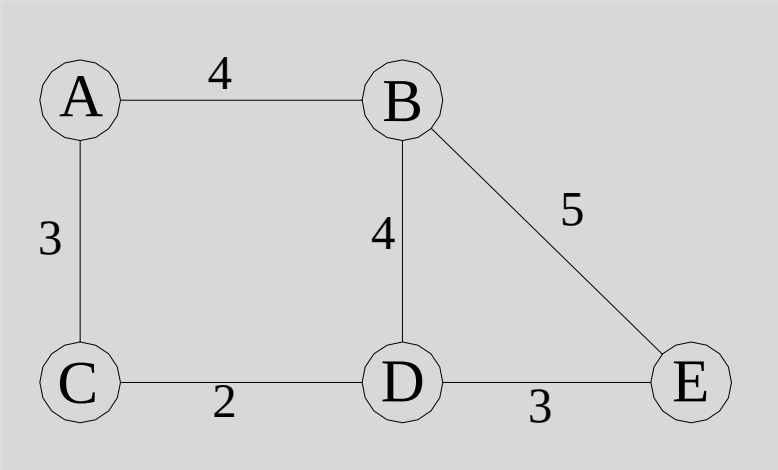
***h* = 17** —5 steps→ **A local minimum with *h* = 1**

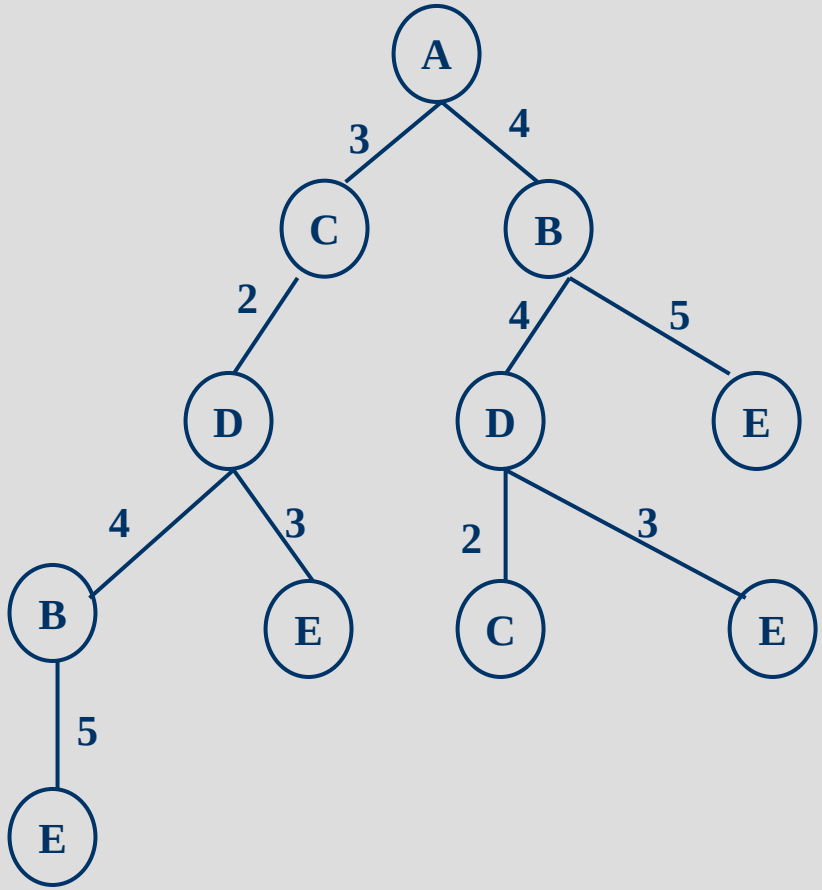
***h* = number of pairs of queens that are attacking each other, either directly or indirectly**

分支界限法(f(n)=g(n))

Branch-Bound
QUEUE:=(s),g(s)=0;
LOOP: IF QUEUE=( ) THEN EXIT(Fail);
PATH:=FIRST(QUEUE),n:=LAST(PATH);
IF GOAL(n) THEN EXIT(SUCCESS);
EXPAND(n) →{m_i},计算g(m_i)=g(n,m_i);
REMOVE(s-n,QUEUE),ADD(s-m_i,QUEUE);
QUEUE中局部（全局）路径g值最小者排在前面；
GO LOOP

例：找一条从A到E的最短路径





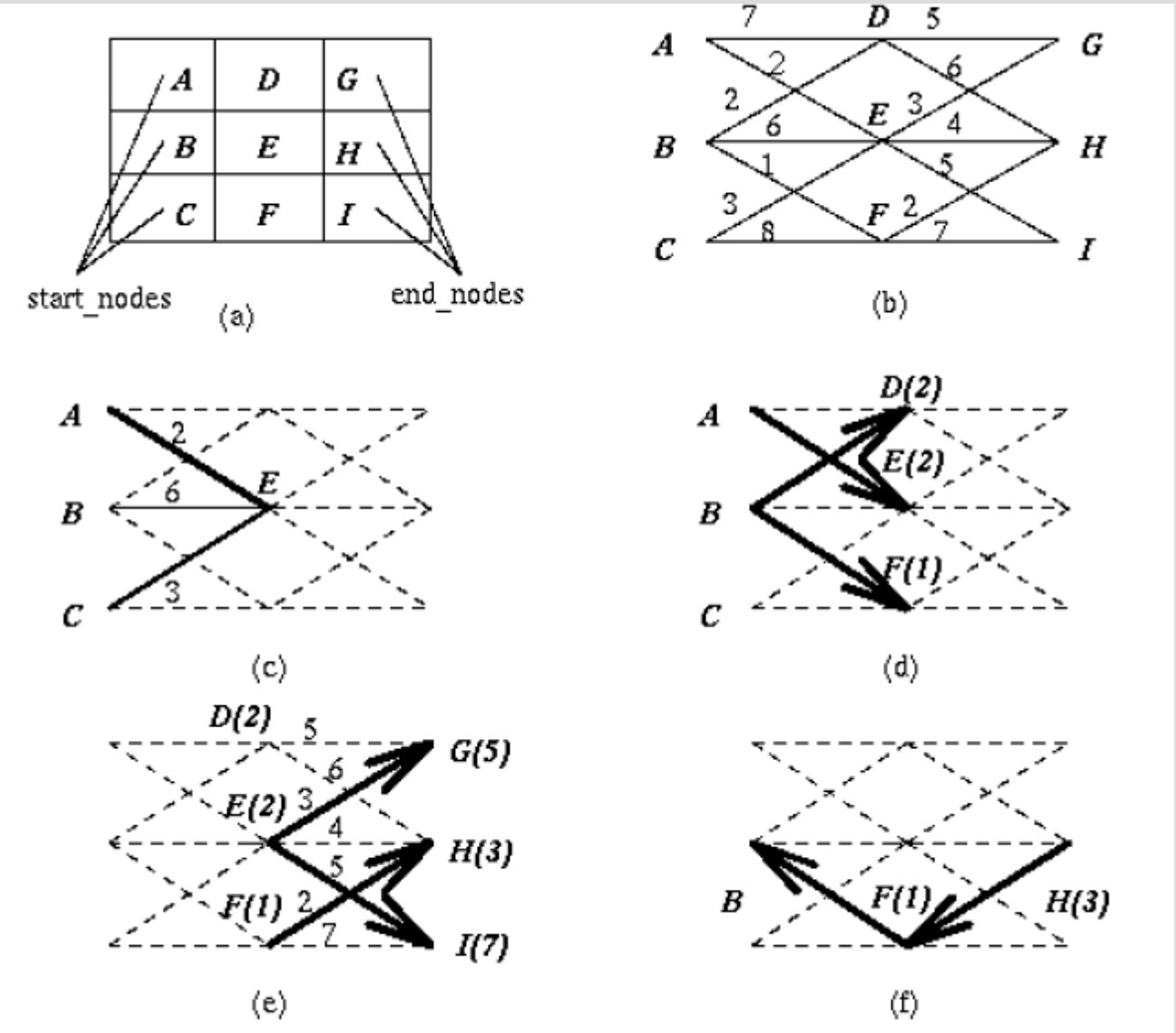
是否需遍历所有路径才可确定最短？

动态规划示图

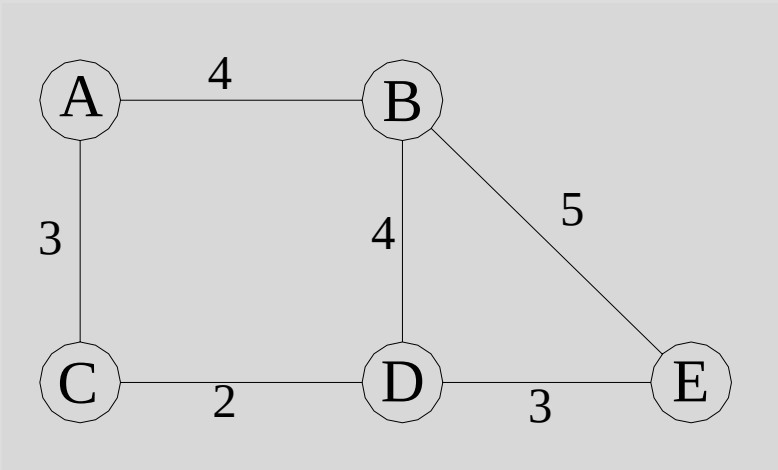
动态规划示例

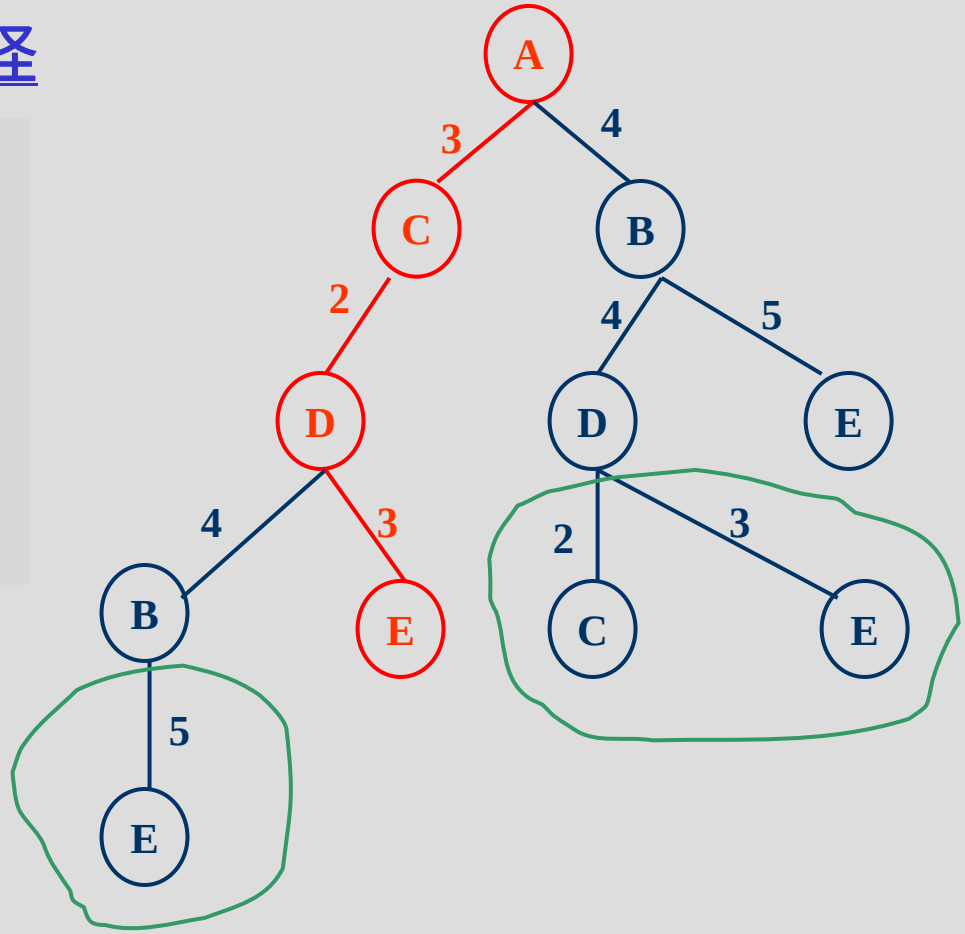
•通常从多个起点和终点中同时搜索最优路径

•若起点S0到终点Sg的最优路径经过点E，则S0到E和E到Sg的路径也是最优的。



例：找一条从A到E的最短路径





是否需遍历所有路径才可确定最短？
求S0到终点Sg的最优路径，对某一中间节点E，只需保留S0到E中耗散最小的局部路径，其余S0到E的路径可剪。

A*算法

- 如果A算法满足可纳条件： $h(n) \leq h^*(n)$　则称为A*算法。
- 分支界限与动态规划及使用下界范围的h相结合的算法

九宫问题解1　　九宫问题解2

A*算法的性质

- A*算法是可纳的：对于可解状态空间图，A*算法在有限步内终止并找到最优解。
- OPEN表上任一具有$f(n)<f^*(s_0)$的节点n，最终都将被A*选做扩展节点
- A*选做扩展的任一节点，都有$f(n) \leq f^*(s_0)$
- 在$h(n) \leq h^*(n)$的条件下， h(n)的值越大，携带的启发式信息越多，扩展的节点数越少，搜索效率越高。

（评价指标是“扩展的节点数”，即同一个节点无论被扩展多少次，都只计算一次）

- A1和A2均为A*算法，$h1(n)<h2(n)$,则在具有一条解路径的隐含图上，A2扩展的节点必定不多于被A1所扩展的节点

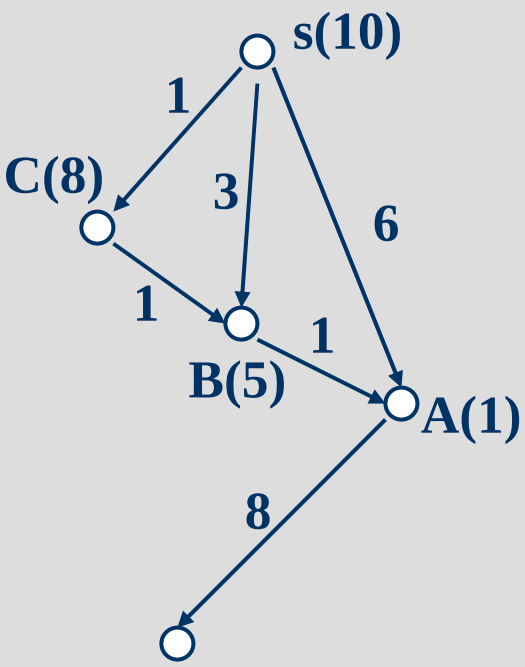
8数码的两种启发式算法？

A*算法的改进

- 问题的提出：

  因A算法第6步对m_l类节点要重新放回OPEN表中，因此可能会多次重复扩展同一个节点，导致搜索效率下降。

一个例子：

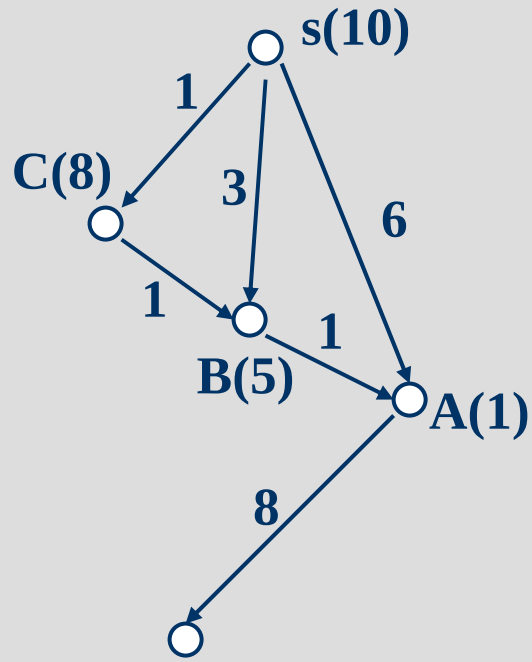


G 目标

OPEN表	CLOSED表
s(10)	s(10)
A(7) B(8) C(9)	A(7) s(10)
B(8) C(9) G(14)	B(8) ~~A(7)~~ s(10)
A(5) C(9) G(14)	A(5) B(8) s(10)
C(9) G(12)	C(9) A(5) ~~B(8)~~ s(10)
B(7) G(12)	B(7) C(9) ~~A(5)~~ s(10)
A(4) G(12)	A(4) B(7) C(9) s(10)
G(11)	

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

- 思路一：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。
- 思路二：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
- 不增加太多算法的复杂性

思路一：对h加以限制

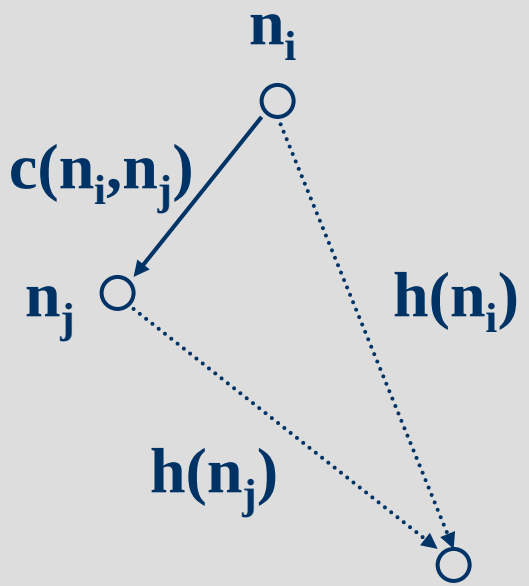
- 定义：一个启发函数h，如果对所有节点n_i和n_j，其中n_j是n_i的子节点，满足

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

或

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

则称h是单调的。



h单调的性质

- 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$g(n)=g^*(n)$。
- 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$f(n_i) \leq f(n_j)$。
- 一般图搜索算法可简化

h单调的例子

- 8数码问题：
  - h为“不在位”的将牌数

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

$h(t) = 0$

$c(n_i, n_j) = 1$

满足单调的条件。 $h(n_i)-h(n_j) \leq c(n_i,n_j)$

$h(t)=0$

h的单调化方法

- 如果令：

  f(n) = max(f(n的父节点), g(n)+h(n))

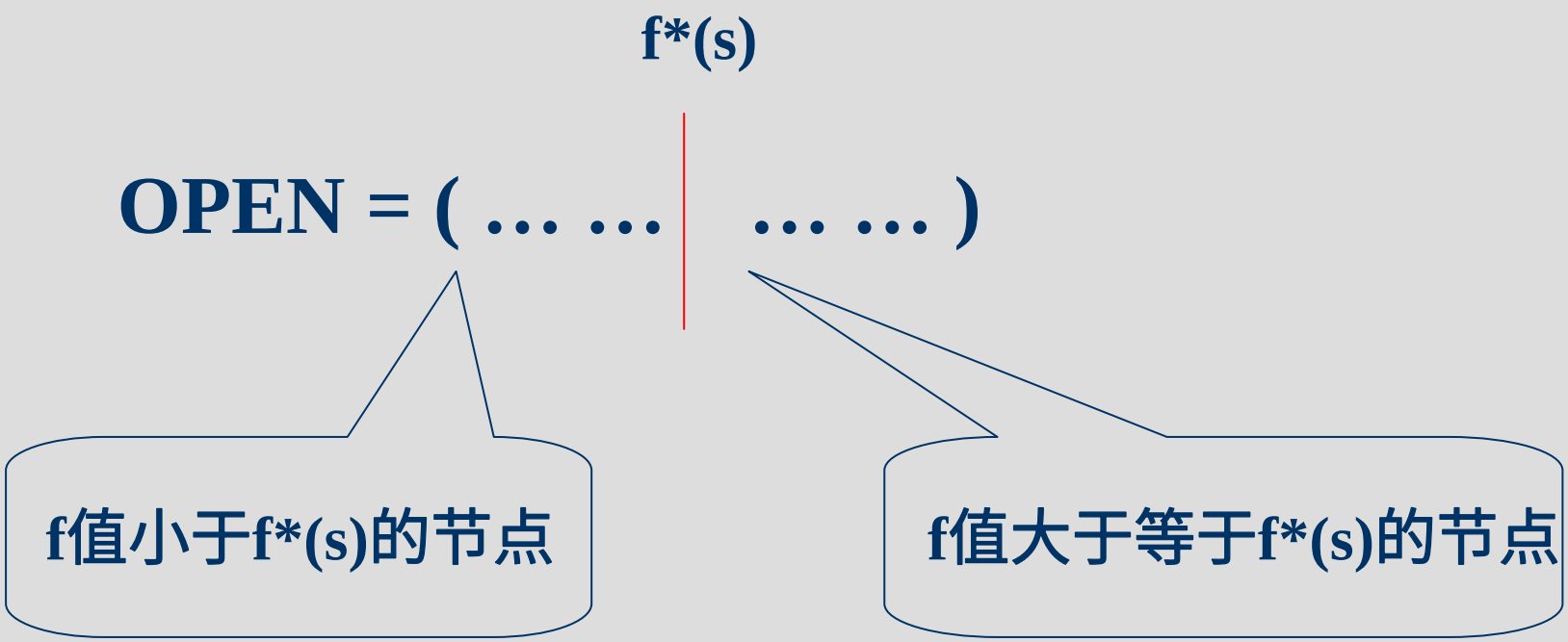
  则容易证明，这样处理后的h是单调的。

但可能影响h的有效性

思路二：对算法加以改进

- 两个结论：
  - OPEN表上任一具有$f(n) < f^*(s)$的节点定会被扩展。
  - A*选作扩展的任一节点，定有$f(n) \leq f^*(s)$。

改进的出发点



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:=f(s);

2, LOOP: IF OPEN=( ) THEN EXIT(FAIL);

3, NEST:={n_i|f(n_i)<f_m}

IF NEST ≠( ) THEN n:=NEST中g最小的节点

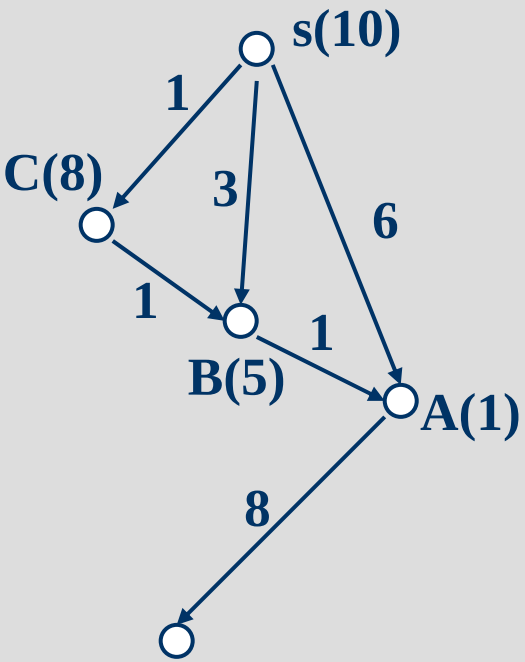
ELSE n:=FIRST(OPEN),

f_m:=f(n);

4, …, 8: 同过程A。

在NEST集合中，令h=0，此时h满足单调性条件

前面的例子：



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

Relaxed problems

- A problem with fewer restrictions on the actions is called a **relaxed problem**
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution

八数码问题，松弛条件3:

一个棋子可以从方格A移动到方格B,如果B是空的

启发函数?

例：修道士与野人问题(1968)

河左岸有N 个Missionaries和k 个Cannibals,1条boat

条件：1）M和C都会划船，船一次只能载2人

2）在任一岸上，M人数不得少于C的人数，否则被吃

目标：安全抵达对岸的最佳方案

- (M,C,b):S_0(5,5,1)
- 松弛条件：无限制时，
- （M,C,1），最少次数：

$$\left[\frac{M+C-3}{2}\right]\times 2+1\geq \frac{M+C-3}{2}\times 2+1=M+C-2$$

- h1(n)=0(A*，35步)
- h2(n)=M+C（非A*，20步）
- h3(n)=M+C-2b（A*，21步）

（M,C,0）最少次数=（M+1，C,1）最少次数+1=M+1+C-2+1=M+C

A*算法的其它不足：

- 大多数情况节点数为解长度的指数级，$|h(n)-h^*(n)| \le O(\log h^*(n))$
- 存储量过大，保留了所有生成的节点，不适用于大规模问题
- 存储限制的A*算法
  - **Solution: *Iterative-deepening A* (IDA*)***
    - **Just like Iterative Deepening, but...**
    - **...instead of using g(N) (the actual depth so far) to cut off searching...**
    - **...use f(N) (the estimated *total* depth)**
  - **IDA* gives the same results as A***
  - **Since IDA* is basically a depth-first search, storage requirements are linear in length of path**

启发式函数的精确度问题实验(8数码问题扩展的节点数)

解的深度	迭代有界深度	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
14		539	113
16		1301	211
18		3056	363
20		7276	676
22		18094	1219
24		39135	1641

搜索的完备性与效率

- 完备性：若问题可解则搜索过程一定能找解，称这样的搜索过程为完备的，完备的搜索过程也称为搜索算法
- 广度类，改进的有界深度，A *——完备
- 搜索效率之外显率(渗透率)：$P=L/T \leq 1$

  L:$S_0 \rightarrow S_g$的最优路径长度； T:搜索生成的节点总数

  P越小效率越低。P=1 ……

搜索效率之有效分枝因数

定义：满足式：$B+B^2+\cdots\cdots+B^L=T$ 的数B

意义：每一有效节点平均生成的节点数目，越小越好

关联：

$$P=\frac{L\times(B-1)}{B\times(B^L-1)},\qquad T=\frac{B\times(B^L-1)}{B-1}$$

解释：（可绘制如下关系图表）

①固定B，则L越大P越小，即有效分枝因数固定的情况下，最优解越远，搜索效率越低

②固定L，则B越大P越小T越大，即最优解一定的情况下，有效分枝因数越大，搜索生成的节点总数越多，搜索效率越低

8数码问题不同搜索算法的有效分支因子比较

解的深度	迭代有界深度	$A^*(h_1)$	$A^*(h_2)$
2	2.45	1.79	1.79
4	2.87	1.48	1.45
6	2.73	1.34	1.30
8	2.80	1.33	1.24
10	2.79	1.38	1.22
12	2.78	1.42	1.24
14	-	1.44	1.23
16	-	1.45	1.25
18	-	1.46	1.26
20	-	1.47	1.27
22	-	1.48	1.28
24	-	1.48	1.26

问题：

•如何在盲目搜索中避免重复状态？对广度和深度优先算法这种策略有何不同？

•树与图的区别是什么？其搜索方法有何不同？如何选择？

•了解若干经典搜索问题的最新进展，如：皇后问题，滑块问题，TSP问题，规划问题，巡游问题等。

•是否还有比较通用的启发函数的构造方法？

•8数码问题的所有状态可以划分为两个互不相交的集合，如何说明？

•能否找到8数码问题更有效的启发式函数？

•A*算法除了具有完备性、最优性之外，是否在效率上优于一般的启发式算法？