1. 什么是图搜索过程？其中，重排 OPEN 表意味着什么，重排的原则是什么？

答：图搜索的一般过程如下：

(1) 建立一个搜索图 G（初始只含有起始节点 S），把 S 放到未扩展节点表中（OPEN 表）中。

(2) 建立一个已扩展节点表（CLOSED 表），其初始为空表。

(3) LOOP: 若 OPEN 表是空表，则失败退出。

(4) 选择 OPEN 表上的第一个节点，把它从 OPEN 表移出并放进 CLOSED 表中。称此节点为节点 n。

(5) 若 n 为一目标节点，则有解并成功退出。此解是追踪图 G 中沿着指针从 n 到 S 这条路径而得到的(指针将在第 7 步中设置)。

(6) 扩展节点 n,生成后继节点的集合 M。

(7) 对那些未曾在 G 中出现过的(既未曾在 OPEN 表上，也未在 CLOSED 表上出现过的)M 成员设置其父节点指针指向 n 并加入 OPEN 表。对已经在 OPEN 或 CLOSED 表中出现过的每个 M 成员，确定是否需要将其原来的父节点更改为 n。对已在 CLOSED 表上的每个 M 成员，若修改了其父节点，则将该节点从 CLOSED 表中移出，重新加入 OPEN 表中。

(8) 按某一任意方式或按某个探试值，重排 OPEN 表。

(9) GO LOOP。

重排 OPEN 表意味着，在第（6）步中，将优先扩展哪个节点，不同的排序标准对应着不同的搜索策略。

重排的原则当视具体需求而定，不同的原则对应着不同的搜索策略，如果想尽快地找到一个解，则应当将最有可能到达目标节点地那些节点排在 OPEN 表的前面部分，如果想找到代价最小的解，则应当按代价从小到大的顺序重排 OPEN 表。

2. 化为子句形有哪些步骤？请结合例子说明。

答：任一谓词演算公式可以化为一个子句集，其变换过程由下列九个步骤组成：

(1) 消去蕴含符号

将蕴含符号化为析取和否定符号。

(2) 减少否定符号的辖域

每个否定符号最多只用到一个谓词符号上，并反复应用狄摩·根定律。

(3) 对变量标准化

对哑元改名以保证每个量词有其唯一的哑元。

(4) 消去存在量词

引入 Skolem 函数，消去存在量词。如果要消去的存在量词不在任何一个全称量词的辖域内，那么就用不含变量的 Skolem 函数即常量。

(5) 化为前束形

把所有全称量词移到公式的左边，并使每个量词的辖域包括这个量词后面公式的整个部分。前束形公式由前缀和母式组成，前缀由全称量词串组成，母式由没有量词的公式组成。

(6) 把母式化为合取范式

反复应用分配律，将母式写成许多合取项的合取的形式，而每一个合取项是一些谓词公式和（或）谓词公式的否定的析取。

(7) 消去全称量词

消去前缀，即消去明显出现的全称量词。

(8) 消去连词符号∧

用{合取项 1，合取项 2}替换明显出现的合取符号。

(9) 更换变量名称

更换变量符号的名称，使一个变量符号不出现在一个以上的子句中。

下面举个例子来说明上述过程，如下：

$$(\forall x)(P(x) \rightarrow ((\forall y)(P(y) \rightarrow P(f(x,y))) \wedge \sim (\forall y)(Q(x,y) \rightarrow P(y))))$$

(1) $(\forall x)(\sim P(x) \vee ((\forall y)(\sim P(y) \vee P(f(x,y)))$
$\wedge \sim (\forall \mathrm{y})(\sim Q(x,y) \vee P(y))))$

(2) $(\forall x)(\sim P(x) \vee ((\forall y)(\sim P(y) \vee P(f(x,y)))$
$\wedge (\exists \mathrm{y})(\sim (\sim Q(x,y) \vee P(y)))))$

$(\forall x)(\sim P(x) \vee ((\forall y)(\sim P(y) \vee P(f(x,y)))$
$\wedge (\exists \mathrm{y})(Q(x,y) \wedge \sim P(y))))$

(3) $(\forall x)(\sim P(x) \vee ((\forall y)(\sim P(y) \vee P(f(x,y)))$
$\wedge (\exists w)(Q(x,w) \wedge \sim P(w))))$

(4) $(\forall x)(\sim P(x) \vee ((\forall y)(\sim P(y) \vee P(f(x,y)))$
$\wedge (Q(x,g(x)) \wedge \sim P(g(x)))))$

式中，$w = g(x)$为一个 Skolem 函数。

(5) $(\forall x)(\forall \mathrm{y})(\sim P(x) \vee ((\sim P(y) \vee P(f(x,y)))$
$\wedge (Q(x,g(x)) \wedge \sim P(g(x)))))$

(6) $(\forall x)(\forall \mathrm{y})((\sim P(x) \vee \sim P(y) \vee P(f(x,y)))$
$\wedge (\sim P(x) \vee Q(x,g(x))) \wedge (\sim P(x) \vee \sim P(g(x))))$

(7) $(\sim P(x) \vee \sim P(y) \vee P(f(x,y)))$
$\wedge (\sim P(x) \vee Q(x,g(x))) \wedge (\sim P(x) \vee \sim P(g(x)))$

(8) $\sim P(x) \vee \sim P(y) \vee P(f(x,y))$

$\sim P(x) \vee Q(x,g(x))$

$\sim P(x) \vee \sim P(g(x))$

(9) 更改变量名称，在上述第（8）步的 3 个子句中，分别以$x1$，$x2$，$x3$代替变量x，得到下列子句集：

$$\sim P(x1) \vee \sim P(y) \vee P(f(x1,y))$$

$$\sim P(x2) \vee Q(x2,g(x2))$$

$$\sim P(x3) \vee \sim P(g(x3))$$

3. 如何通过消解反演求解问题的答案？

答：给出一个公式集 S 和目标公式 L，通过反证或者反演来求证目标公式 L，其证明步骤如下：

(1) 否定 L，得～L；

(2) 把～L 添加到 S 中去；

(3) 把新产生的集合{～L，S}化为子句集；

(4) 应用消解原理，力图推导出一个表示矛盾的空子句。

4. 用有界深度优先搜索方法求解图 3.30 所示八数码难题。

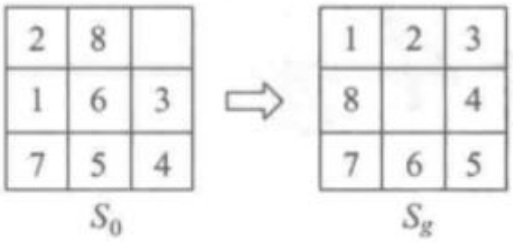
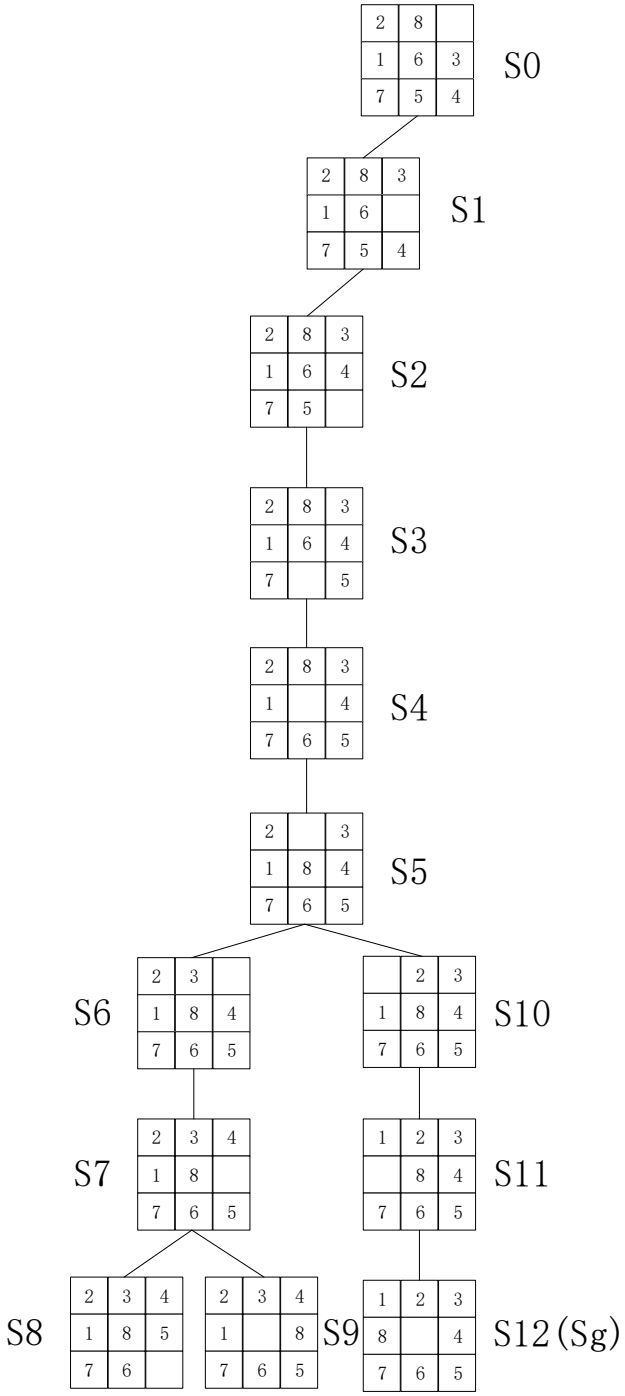


图 3.30　八数码难题

答：按顺时针方向移动空格，将最大深度定为 9，其有界深度优先搜索树如下所示：



5. 规则演绎系统和产生式系统有哪几种推理方式？各自的特点是什么？

答：规则演绎系统得推理方式有正向推理、逆向推理和双向推理。

项目	正向推理	逆向推理
推理方向	从 if 部分向 then 部分推理的过程，它是从事实或状况向目标或动作进行操作的	从 then 部分向 if 部分推理的过程，它是从目标或动作向事实或状况进行操作的
目标表达式	文字的析取	任意形式

事实表达式	任意形式	文字的合取

双向推理组合了正向推理和逆向推理的优点，克服了各自的缺点，具有更高的搜索求解效率。

产生式系统的推理方式有正向推理、逆向推理和双向推理。

项目	正向推理	逆向推理
驱动方式	数据驱动	目标驱动
推理方法	从一组数据出发向前推导结论	从可能的解答出发，向后推理验证解答
启动方法	从一个事件启动	由询问关于目标状态的一个问题而启动
透明程序	不能解释其推理过程	可解释其推理过程
推理方向	由底向上推理	由顶向下推理
优点	算法简单，容易实现	搜索目的性强，推理效率高
缺点	盲目搜索，可能会求解许多与总目标无关的子目标，每当总数据库内容更新后都要遍历整个规则库，推理效率低	目标的选择具有盲目性，可能会求解许多假的目标；当可能的结论数目很多时，推理效率不高；当规则的右部是执行某种动作而不是结论时，逆向推理不便使用
适用场合	已知初始数据库，而无法提供推理目标，或解空间很大的一类问题，如监控、预测、规划、设计等问题	结论单一或者已知目标结论，而要求验证的系统，如选择、分类、故障诊断等问题
典型系统	CLIPS、OPS	PROLOG

双向推理结合了正向推理和逆向推理的长处，克服了两者的短处，其控制策略比两者都要复杂。

6. 下列语句是一些几何定理，把这些语句表示为基于规则的几何证明系统的产生式规则：

(1) 两个全等三角形的各对应角相等。

(2) 两个全等三角形的各对应边相等。

(3) 各对应边相等的三角形是全等三角形。

(4) 等腰三角形的两底角相等。

答：规则如下：

规则（1）：IF　两个三角形全等

　　THEN 各对应角相等

规则（2）：IF　两个三角形全等

　　THEN 各对应边相等

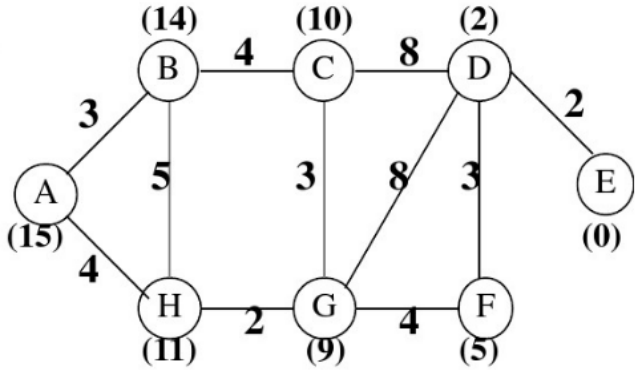
规则（3）：IF　两个三角形各对应边相等

　　THEN 两个三角形全等

规则（4）：IF　它是等腰三角形

　　THEN 它的两底角相等

7. **从用 5 种搜索方法（广度、深度、等代价、有序、A*）分别求解其搜索路径，并给出对应的 OPEN 表和 CLOSED 表，A 为起点，E 为终点。**



答：

(1) 广度

第一次：

OPEN 表

节点	父节点
A	NULL

CLOSED 表

节点	父节点

第二次：

OPEN 表

节点	父节点
B	A
H	A

CLOSED 表

节点	父节点
A	NULL

第三次：

OPEN 表

节点	父节点
H	A
C	B

CLOSED 表

节点	父节点
A	NULL
B	A

第四次：

OPEN 表

节点	父节点
C	B
G	H

CLOSED 表

节点	父节点
A	NULL
B	A
H	A

第五次：

OPEN 表

节点	父节点
G	H
D	C

CLOSED 表

节点	父节点
A	NULL
B	A
H	A
C	B

第六次：

OPEN 表

节点	父节点
D	C
F	G

CLOSED 表

节点	父节点
A	NULL
B	A
H	A
C	B
G	H

第七次：

OPEN 表

节点	父节点
F	G
E	D

CLOSED 表

节点	父节点
A	NULL
B	A
H	A
C	B
G	H
D	C

此时，有一后继节点为目标节点 E，算法结束，搜索得到的最优路径为 ABCDE。

(2) 深度

第一次：

OPEN 表

节点	父节点
A	NULL

CLOSED 表

节点	父节点

第二次：

OPEN 表

节点	父节点
B	A
H	A

CLOSED 表

节点	父节点
A	NULL

第三次：

OPEN 表

节点	父节点
C	B
H	A

CLOSED 表

节点	父节点
A	NULL
B	A

第四次：

OPEN 表

节点	父节点
D	C

H	A

CLOSED 表

节点	父节点
A	NULL
B	A
C	B

第五次：

OPEN 表

节点	父节点
E	D
H	A

CLOSED 表

节点	父节点
A	NULL
B	A
C	B
D	C

此时，有一后继节点为目标节点 E，算法结束，搜索得到的最优路径为 ABCDE。

(3) 等代价

第一次：

OPEN 表

节点	父节点	代价
A	NULL	0

CLOSED 表

节点	父节点	代价

第二次：

OPEN 表

节点	父节点	代价
B	A	3
H	A	4

CLOSED 表

节点	父节点	代价
A	NULL	0

第三次：

OPEN 表

节点	父节点	代价
H	A	4
C	B	7

CLOSED 表

节点	父节点	代价
A	NULL	0

B	A	3

第四次：

OPEN 表

节点	父节点	代价
C	B	7
G	H	6

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3
H	A	4

第五次：

OPEN 表

节点	父节点	代价
C	B	7
D	G	14
F	G	10

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3
H	A	4
G	H	6

第六次：

OPEN 表

节点	父节点	代价
D	G	14
F	G	10

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3
H	A	4
G	H	6
C	B	7

第七次：

OPEN 表

节点	父节点	代价
D	F	13

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3

H	A	4
G	H	6
C	B	7
F	G	10

第八次：

OPEN 表

节点	父节点	代价
E	D	15

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3
H	A	4
G	H	6
C	B	7
F	G	10
D	F	13

第九次：

OPEN 表

节点	父节点	代价

CLOSED 表

节点	父节点	代价
A	NULL	0
B	A	3
H	A	4
G	H	6
C	B	7
F	G	10
D	F	13
E	D	15

此时，OPEN 表为空，算法结束，搜索得到的最优路径为 AHGFDE，最短距离为 15。

(4) 有序

令$f(n)=d(n)+W(n)$，其中，$d(n)$是节点 n 的深度，$W(n)$是起始节点到节点 n 的距离。

第一次：

OPEN 表

节点	父节点	$f(n)$
A	NULL	0

CLOSED 表

节点	父节点	$f(n)$

第二次：

OPEN 表

节点	父节点	$f(n)$
B	A	4
H	A	5

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0

第三次：

OPEN 表

节点	父节点	$f(n)$
H	A	5
C	B	9

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4

第四次：

OPEN 表

节点	父节点	$f(n)$
C	B	9
G	H	8

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5

第五次：

OPEN 表

节点	父节点	$f(n)$
C	B	9
F	G	13

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5
G	H	8

第六次：

OPEN 表

节点	父节点	$f(n)$
F	G	13
D	C	18

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5
G	H	8
C	B	9

第七次：

OPEN 表

节点	父节点	$f(n)$
D	F	17

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5
G	H	8
C	B	9
F	G	13

第八次：

OPEN 表

节点	父节点	$f(n)$
E	D	19

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5
G	H	8
C	B	9
F	G	13
D	F	17

第九次：

OPEN 表

节点	父节点	$f(n)$

CLOSED 表

节点	父节点	$f(n)$
A	NULL	0
B	A	4
H	A	5
G	H	8
C	B	9

F	G	13
D	F	17
E	D	19

此时,CLOSED 表中的 E 节点为目标节点,算法结束,最优路径为 AHGFDE,最短距离为 15。

(5) A*

令$f(n)=g(n)+h(n)$，其中，$g(n)$是从起始节点到节点 n 的最佳路径的距离，$h(n)$是从节点 n 到目标节点的最少操作次数。

第一次：

OPEN 表

节点	父节点	$f(n)$
A	NULL	4

CLOSED 表

节点	父节点	$f(n)$

第二次：

OPEN 表

节点	父节点	$f(n)$
B	A	6
H	A	7

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4

第三次：

OPEN 表

节点	父节点	$f(n)$
H	A	7
C	B	9

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6

第四次：

OPEN 表

节点	父节点	$f(n)$
C	B	9
G	H	8

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7

第五次：

OPEN 表

节点	父节点	$f(n)$
C	B	9
G	H	8

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7

第六次：

OPEN 表

节点	父节点	$f(n)$
C	B	9
F	G	12
D	G	15

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7
G	H	8

第七次：

OPEN 表

节点	父节点	$f(n)$
F	G	12
D	G	15

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7
G	H	8
C	B	9

第八次：

OPEN 表

节点	父节点	$f(n)$
D	F	14

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7
G	H	8

C	B	9
F	G	12

第九次：

OPEN 表

节点	父节点	$f(n)$
E	D	15

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7
G	H	8
C	B	9
F	G	12
D	F	14

第十次：

OPEN 表

节点	父节点	$f(n)$

CLOSED 表

节点	父节点	$f(n)$
A	NULL	4
B	A	6
H	A	7
G	H	8
C	B	9
F	G	12
D	F	14
E	D	15

此时，CLOSED 表中 E 节点为目标节点，算法结束，最优路径为 AHGFDE，最短距离为 15。