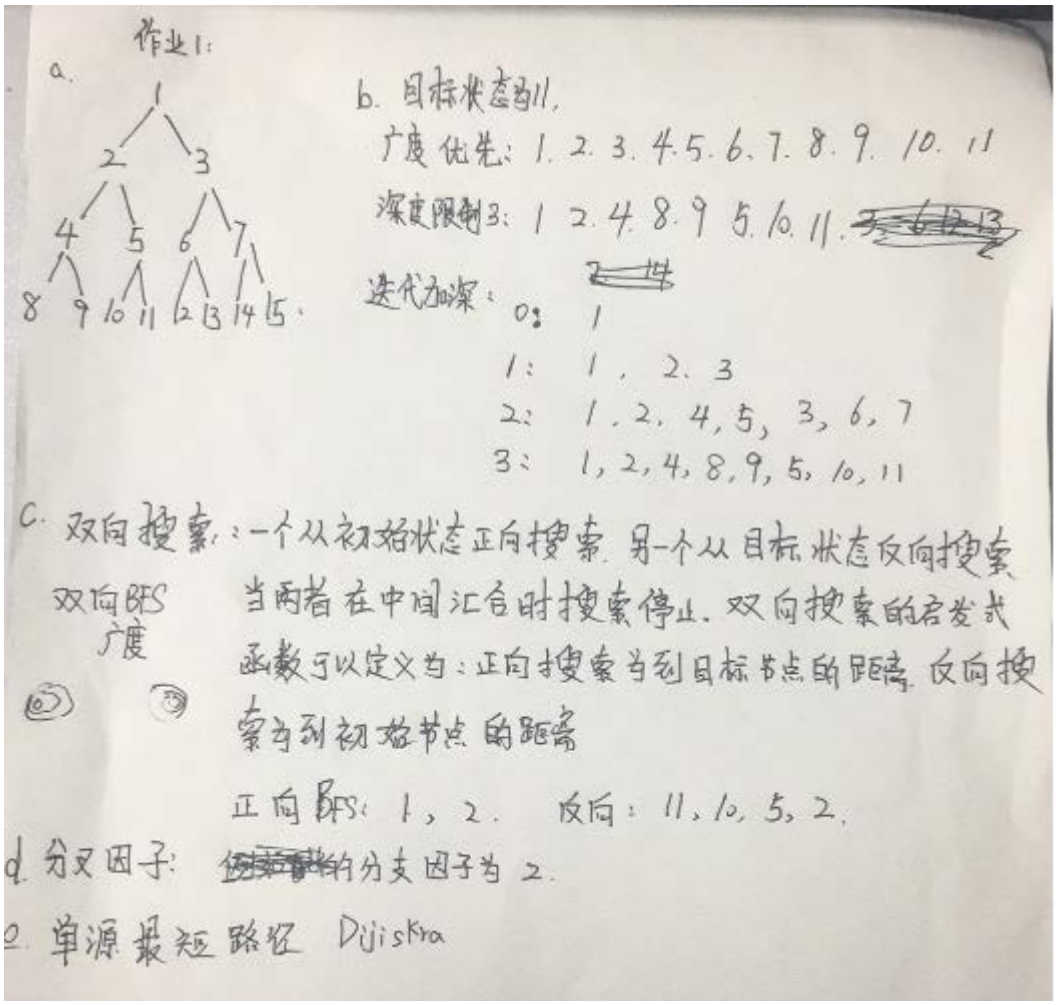
- 考虑一个状态空间，其初始编号为 1，状态 n 的后继函数返回编号为 2n 与 2n+1 的两个状态。
    a. 画出状态 1 到 15 的部分状态空间图。
    b. 假设目标状态是 11，列出用以下算法访问节点的顺序：广度优先搜索、深度限制为 3 的有限深度搜索和迭代加深搜索
    c. 双向搜索是否适合该问题？如适合，详述其工作原理
    d. 在双向搜索中每个方向上的分叉因子是什么？
    e. 对(c)的回答是否能提出该问题的另一种形式化，使得你可以几乎不用搜索来求解从状态 1 到达目标状态的问题？



- 有名的传教士和野人问题（Amarel，1968）：三个传教士和三个野人需从河的一边用一条只能载最多 2 个人的船渡到对岸。但是需保证任意时刻野人的个数不大于传教士个数（否则传教士会被野人吃掉）。
    a. 精确地形式化该问题，画出完全的状态空间。
    b. 用一个合适的搜索算法求该问题的最优解。检查重复状态是个好主意吗？
    c. 既然该问题的状态空间如此简单，为何人们求解她却显得困难？

a．

（1）设定状态变量及确定值域。

为了建立这个问题的状态空间，设**左岸传教士数**为 **m**，则

m ={0,1,2,3}；

对应右岸的传教士数为 3 - m； **左岸的野人数**为 **c**，则有

c ={0,1,2,3}；

对应右岸野人数为 3 - c；**左岸船数**为 **b**，故又有 **b={0,1}**，右岸的船数为 1 - b.

（2）确定状态组，分别列出初始状态集和目标状态集。

问题的状态可以用一个三元数组来描述，以左岸的状态来标记，即

Sk =（m,c,b），

右岸的状态可以不必标出。

初始状态一个： **S0 =（3,3,1）**，初始状态表示全部成员在河的左岸；

目标状态也只**一个**： **Sg =（0,0,0）**，表示全部成员从河左岸渡河完毕。

（3）定义并确定操作集。

仍然以河的左岸为基点来考虑，把船**从左岸划向右岸**定义为 **Pij** 操作。其中,第一下标 i 表示船载的传教士数，第二下标 j 表示船载的野人数；同理，**从右岸将船划回左岸**称之为 **Qij** 操作，下标的定义同前。则共有 10 种操作，操作集为

F={P01，P10，P11，P02，P20，Q01，Q10，Q11，Q02，Q20}

（4）估计全部的状态空间数，并尽可能列出全部的状态空间或予以描述之。

在这个问题世界中，S0 =（3,3,1）为初始状态，S31 = Sg =（0,0,0）为目标状态。全部的可能状态共有 32 个，如表所示。

状态	m,c,b	状态	m,c,b	状态	m,c,b	状态	m,c,b
S0	3 3 1	S8	1 3 1	S16	3 3 0	S24	1 3 0
S1	3 2 1	S9	1 2 1	S17	3 2 0	S25	1 2 0
S2	3 1 1	S10	1 1 1	S18	3 1 0	S26	1 1 0
S3	3 0 1	S11	1 0 1	S19	3 0 0	S27	1 0 0
S4	2 3 1	S12	0 3 1	S20	2 3 0	S28	0 3 0
S5	2 2 1	S13	0 2 1	S21	2 2 0	S29	0 2 0
S6	2 1 1	S14	0 1 1	S22	2 1 0	S30	0 1 0
S7	2 0 1	S15	0 0 1	S23	2 0 0	S31	0 0 0

表 1 传教士和野人问题的全部可能状态

注意：按题目规定条件，应划去非法状态，从而加快搜索效率。

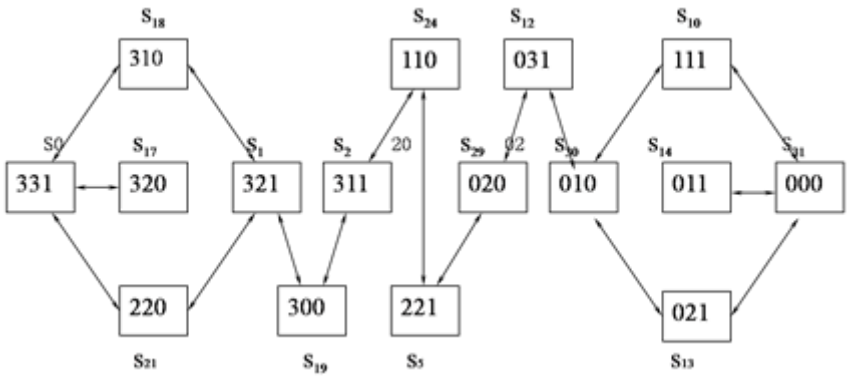
1）首先可以划去左岸边**野人**数目超过传教士的情况，即 S4、S8、S9、S20、S24、S25 等 6 种状态是不合法的；

2）应划去右岸边**野人**数目超过修道士的情况，即 S6、S7、S11、S22、S23、S27 等情况；

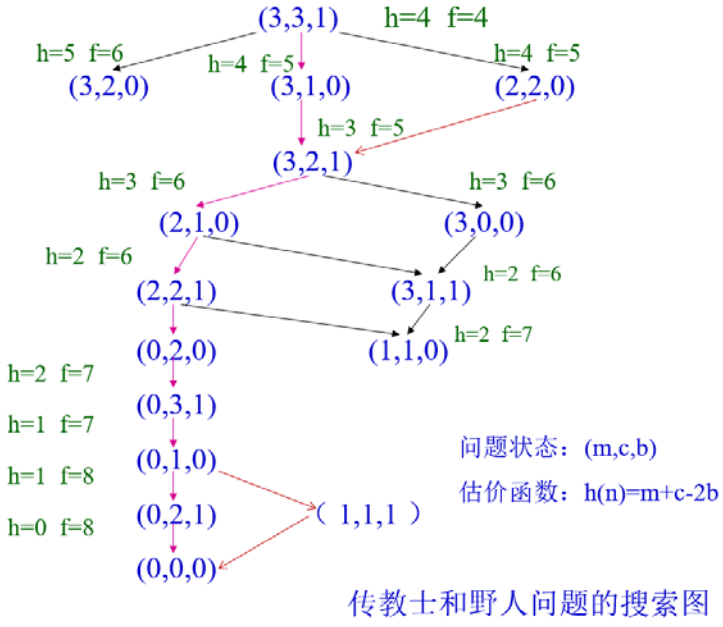
3）应划去 4 种不可能出现状态：划去 S15 和 S16——船不可能停靠在无人的岸边；划去 S3——传教士不可能在数量占优势的**野人**眼皮底下把船安全地划回来；划去 S28——传教士也不可能在数量占优势的**野人**眼皮底下把船安全地划向对岸。可见，在状态空间中，真正符合题目规定条件的**只有 16 个合理状态。**

（5）当状态数量不是很大时，按问题的有序元组画出状态空间图，依照状态空间图搜索求解。

根据上述分析，共有 16 个合法状态和允许的操作，可以划出传教士和食人者问题的状态空间图，如图所示。



B .



传教士和野人问题的搜索图

C.

在讨论用产生式系统求解问题时， 有时引入状态空间图的概念很有帮助。状态空间图是一个有向图， 其节点可表示问题的各种状态(综合数据库)， 节点之间的弧线代表一些操作(产生式规则)， 它们可把一种状态导向另一种状态。这样建立起来的状态空间

图， 描述了问题所有可能出现的状态及状态和操作之间的关系， 因而可以较直观地看出问题的解路径及其性质。实际上只有问题空间规模较小的问题才可能作出状态空间图， 例如 N=3 的 M-C 问题,的其状态空间图如下图所示， 此时采用的控制策略为顺序选取规则。由于每个摆渡操作都有对应的逆操作， 即 pmc 对应 qmc， 所以该图也可表示成具有双向弧的形式。

当空间规模较大的时候就不好求解

容许启发函数不好寻找

- 编写一个程序，当输入两个网页的 URL 后，试着找到从一个网页到另一个网页的链接路径。什么样的搜索策略比较合适？双向搜索是好主意吗？能用搜索引擎实现一个前驱函数？

基本原理:

迭代加深搜索是以 DFS 为基础的，它限制 DFS 递归的层数。

迭代加深搜索的基本步骤是:

1、设置一个固定的深度 depth，通常是 depth = 1，即只搜索初始状态，即只匹配 1 级网页，符合则找到

2、DFS 进行搜索，限制层数为 depth，如果找到答案，则结束，如果没有找到答案 则继续下一步

3、如果 DFS 途中遇到过更深的层，则++depth，并重复 2；如果没有遇到，说明搜 索已经结束，没有答案

1.利用迭代加深搜索比较合适:

如果用 DFS 的话，很可能进入一个网页后无限的深度下去，找不到想要的第二个网页。

利用 BFS，所需空间大

而迭代加深搜索，IDDFS 与广度优先算法是等价的，但对内存的使用会少很多；在每一步迭代中，它会按深度优先算法中的顺序，遍历搜索树中的节点，但第一次访问节点的累积顺序实际上是广度优先的。

2.个人认为双向搜索不是个好主意，因为很可能存在一种情况，某个网页只是通过签一个网页搜索来的，本身不能继续搜索到另一个网页，或者找不到前驱网页，不适用于双向搜索

3.将搜索到的路径存储起来，即作为父节点，通过寻找父节点可以找到前驱函数