**人工智能作业**

**171491419 孟维平**

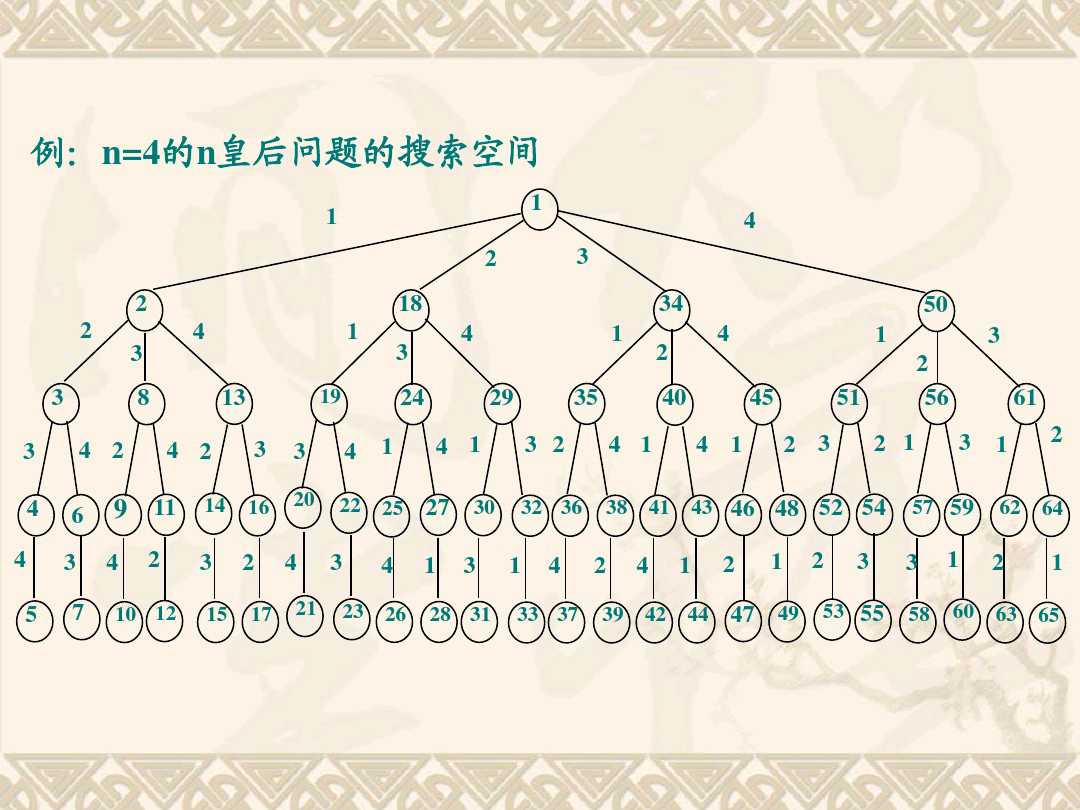
一、4×4皇后问题：

1.确定问题状态：问题的状态即棋盘的布局状态。

2.构造状态空间树：状态空间树的根为空棋盘，每个布局的下一步可能布局是该布局结点的子结点。（可知道，每行中有且只有一个皇后，可以采用逐行布局的方式，也就是说每个布局有4个结点）设某个皇后为，分别在 第i行（i=1，2，3，4）

3.问题的解状态：可以用（1，），（2，）……..表示4个皇后的位置

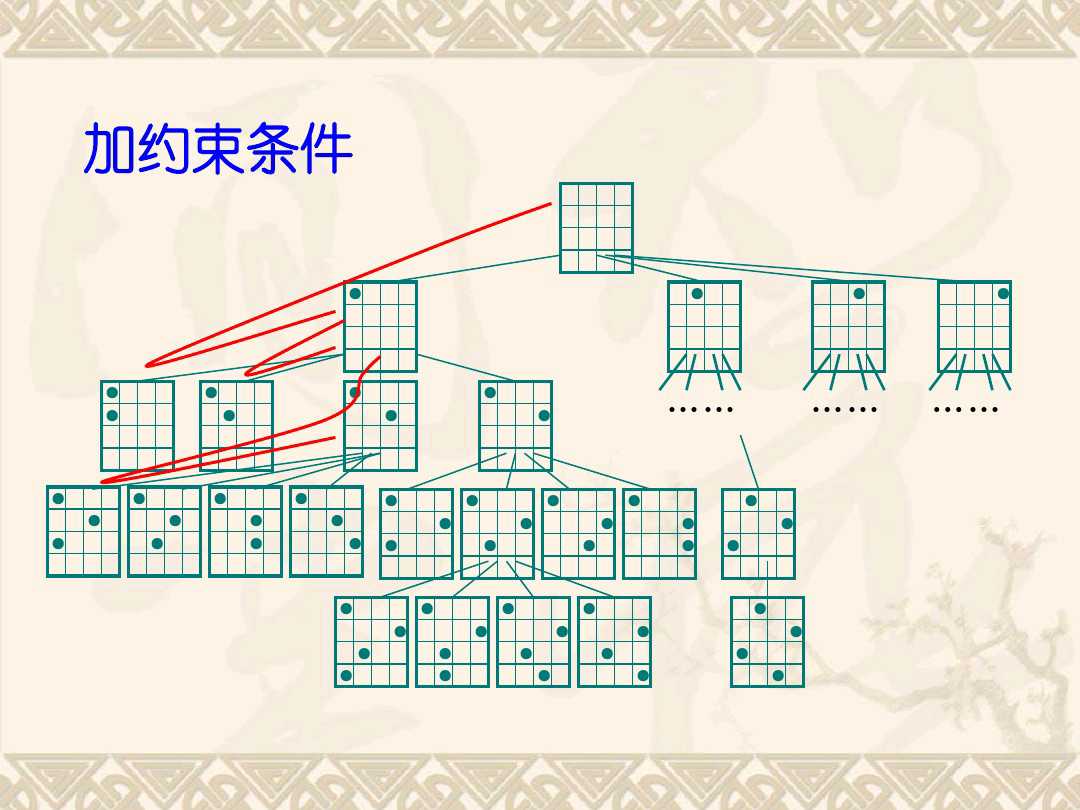
4.问题的解空间：（x1，x2，x3，x4），1<=<=4(i=1，2，3，4)，共4！个状态



5.约束条件：不在同一行，不在同一列，不在同一对角线

6.回溯：当所有的j都不满足条件，第i+1个皇后无法安置时，此时应回溯第i个皇后。

7.加约束条件：



7.搜索解空间，剪枝：

（1）从空棋盘起，逐行放置棋子。

（2）每在一个布局中放下一个棋子

（3）如果当前行上没有可合法放置的棋子的位置，则回溯到上一行。

8.最后的棋盘：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

源代码：

#include <stdio.h>  
#define N 4  
char board[N][N];  
int t;  
int col[N];                 int safetyPlace(int x,int y)   {  
int i,j;  
for(i=0;i<x;i++)  
{  
j=col[i];  
if(x==i||y==j)  
return 0;

return 0;       
}  
return 1;  
}  
void get\_position(int i)  {  
int w,j;  
char a[1]={3};  
if(i==N)               {  
for (w=0;w<N;w++)  
{  
for (j=0;j<N;j++)  
{  
if(board[w][j]==001)  
printf("%c  ",board[w][j]);  
else  
{  
printf("%c",a[0]);  
printf("%c ",board[w][j]);  
}  
}  
printf("\n");  
}  
printf("\n");  
printf("--------------\n");  
t++;  
}  
else  
{  int u;  
for (u=0;u<N;u++)  
{    if (safetyPlace(i,u)==1)  
{  
col[i]=u;

board[i][u]=001;

get\_position(i+1);

col[i]=0;

board[i][u]=0;      } }  
}  
}  
时间复杂度为O(n！)，空间复杂度为O（n）。

2.分析BFS

（1）完整性：如果breadth是有限的，目标存在，BFS一定能找到，目标不存在，且为无限大，则BFS永远不会结束，不会找到目标。

（2）最优性：只有当结点和结点之间的值和结点的深度成正比的时候，尤其是当每一结点到根结点的值相等的时候，广度优先搜索能够得到最优解；如果二者不成正比，那可能得到的解不是最优解。

改进BFS使其得到最优解：找到一个目标结点时，不是立即离开，而记录下目标结点的路径和值，如果有多个目标结点就加以比较，留下较优的结点，把所有可能的路径都搜索完之后，才输出记录的最优路径。

（3）时间复杂度：假设搜索树的状态空间中每个状态都有b个孩子，搜索树的根结点生成第一层的b个结点，每个子结点又生成b个子结点，于是第二层有了个结点，以此类推，可以得到1+b++…..+=O（）个结点总数，其中d为深度，O（）也就是BFS的时间复杂度。

（4）空间复杂度：最好的情况下，每层只包含一个结点，故空间复杂度为O（1），最坏情况下将会储存（-1）个结点和一个无用的树，这时空间复杂度为O（）。