4皇后问题的解

**状态空间**：

利用栈存储第i行的状态，第I行（i<n）,不满足皇后条件的话，则弹出，对新的栈顶皇后行，状态改变，然后在对下一行压栈状态，知道满足，如果最后达到最大值则成功，如果栈底全部遍历过了还不能达到栈顶，则失败，没有方法了，首先想要设计几个函数，对每一个行进行遍历，每一行应该有一个状态，表示是够有皇后

**操作**：

X[i] = X[s]，则第i行和第s行皇后在同一列上

如果第i行的皇后在第j列，第s行的皇后在第t列，即X[i] = j 和 X[s] = t，则只要i - j = s- t 或者 i + j = s + t，说明两个皇后在对角线上，对两个等式进行变换后，得到结论，只要|i- s| = |j - t|（即|i- s| = |x[i] - x[s]|），则皇后在同一对角线上

**N皇后问题源程序：**

#include"stdio.h"

#include"math.h"

#include"malloc.h"

#define ture 1

#define false 0

void Trial(int i, int n, int \*X);

int place(int s, int \*X);

int place(int s, int \*X)

{

int i;

for(i=1; i < s; i++)

{

if(abs(i-s) == abs(X[i]-X[s]) || X[i] == X[s])

{

return false;

}

}

return true;

}

void Trial(int i, int n, int \*X)

{

int j, k;

if(i > n)

{

for(k = 1; k <= n; k++)

{

printf("%d ", X[k]);

}

printf(" \n");

}

else

{

for(j = 1; j <= n; j++)

{

X[i] = j;

if(place(i, X))

{

Trial(i+1, n, X);

}

}

}

}

void main()

{

int n;

int \*Queen = NULL;

printf("请输入皇后的个数.\n");

scanf("%d", &n);

Queen = (int \*)malloc(sizeof(int) \* (n+1));

Trial(1, n, Queen);

}

Completeness：Yes

Optimality：Yes

Time complexly:最坏情况下o（n的n+1次幂）

Space complexly:o(n)

**2.分析深度优先搜索的四个特性**

完整性：完整性是指能否确定问题是否有解，并且能够至少输出一个解。

对于深度优先搜索来说，当树的深度是确定的时候，那么是完整的，但是如果深度是无限的那么就不是完整的，算法会陷入循环之中。

最优性：最优性是指算法能够找到一个所有解中的最优的一个，对于深度优先搜索来说，它不具备最优性，因为它的搜索方向是最开始人工指定的，只有把某一分支遍历到最深的一层，才会搜索另一个分支，在搜索到一个解之后就会停止搜索。

时间复杂度：时间复杂度是指算法的所需操作步骤数量的估计值。深度优先遍历的时间复杂度为，其中b为目标的深度，m为树的深度。

空间复杂的：空间复杂度是指算法在执行操作的时候需要储存数据的大小估计。深度优先遍历的时间复杂度是，其中b为目标的深度，m为树的深度。