Completeness:一个算法是否能过总也找到问题的全部的解

Optimality:一个算法是否能够总也找到这个问题的最优的解

Time complexly（时间复杂度）：一个算法运行所需要的时间

Space complexity（空间复杂度）:一个算法运行所需要的空间

b：搜索树的最大宽度

d:最小成本解的搜索深度

m：状态空间的最大深度

4皇后问题的解

**状态空间**：

将每个可行解表示为一个4维向量<x1,x2,x3,x4>,其中xi表示第i行放置皇后的位置（列号），所有可行解为4维向量构成的集合{<x1,x2,x3,x4> | 1<=xi<=4,1<=i<=4},将它排列为完全四叉树。

**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)