硬件一班 王倩倩 171491121

求解4\*4皇后

**一 问题描述**

在4x4 的棋盘上放四个皇后，要求这四个皇后相互之间不在同一列也不在同一行，也不在同一斜线上。

**二 解决方案**

我们可以先利用穷举法，这样会有4x4x4x4=256种方案，在这里，我们就可以利用回溯法，按列来摆放，先在第一列四个位置任选一个位置放置第一个皇后，然后在第二列的四个位置上分别放上四个皇后，由于第一行第二列和第二行第二列不符合要求了，所以我们就考虑第三行第二列和第四行第二列，然后接着一步一步进行下去....最终我们会得到两种解，

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

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

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

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

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

↓

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

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

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

.....

如下图。这样就可以避免大量的无用功。

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

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

三 算法实现

#include<stdio.h>

int count1=0;

int facs(int i,int j,int (\*Q)[4])

{

int s,t;

for(s=i,t=0;t<4;t++)

{

if(Q[s][t]==1&&t!=j)

return 0;

}

for(t=j,s=0;s<4;s++)

{

if(Q[s][t]==1&&s!=i)

return 0;

}

for(s=i-1,t=j-1;s>=0&&t>=0;s--,t--)

{

if(Q[s][t]==1)

return 0;

}

for(s=i-1,t=j+1;s>=0&&t<4;s--,t++)

{

if(Q[s][t]==1)

return 0;

}

for(s=i+1,t=j+1;s<4&&t<4;s++,t++)

{

if(Q[s][t]==1)

return 0;

}

for(s=i+1,t=j-1;s<4&&t>=0;s++,t--)

{

if(Q[s][t]==1)

return 0;

}

return 1;

}

void queen(int j,int (\*Q)[4])

{

int i,k;

if(j==4)

{

for(i=0;i<4;i++)

{

for(k=0;k<4;k++)

printf("%d ",Q[i][k]);

printf("\n");

}

printf("\n");

count1++;

return;

}

for(i=0;i<4;i++)

{

if(facs(i,j,Q))

{

Q[i][j]=1;

queen(j+1,Q);

Q[i][j]=0;

}

}

}

int main()

{

int Q[4][4];

int i,j;

for(i=0;i<4;i++)

{

for(j=0;j<4;j++)

Q[i][j]=0;

}

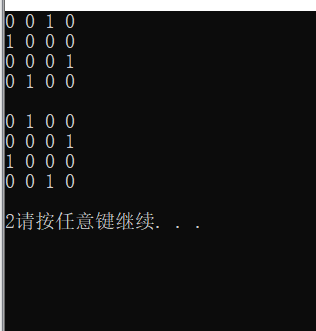
queen(0,Q);

printf("%d",count1);

return 0;

}

四 实验结果



分析深度优先搜索的四个特性

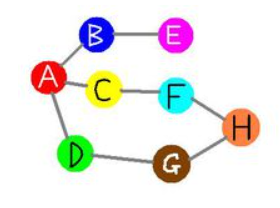
深度优先遍历图的方法是，从图中某顶点v出发：

（1）访问顶点v；

（2）依次从v的未被访问的邻接点出发，对图进行深度优先遍历；直至图中和v有路径相通的顶点都被访问；

（3）若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行深度优先遍历，直到图中所有顶点均被访问过为止。

举例说明：



1. 完整性

举例说明：上图是一个无向图，如果我们从A点发起深度优先搜索（以下的访问次序并不是唯一的，第二个点既

可以是B也可以是C,D），则我们可能得到如下的一个访问过程：A->B->E（回溯到A)->C->F->H->G->D

（没有路，最终回溯到A,A也没有未访问的相邻节点，本次搜索结束）

对于这个无向图来说，我们最终会访问他的所有的顶点。

1. 最优性

深度优先搜索对图进行深度优先遍历，将所要解答的问题划分成若干个阶段或者步骤，找到最优解。

1. 时间复杂度O（b\*）
2. 空间复杂度 O（bm）