backtracking

0）回溯算法：

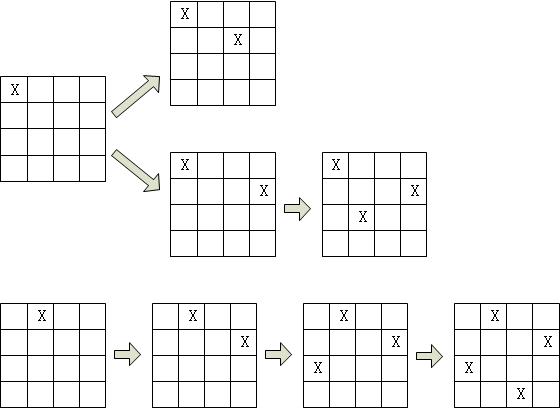
回溯算法也算是遍历算法的一种，回溯算法是对Brute-Force算法的一种改进算法，一个典型的应用是走迷宫问题，当我们走一个迷宫时，如果无路可走了，那么我们就可以退一步，再在其他的路上尝试一步，如果还是无路可走，那么就再退一步，尝试新的路，直到走到终点或者退回到原点。

1）皇后问题：

N皇后问题是指在N\*N的棋盘上放置N个皇后，使这N个皇后无法吃掉对方（也就是说两两不在一行，不在一列，也不在对角线上）。经典的是8皇后问题，这里我们为了简单，以4皇后为例。

首先利用回溯算法，先给第一个皇后安排位置，如下图所示，安排在（1,1）然后给第二个皇后安排位置，可知（2,1），（2,2）都会产生冲突，因此可以安排在（2,3），然后安排第三个皇后，在第三行没有合适的位置，因此回溯到第二个皇后，重新安排第二个皇后的位置，安排到（2,4），然后安排第三个皇后到（3,2），安排第四个皇后有冲突，因此要回溯到第三个皇后，可知第三个皇后也就仅此一个位置，无处可改，故继续向上回溯到第二个皇后，也没有位置可更改，因此回溯到第一个皇后，更改第一个皇后的位置，继续上面的做法，直至找到所有皇后的位置，如下图所示。

这里为什么我们用4皇后做例子呢？因为3皇后是无解的。同时我们也可以看到回溯算法虽然也是Brute-Force，但是它可以避免去搜索很多的不可能的情况，因此算法是优于Brute-Force的。



下面我们来编程实现：

#include<stdio.h>

#include<windows.h>

#define N 8

#define abs(x)(((x)>=0)?(x):-(x))

int col[N+1];

int count=0;

void Output();

void Queen(int i,int n);

void main()

{

int i;

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

{

col[1]=i;

Queen(2,N);

}

printf("%d\n",count);

system("pause");

}

void Queen(int i,int n)

{

if(i>n)

Output();

else

{

int j;

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

{

col[i]=j;

int k=1;//已经安排了位置的皇后的游标指示

while(k<i)//比较现在的皇后与之前的皇后有没有冲突

{

if((col[k]-col[i])\*(abs(col[k]-col[i])-abs(k-i))!=0)//冲突条件

{

k++;

if(k==i)

Queen(i+1,n);

}

else

break;

}

}

}

}

void Output()

{

int i;

count++;

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

{

printf("(%d,%d)\n",i,col[i]);

}

printf("\n");

}

如果要计算8皇后问题，则只需要把上面的N=4改为N=8就可以了。8皇后问题有92中解法，可以试一下。