分而治之：原问题分解为若干子问题，解决子问题，合并问题解

可以使用递归，也可以使用非递归方式实现

递归的两个重要概念：递归边界，递归式

可以画递归图来理解

N!问题

F(N) = N! = 1\*2\*3\*....\*(N-1)\*N

F(N) = F(N-1)\*N

...

F(0) = 1 递归出口

Int F(int n){

If(n==0) return 1;

Else return F(n-1)\*n;

}

Fibonacii数列

Int F(n)

{

If(n == 0 || n==1){

Return 1;

}else {

Return F(n-1) + F(n-2);

}

}

全排列问题

#include<stdio.h>

int n;

const int MaxN = 100;

int P[MaxN];

int hashTable[MaxN] = {false};

void generateP(int index);

int main()

{

n = 3;

generateP(1); //

return 0;

}

void generateP(int index)

{

if(index == n+1) // 递归出口,输出

{

for(int i =1; i<=n;i++)

{

printf("%d",P[i]);

}

printf("\n");

return;

}

for(int x = 1; x<= n;x++)

{

if(hashTable[x] == false){

P[index] = x;

hashTable[x] = true;

generateP(index+1); // 处理index+1;

hashTable[x] = false;

}

}

}

N皇后问题

暴力法：

#include<stdio.h>

#include<math.h>

int n;

const int MAXN = 1000;

int P[MAXN];

int count =0;

bool hashTable[MAXN] = {false};

void generateP(int index);

int main()

{

n = 8;

generateP(1);

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

return 0;

}

void generateP(int index)

{

if(index == n+1){ // 生成一个排列，判断是否合法,只需判断不在一个对角线上即可

int flag = true;

for(int i = 1; i<=n;i++)

{

for(int j = i+1; j<=n;j++){

if(abs(i-j) == abs(P[i]-P[j])){

flag = false;

}

}

}

if(flag)

{

count++;

}

return ;

}

for(int x = 1; x<=n;x++)

{

if(hashTable[x] == false)

{

P[index] = x;

hashTable[x] = true;

generateP(index+1);

hashTable[x] = false;

}

}

}

回溯法：

#include<stdio.h>

#include<math.h>

int n;

const int MAXN = 1000;

int P[MAXN];

int count =0;

bool hashTable[MAXN] = {false};

void generateP(int index);

int main()

{

n = 8;

generateP(1);

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

return 0;

}

void generateP(int index)

{

if(index == n+1){ // 生成一个合法的排列

count++;

return ;

}

for(int x = 1; x<=n;x++)

{

if(hashTable[x] == false)

{

bool flag = true;

for(int pre = 1; pre<index;pre++)

{

if(abs(pre-index) == abs(x-P[pre])){

flag = false; // 有冲突

break;

}

}

if(flag){

P[index] = x;

hashTable[x] = true;

generateP(index+1);

hashTable[x] = false;

}

}

}

}

1. 吃糖果
2. 数列
3. 神奇的口袋

#### **分析：**

题目的意思就是在n个数里面找到和为40的组合个数，和为40的组合种类。

1. 递归方法

* 考虑数组里的一个数a[i], 可以选择加入这个数a[i], 也可以选择不加入
* 当选择要加入a[i]时，组合种类就是count(i + 1, sum - a[i])
* 当选择不加入a[i]时， 组合的种类就是count(i + 1, sum)

1. 八皇后

解题思路：