DFS基本入门模板

出处：<https://blog.csdn.net/qq_40763929/article/details/81629800>

核心代码：关于dfs参数问题，什么在变化，就把什么设置成参数。

void dfs()//参数用来表示状态

{

if(到达终点状态)

{

...//根据题意添加

return;

}

if(越界或者是不合法状态)

return;

if(特殊状态)//剪枝

return ;

for(扩展方式)

{

if(扩展方式所达到状态合法)

{

修改操作;//根据题意来添加

标记；

dfs（）；

(还原标记)；

//是否还原标记根据题意

//如果加上（还原标记）就是 回溯法

}

}

}

**dfs全排列：**

#include<iostream>

#include<cmath>

using namespace std;

int p[10] = { 0 };

bool vis[10] = { 0 };

int n;

void dfs(int x)

{

if (x == n + 1)

{

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

cout << p[i] << " ";

cout << endl;

return;

}

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

{

if (vis[i] == false)

{

p[x] = i;

vis[i] = true;

dfs(x + 1);

vis[i] = false;

}

}

}

int main()

{

n = 4;

dfs(1);

return 0;

}

在全排列的代码基础上，如果要求是顺序输出，则要剪枝，（只需要加一句代码）：

#include<iostream>

#include<cmath>

using namespace std;

int p[10] = { 0 };

bool vis[10] = { 0 };

int n;

void dfs(int x)

{

if (x == n + 1)

{

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

cout << p[i] << " ";

cout << endl;

return;

}

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

{

if (vis[i] == false && i>p[x]) //加上这句代码，也就是剪枝，可以避免下面的递归

{

p[x] = i;

vis[i] = true;

dfs(x + 1);

vis[i] = false;

}

}

}

int main()

{

n = 4;

dfs(1);

return 0;

}