算法 DFS

一、dfs的简要说明

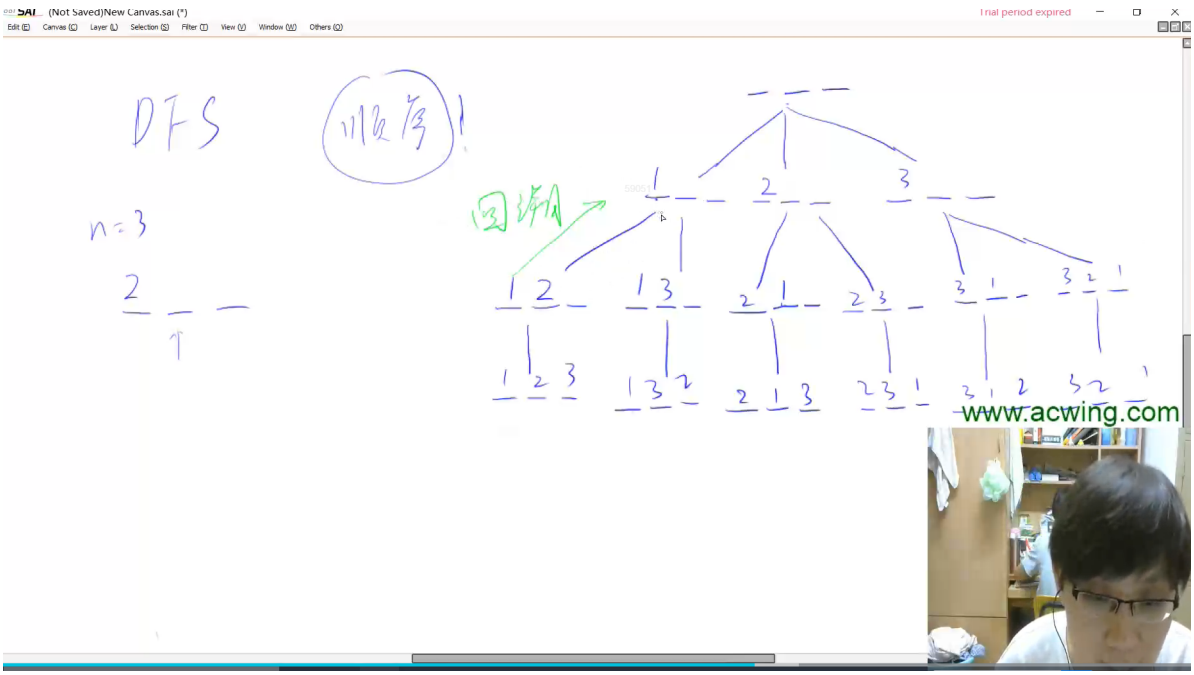
(1)：深度优先搜索（Depth-First-Search）是搜索算法的一种。是沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所有边都已被探寻过，搜索将回溯到发现节点v的那条边的起始节点。这一过程一直进行到已发现从源节点可达的所有节点为止。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被访问为止。属于盲目搜索。

(2)：DFS是图论里面的一种搜索算法，他可以由一个根节点出发，遍历所有的子节点，进而把图中所有的可以构成树的集合都搜索一遍，达到全局搜索的目的。所以很多问题都可以用dfs来遍历每一种情况，从而得出最优解，但由于时间复杂度太高，我们也叫做**暴力搜索**。

(3)：DFS如同数据结构中的栈结构，是属于一种后进先出的结构，比如说一个羽毛球筒，把它装满之后，我们开始使用时，拿的总是最后放进去的那一个。所以这就导致了所有的点进入栈时有一个顺序，我们称之为： **DFS序**。

(4)：根据dfs的英文全写，我们可以知道这是一种深度优先搜索的搜索算法，什么叫做深度优先？意思就是它搜索一颗子树时，它要遍历到底才会 “回头”，比如说：上有向图中的 搜索模式 为(以DFS序来描述)：a->b->e->b->f->c->f->b->c->b->a->c->a->d->g->d->a，有人就会问为什么搜索到c的时候不搜索a呢？因为我们假设的这是一个有向图。而且我们可以看到如果 你面对的图是一个 无向图，这个时候这个树将会持续搜索因为可能会形成环路，使得栈的结构一直进进出出，导致程序崩溃，所以我们也因该注意，在写DFS时，如果面对的是一个无向图的话我们需要进行标记。一般的标记方法有:①这个点的父节点被标记，使得子节点不能回到父节点造成循环②访问过的节点被标记。这两种方法视情况而定。

(5)：对于暴搜来说，因其复杂度太高，我们就会想去优化它的复杂度，这一过程称为剪枝，剪纸分为可行性剪枝和最优化剪枝，关于剪枝引申出一种名为记忆化搜索的方法，该方法与动态规划类似。(对于剪枝来说，我自己也是不太精通，刚入编程界半年，以后搞懂必会写总结)。



y总代码：

#include

using namespace std;

const int N = 10;

int n;
int path[N];

void dfs(int u, int state)
{
if (u == n)
{
for (int i = 0; i < n; i ++) printf("%d ", path[i]);
puts("");

```
    return;
}

for (int i = 0; i < n; i ++ )
    if (!(state >> i & 1))
    {
        path[u] = i + 1;
        dfs(u + 1, state + (1 << i));
    }
```

}

int main()
{
scanf("%d", &n);

```
dfs(0, 0);

return 0;
```

}