DFS（1）

给定一个整数 n，将数字 $1 \sim n$ 排成一排，将会有很多种排列方法。

现在，请你按照字典序将所有的排列方法输出。

输入格式

共一行，包含一个整数 n。

输出格式

按字典序输出所有排列方案，每个方案占一行。

数据范围

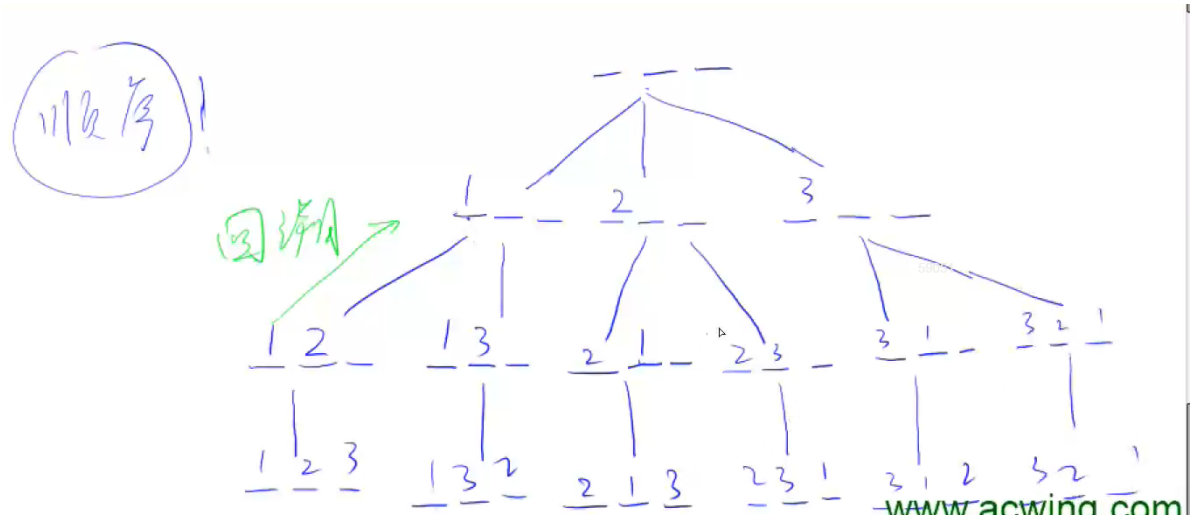
$1 \leq n \leq 7$

输入样例：

```
3
```

输出样例：

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```



1. 此题中dfs表示的含义是：`求出从第u行到最后一行的所有path`。
2. dfs的求法：根据通项公式的含义，假设`已知第u+1行到最后一行的所有path`，综合1和2得出：path[u] 与 path[u+1] 合并后，即为dfs的解。
3. 回溯的特征是：递归的最外层是一个循环。因为一次dfs得到的是所有的path。每一次都是从当前现场中去取得剩下未访问的元素。（这一块自己画个图就很容易理解）。
   `反证：如果不进行现场的恢复，则在第一次完成深搜后，所有元素都已经被访问过了。`这样在回溯到上一层时，上层的现场中的状态都被下层更改了，数据就会乱套。

y总代码：

```
#include <iostream>

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;
}
```


理解代码：

```
#include <iostream>
using namespace std;
const int N = 10;

int path[N];
bool st[N];
int n;

// 计算u->n的所经过的路径path
void dfs(int u)
{
    // 边界条件
    if (u == n)
    {
        for (int i = 0; i < n; i++) printf("%d ", path[i]);
        puts("");
        return;
```

```
    }
    else
    {
        for (int i = 1; i <= n; i++)
        {
            if (!st[i])
            {
                path[u] = i;
                st[i] = true;
                dfs(u+1); // 回溯之后，为何要恢复现场？
                path[u] = 0;
                st[i] = false;
            }
        }
    }
}

int main()
{
    scanf("%d", &n);
    dfs(0);
    return 0;
}
```