1317：组合的输出

排列与组合是常用的数学方法，其中组合就是从n个元素中抽出r个元素(不分顺序且r≤n)，我们可以简单地将n个元素理解为自然数1，2，…，n，从中任取r个数。

现要求你用递归的方法输出所有组合。

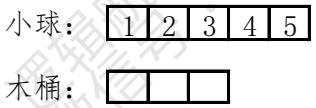
例如n＝5，r＝3，所有组合为：

1 2 3　1 2 4　1 2 5　1 3 4　1 3 5　1 4 5　2 3 4　2 3 5　2 4 5　3 4 5

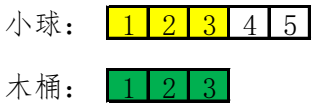
解析

这是一道的基本全排列。现在，我们对这道题进行一下想象：我们面前有一堆号码不同的小球，以及r个桶。我们要做的事情，就是往每一个桶里放一个球。

参考题目中给定的测试数据，我们现在有5个球，3个桶，如下图所示：

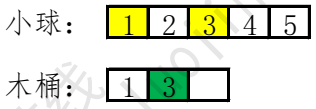


首先，我们开始向木桶中投入小球。前三个很明显是1,2,3，我们用黄色表示这个数字已经被使用过，白色表示未使用。结果如下：



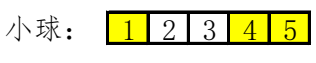
接下来很明显，我们应该把3从小桶中拿出，再依次放入4和5，结果如下：

现在，最后一个桶已经没有数字可以再被使用，此时将发生回溯，即把第二桶的小球进行调整，然后重新向第三个桶内放入小球。我们现在向第二个桶内放入3号球。别忘了，要把2号球标记为未使用。



这时，我们的第三个小桶，就存在以下两种情况。

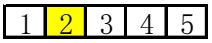
接下来第二个桶变成了4，那么答案只有唯一的1个。结果如下：



木桶：

这个时候同样发生回溯，即，把1号桶里的小球进行调换，结果如下：

小球：



木桶：

注意：当每一个小球从桶中取出的时候，一定要将其标记成为未使用。

现在我们开始把上面的图像转变为代码

首先，我们需要考虑的第一个问题是循环何时终止？
很容易想到的是只要把桶装满即可。所以，我们只需要判断当前桶内已经装入的小球数量。

接下来，我们需要考虑的问题是题目的另一个限制，即后面的数字永远比前面的大。这个就很简单了，我们只要在每一次放置中做一个小小的条件限制即可。

完整代码如下：

编码

```
#include <bits/stdc++.h>

using namespace std;

//排列数组和标记数组
int box[10], book[10];
//n个数字，筛选r个
int n, r;

//深度优先搜索
void dfs(int num) {
    //达到了放置数量的最大上限
    if (num == r + 1) {
        //打印当前排列
        for (int i = 1; i <= r; i++) {
            printf("%3d", box[i]);
        }
        printf("\n");
        return;
    }
    //从数字1开始遍历
    for (int i = 1; i <= n; i++) {
        //当前没有使用过的牌
        if (book[i] == 0) {
            //因为本题要求后一个数字必须大于前一个数字，所以需要在此进行判
            //如果没有本限制，那么就是全排列了。
            if (i > box[num - 1]) {
                //将牌放入盒子中
                box[num] = i;
                //记录当前牌已经使用过
                book[i] = 1;
                //向下一个盒子进行放置
                dfs(num + 1);
                //当前的这个数字已经完全的测试过，需要进行下一个数字的测试
```

```
                //因此必须需将数字收回，此步骤非常重要
                book[i] = 0;
            }
        }
    }
}


int main(int argc, char **argv) {
    cin >> n >> r;
    dfs(1);
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。