# 题目

已有方法rand7可生成1到7范围内的均匀随机整数，试写一个方法 rand10生成1到10范围内的均匀随机整数。

不要使用系统的Math.random()方法。

示例 1:

输入: 1

输出: [7]

示例 2:

输入: 2

输出: [8,4]

示例 3:

输入: 3

输出: [8,1,10]

提示:

rand7 已定义。

传入参数: n 表示 rand10 的调用次数。

进阶:

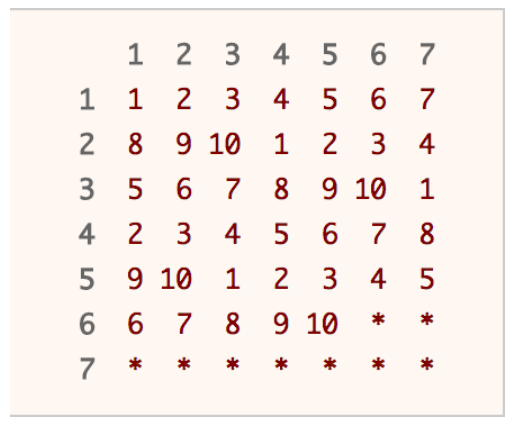
rand7()调用次数的 期望值 是多少 ?

你能否尽量少调用rand7() ?

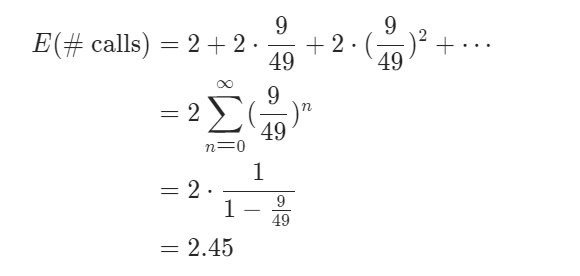
# 分析

## 方法一：拒绝采样

我们可以用拒绝采样的方法实现Rand10()。在拒绝采样中，如果生成的随机数满足要求，那么久返回该随机数，否则会不断生成直到一个满足要求的随机数为止。若我们调用两次Rand7()，那么可以生成[1, 49]之间的随机整数，我们只用到其中的40个，用来实现Rand10()，而拒绝剩下的9个数，如下图所示。



我们来分析这种方法在平均情况下需要调用Rand7()的次数。我们称连续调用两次Rand7()为一轮，在第一轮中，有40/49的概率不被拒绝，而有9/49的概率被拒绝，进入第二轮。在第二轮中也是如此，因此调用Rand7() 的期望次数为：



代码：

class Solution {

public:

int rand10() {

int row, col, idx;

do {

row = rand7();

col = rand7();

idx = col + (row - 1) \* 7;

} while (idx > 40);

return 1 + (idx - 1) % 10;

}

};

复杂度分析：

时间复杂度：期望时间复杂度为O(1)，但最坏情况下会达到 O(∞)（一直被拒绝）。

空间复杂度：O(1)。

## 方法二：拒绝采样

拒绝采样(reject sampling)就是针对复杂问题的一种随机采样方法。

已知rand\_N()可以等概率的生成[1, N]范围的随机数

那么：

(rand\_X() - 1) × Y + rand\_Y() ==>可以等概率的生成[1, X \* Y]范围的随机数，即实现了rand\_XY()。

要实现rand10()，就需要先实现rand\_N()，并且保证N大于10且是10的倍数。这样再通过rand\_N() % 10 + 1就可以得到[1,10]范围的随机数了。

而实现rand\_N()，如下：

(rand7()-1) × 7 + rand7() ==> rand49()

但是这样实现的N不是10的倍数！这该怎么处理？这里就涉及到了“拒绝采样”的知识了，也就是说，如果某个采样结果不在要求的范围内，则丢弃它。

**代码：**

class Solution extends SolBase {

public int rand10() {

while(true) {

int num = (rand7() - 1) \* 7 + rand7();

// 等概率生成[1,49]范围的随机数（其实只使用1-40）

if(num <= 40) return num % 10 + 1;

// 拒绝采样，并返回[1,10]范围的随机数

}

}

}