```
                return;
            }
            printProcess(i + 1, N, true);
            System.out.println(down ? "down " : "up ");
            printProcess(i + 1, N, false);
    }
```

纸条连续对折 n 次之后一定产生 2^{n-1} 条折痕，所以要打印所有的节点，不管用什么方法，其时间复杂度肯定都是 $O(2^n)$，因为解的空间本身就有这么大，但是本书提供的方法的额外空间复杂度为 $O(n)$，也就是这棵满二叉树的高度，额外空间主要用来维持递归函数的运行，也就是函数栈的大小。

蓄水池算法

【题目】

有一个机器按自然数序列的方式吐出球（1 号球，2 号球，3 号球，……），你有一个袋子，袋子最多只能装下 K 个球，并且除袋子以外，你没有更多的空间。设计一种选择方式，使得当机器吐出第 N 号球的时候（$N>K$），你袋子中的球数是 K 个，同时可以保证从 1 号球到 N 号球中的每一个，被选进袋子的概率都是 K/N。举一个更具体的例子，有一个只能装下 10 个球的袋子，当吐出 100 个球时，袋子里有 10 个球，并且 1~100 号中的每一个球被选中的概率都是 10/100。然后继续吐球，当吐出 1000 个球时，袋子里有 10 个球，并且 1~1000 号中的每一个球被选中的概率都是 10/1000。继续吐球，当吐出 i 个球时，袋子里有 10 个球，并且 1~i 号中的每一个球被选中的概率都是 $10/i$，即吐球的同时，已经吐出的球被选中的概率也动态地变化。

【难度】

尉 ★★☆☆

【解答】

这道题的核心解法就是蓄水池算法，我们先说这个算法的过程，然后再证明。

1．处理 1~k 号球时，直接放进袋子里。

2．处理第 i 号球时($i>k$)，以 k/i 的概率决定是否将第 i 号球放进袋子。如果不决定将第 i 号球放进袋子，直接扔掉第 i 号球。如果决定将第 i 号球放进袋子，那么就从袋子里的 k

个球中随机扔掉一个，然后把第 i 号球放入袋子。

3．处理第 i+1 号球时重复步骤 1 或步骤 2。

过程非常简单，但为什么这个过程就能保证从 1 号球到 n 号球中的每一个，被选进袋子的概率都是 k/n 呢？以下是证明过程。

假设第 i 号球被选中($1 \leqslant i \leqslant k$)，那么在选第 k+1 号球之前，第 i 号球留在袋子中的概率是 1。

在选第 k+1 号球时，在什么样的情况下第 i 号球会被淘汰呢？只有决定将第 k+1 号球放进袋子，同时在袋子中的第 i 号球被随机选中并决定扔掉，这两个事件同时发生时第 i 号球才会被淘汰。也就是说，第 i 号球会被淘汰的概率是$(k/(k+1))\times(1/k)=1/(k+1)$，所以第 i 号球留下来的概率就是 $1-(1/(k+1))=k/(k+1)$，这也是 1 号球到第 k+1 号球的过程中，第 i 号球留下来的概率。

在选第 k+2 号球时，什么样的情况下第 i 号球会被淘汰？只有决定将第 k+2 号球放进袋子，同时在袋子中的第 i 号球被随机选中并决定扔掉，这两个事件同时发生时第 i 号球才会被淘汰。也就是说，第 i 号球会被淘汰的概率是$(k/(k+2))\times(1/k)=1/(k+2)$，则第 i 号球留下来的概率就是 $1-(1/(k+2)) = (k+1)/(k+2)$，那么从 1 号球到第 k+2 号球的过程中，第 i 号球留在袋子中的概率是 $k/(k+1)\times(k+1)/(k+2)$。

在选第 k+3 号球时，……。那么从 1 号球到第 k+3 号球的过程中，第 i 号球留在袋子中的概率是 $k/(k+1)\times(k+1)/(k+2)\times(k+2)/(k+3)$。

依此类推，在选第 N 号球时，从 1 号球到第 N 号球的全部过程中，第 i 号球最终留在袋子中的概率是 $k/(k+1)\times(k+1)/(k+2)\times(k+2)/(k+3)\times(k+3)/(k+4)\times\cdots\times(N-1)/N=k/N$。

假设第 i 号被选中($k<i\leqslant k$)，那么在选第 i 号球时，第 i 号球被选进袋子的概率是 k/i。

在选第 i+1 号球时，在什么样的情况下第 i 号球会被淘汰？只有决定将第 i+1 号球放进袋子，同时在袋子中的第 i 号球被随机选中决定扔掉，这两个事件同时发生时第 i 号球才会被淘汰。也就是说，第 i 号球会被淘汰的概率是$(k/(i+1)) \times (1/k) = 1/(i+1)$。那么第 i 号球留下来的概率就是 $1 - 1/(i+1) = i/(i+1)$，那么从 i 号球被选中到第 i+1 号球的过程中，第 i 号球留在袋子中的概率是$(k/i) \times (i/(i+1))$。

在选第 i+2 号球时，从 i 号球被选中到第 i+2 号球的过程中，第 i 号球留在袋子中的概率是$(k/i) \times (i/(i+1)) \times ((i+1)/(i+2))$。

依此类推，在选第 N 号球时，从 i 号球被选中到第 N 号球的过程中，第 i 号球最终留在袋子中的概率是$(k/i) \times (i/(i+1)) \times ((i+1)/(i+2)) \times\cdots \times (N-1)/N = k/N$。

综上所述，按照步骤 1~3 操作，当吐出球数为 N 时，每一个球被选进袋子都是 k/N。

具体过程请参看如下代码中的 getKNumsRand 方法。

```
// 一个简单的随机函数，决定一件事情做还是不做
public int rand(int max) {
        return (int) (Math.random() * max) + 1;
}

public int[] getKNumsRand(int k, int max) {
        if (max < 1 || k < 1) {
                return null;
        }
        int[] res = new int[Math.min(k, max)];
        for (int i = 0; i != res.length; i++) {
                res[i] = i + 1; // 前 k 个数直接进袋子
        }
        for (int i = k + 1; i < max + 1; i++) {
                if (rand(i) <= k) { // 决定 i 进不进袋子
                        res[rand(k) - 1] = i; // i 随机替掉袋子中的一个
                }
        }
        return res;
}
```

设计有 setAll 功能的哈希表

【题目】

哈希表常见的三个操作是 put、get 和 containsKey，而且这三个操作的时间复杂度为 $O(1)$。现在想加一个 setAll 功能，就是把所有记录的 value 都设成统一的值。请设计并实现这种有 setAll 功能的哈希表，并且 put、get、containsKey 和 setAll 四个操作的时间复杂度都为 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

加入一个时间戳结构，一切问题就变得非常简单了。具体步骤如下：

1. 把每一个记录都加上一个时间，标记每条记录是何时建立的。
2. 设置一个 setAll 记录也加上一个时间，标记 setAll 记录建立的时间。