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。

3. 查询记录时，如果某条记录的时间早于 setAll 记录的时间，说明 setAll 是最新数据，返回 setAll 记录的值。如果某条记录的时间晚于 setAll 记录的时间，说明记录的值是最新数组，返回该条记录的值。

具体请参看如下的 MyHashMap 类。

```
public class MyValue<V> {
        private V value;
        private long time;

        public MyValue(V value, long time) {
                this.value = value;
                this.time = time;
        }

        public V getValue() {
                return this.value;
        }

        public long getTime() {
                return this.time;
        }
}

public class MyHashMap<K, V> {
        private HashMap<K, MyValue<V>> baseMap;
        private long time;
        private MyValue<V> setAll;

        public MyHashMap() {
                this.baseMap = new HashMap<K, MyValue<V>>();
                this.time = 0;
                this.setAll = new MyValue<V>(null, -1);
        }

        public boolean containsKey(K key) {
                return this.baseMap.containsKey(key);
        }

        public void put(K key, V value) {
                this.baseMap.put(key, new MyValue<V>(value, this.time++));
        }

        public void setAll(V value) {
                this.setAll = new MyValue<V>(value, this.time++);
        }

        public V get(K key) {
           if (this.containsKey(key)) {
                if (this.baseMap.get(key).getTime() > this.setAll.getTime()) {
                        return this.baseMap.get(key).getValue();
```

```
                } else {
                        return this.setAll.getValue();
                }
            } else {
                return null;
            }
        }
    }
```

最大的 leftMax 与 rightMax 之差的绝对值

【题目】

给定一个长度为 N（N>1）的整型数组 arr，可以划分成左右两个部分，左部分为 arr[0..K]，右部分为 arr[K+1..N-1]，K 可以取值的范围是[0,N-2]。求这么多划分方案中，左部分中的最大值减去右部分最大值的绝对值中，最大是多少？

例如：[2,7,3,1,1]，当左部分为[2,7]，右部分为[3,1,1]时，左部分中的最大值减去右部分最大值的绝对值为 4。当左部分为[2,7,3]，右部分为[1,1]时，左部分中的最大值减去右部分最大值的绝对值为 6。还有很多划分方案，但最终返回 6。

【难度】

校 ★★★☆

【解答】

方法一：时间复杂度为 $O(N^2)$，额外空间复杂度为 $O(1)$。这是最笨的方法，在数组的每个位置 i 都做一次这种划分，找到 arr[0..i]的最大值 maxLeft，找到 arr[i+1..N-1]的最大值 maxRight，然后计算两个值相减的绝对值。每次划分都这样求一次，自然可以得到最大的相减的绝对值。具体请参看如下代码中的 maxABS1 方法。

```
public int maxABS1(int[] arr) {
        int res = Integer.MIN_VALUE;
        int maxLeft = 0;
        int maxRight = 0;
        for (int i = 0; i != arr.length - 1; i++) {
                maxLeft = Integer.MIN_VALUE;
                for (int j = 0; j != i + 1; j++) {
                        maxLeft = Math.max(arr[j], maxLeft);
                }
                maxRight = Integer.MIN_VALUE;
```