1

(1)

构造后的散列表：

下标	0	1	2	3	4	5	6	7	8	9	10
关键字	22		41	30	1	53	46	13	67		

平均查找长度为

$$\frac{1}{8}(1+1+1+1+2+2+2+6)=2.$$

(2)

构造后的散列表：

下标	0	1	2	3	4	5	6	7	8	9
关键字	30	13	67	41	1		22		46	53

其平均查找长度为

$$\frac{1}{8}(1+1+1+1+1+3+2+2)=1.5,$$

小于 $N=11$ 时.

2

(1)

根据算法说明，容易计算出探查序号 i 对应的探查位置为 $h(k)+\frac{i(i+1)}{2}$，因此这个算法属于二次探查，对应 $c_1=c_2=\frac{1}{2}$.

(2)

显然，m 次查找能探查到表中的每个位置，当且仅当 $i\in\{0,1,\cdots,m-1\}$ 时 $h(k,i)$ 互不相同，我们接下来证明这一点.

反之，设 $m=2^t$，并设

$$h(k,i)\equiv h(k,j)\mod m,$$

其中 $0\leq i<j\leq m-1$.

整理得

$$(j-i)(j+i+1)\equiv 0\mod 2^{t+1}.$$

于是由 $j-i$ 和 $j+i+1$ 一奇一偶即知矛盾.

3

设 $\max(a)-\min(a)=D$.

首先，对于给定的值 x，我们给出一个 $O(N)$ 地求出矩阵中落入 $[\min(a), x]$ 的元素的个数的算法. 先给出其 C++ 代码实现：

```
int A[N][N];
int count(int x){
    int i = n-1, j = 0, res = 0;
    while (i && j < n)
        if (A[i][j] <= x) res += i+1, ++j;
        else --i;
    return res;
}
```

这个算法充分利用了矩阵的有序性. 我们初始化 (i,j) 指向矩阵左下角元素，如果该元素小于等于 x，则其所在列所有元素都小于等于 x，该列处理结束；如果该元素大于 x，则其所在行所有元素都大于 x，该行处理结束.

像这样做下去，至多 $2N$ 次比较就能处理完整个矩阵，因此这个算法是 $O(N)$ 的.

回到该题，初始化 $l=\min(A), r=\max(A), mid=\frac{l+r}{2}$. 我们先计算出矩阵中落入 $[\min(a), mid]$ 的元素的个数 m.

如果 $m>K$，整个矩阵中第 K 小的元素就落在 $[l, mid]$ 中，令 $r=mid$；如果 $m=K$，mid 即为所求；如果 $m>K$，整个矩阵中第 K 小的元素就落在 $[mid+1, r]$ 中，令 $l=mid+1$.

像这样做下去，每次区间长度减少一半，最坏到 $l==r$ 时算法终止，此时运行了 $O(\log D)$ 轮，因此整个算法的时间复杂度为 $O(N\log D)$.

最后给出 C++ 代码实现：

```
int A[N][N];
int count(int x);

int find_kth_smallest(int K){
    int l = A[0][0], r = A[N-1][N-1], mid, t;
    while (l < r){
        mid = (l+r)/2;
        t = count(mid);
        if (t == K) return mid;
        if (t < K) l = mid+1;
        else r = mid;
    }
    return l;
}
```

4

设随机数的个数为 N，随机数的范围为 $[1, M], M \approx 10N$.

算法一：标记.

开辟一个大小为 M 的布尔数组，初始化为零. 遍历所有随机数，将布尔数组的对应位置标记为 `true`. 遍历后布尔数组中为 `false` 的位置就对应未出现的数.

算法二：排序.

将随机数从小到大排序，再遍历一次即可得到所有未出现的数.

算法三：哈希.

开辟一个大小为 K 的哈希表，采用开散列存储. 将所有随机数加入散列表，再对 $[1, M]$ 的每个数做一次查询即可.

算法	**标记**	**排序**	**哈希**
平均时间	$O(M+N)$	$O(N\log N+M)$	$O(N+M\frac{N}{K})$
额外空间	$O(M)$	$O(1)$	$O(N)$