提高组模拟赛

Day1

September 30, 2019

题目名称	前缀	快速排序	果实摘取
目录	prefix	qsort	garden
可执行文件名	prefix	qsort	garden
输入文件名	prefix.in	qsort.in	garden.in
输出文件名	prefix.out	qsort.out	garden.out
每个测试点时限	$1s$	$1s$	$2s$
内存限制	512MB	512MB	512MB
试题总分	100	100	100
测试点数目	10	20	10
每个测试点分值	10	5	10
是否有部分分	否	是	否
题目类型	传统型	传统型	传统型

提交的源程序文件名

对于 C++ 语言	prefix.cpp	qsort.cpp	garden.cpp
对于 C 语言	prefix.c	qsort.c	garden.c

编译开关

对于 C++ 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm

1 前缀

1.1 题目描述

有一个 N 个非空字符串的集合 $S = \{S_1, S_2, ...S_n\}$

考虑任意一个由 M 个非空字符串组成的集合 $P = P_1, P_2, ..., P_m$，如果在以下条件下 S 可以恰好分成 M 个子集：

1. $i = 1...M : T_i = \{S_j \mid P_i \sqsubseteq S_j\}$

2. $\forall i = 1...M : T_i \neq \emptyset$

3. $\forall i = 1...M, j = 1...M : T_i \cap T_j = \emptyset$

4. $\cup T = S$

其中 $P_i \sqsubseteq S_j$ 表示 P_i 为 S_j 的前缀。

我们就称 P 为 S 的前缀集合。

现在要求出有多少个集合 P 为 S 的前缀集合，对 $10^9 + 7$ 取模。

1.2 输入格式

从文件 prefix.in 中读取数据。

第一行两个整数 N, M，描述 S 集合、P 集合的大小。

第二行到第 $N + 1$ 行，每行一个字符串 S_i，意义见题目描述。

1.3 输出格式

输出到文件 prefix.out 中。

输出一行一个整数 Ans，表示前缀集合的个数对 $10^9 + 7$ 取模的结果。

1.4 样例 1 输入

```
4 2
aba
cab
cad
abcd
```

1.5 样例 1 输出

```
4
```

1.6 样例解释

$P = \{a, c\}, \{ab, c\}, \{a, ca\}, \{ab, ca\}$，答案为 4.

1.7 样例 2

见下发文件 ex_prefix2.in/ans。

1.8 数据范围和约定

对于 20% 的数据，$N, M \leq 16$

对于 40% 的数据，$N, M \leq 200$

对于 100% 的数据，$1 \leq M \leq N \leq 2000$，每个字符串长度 ≤ 200。

2 快速排序

2.1 题目描述

小 D 刚学习了快速排序，觉得这个算法很高明，这是一个将 N 个数升序排列的算法。

小 D 又听说，如果随机种子取的不好，这个算法会被卡成 $O(N^2)$，于是他写了如下代码来验证这件事：

```
int qsort(int L, int R) {
    if (L >= R) return 0;
    int i = L, j = R;
    int x = Random.Next();
    int x0 = A[x];
    assert(L <= x && x <= R);
    while (i < j) {
        while (A[i] < A[x]) i++;
        while (A[j] > A[x]) j--;
        if (i <= j) {
            if (i == x) x = j;
            else if (j == x) x = i;
            swap(A[i], A[j]);
            i++; j--;
        }
    }
    swap(A[x], A[x0]);
    return R - L + qsort(L, x0 - 1) + qsort(x0 + 1, R);
}
```

在最后一行, 函数从左到右执行：先执行 qsort(L, x0 - 1)，再执行 qsort(x0 + 1, R)；

给定排序数组的长度 N，和可能用到的前 N 个随机数。小 D 想知道对于所有的长度为 N 的排列，qsort(1,N) 可能的最大返回值是多少。

注意在小 D 的程序中，随机数并不取模，所以可能出现 x 越界的情况，即排序可能中途被 assert(L <= x && x <= R); 语句终止，此时排序结束且没有返回值。

（你可以在下发文件中的 sample_qsort.cpp 中找到该程序）。

2.2 输入格式

从文件 qsort.in 中读取数据。

第一行两个整数 N，表示排序序列的长度。

第二行 N 个整数 $Random_1, Random_2, ..Random_N$，表示可能用到的前 N 个随机数。

2.3 输出格式

输出到文件 qsort.out 中。

如果对于所有的排列，qsort(1,N) 都没有返回值，输出一行'No solution'。（注意大小写，不包含"）；

否则输出三行：

在第一行输出'Solution exists'；

在第二行输出可能的最大返回值；

在第三行输出一个长度为 N 的排列，使其被 qsort(1,N) 调用时返回值为可能的最大返回值。（如果有多解，输出任意一组）

2.4 样例 1 输入

```
3
1 2 3
```

2.5 样例 1 输出

```
Solution exists
3
1 2 3
```

2.6 样例解释

初始序列为 $\{1,2,3\}$；

{1,2,3} 经过 qsort(1,3) 后变为 {} 以及 $\{2,3\}$，$R-L=2$。

{2,3} 经过 qsort(2,3) 后变为 {} 以及 $\{3\}$，$R-L=1$

所以返回值为 3，可以证明这是最大的返回值。

2.7 样例 2 输入

```
7
1 7 1 7 1 7 1
```

2.8 样例 2 输出

```
No solution
```

2.9 样例 3

见下发文件 ex_qsort3.in/ans。

2.10 数据范围和约定

对于 20% 的数据：$N \leq 8$；

对于 35% 的数据：$N \leq 16$；

对于另外 15% 的数据：保证 $Random_i = i$；

对于另外 15% 的数据：保证 $Random_i$ 都相同；

对于 100% 的数据，$N \leq 50$，$1 \leq Random_i \leq N$。

此外，如果你的程序正确输出了前两行，可以得到 40% 的分数。（注意此时你仍需要在第三行输出 N 个数）。

3 果实摘取

3.1 题目描述

小 D 的家门口有一片果树林，果树上果实成熟了，小 D 想要摘下它们。

为了便于描述问题，我们假设小 D 的家在二维平面上的 $(0,0)$ 点，所有坐标范围的绝对值不超过 N 的整点坐标上都种着一棵果树。（$(0,0)$ 这个点没有果树）

小 D 先站在 $(0,0)$ 处，正对着 $(1,0)$ 的方向。

每次摘果实时，小 D 会逆时针选择他能看到的第 K 棵还未摘取果实的果树，然后向着这个方向走去，在行走的过程中摘下沿路的所有的果树上的果树果实，直到走到果树林的边缘。

接下来，小 D 回到 $(0,0)$ 处，正对着上一次摘果实的果树的方向。

小 D 会重复这个过程，直到所有的果实都被摘取，小 D 感兴趣的是，最后一棵被摘下果实的果树是哪一棵？

注意小 D 不能看到被任何其他果树遮挡着的果树。

3.2 输入格式

从文件 garden.in 中读取数据。

一行两个整数，N, K，意义见题面。

3.3 输出格式

输出到文件 garden.out 中。

一行两个整数，表示最后一棵被摘下果实的果树的坐标。

3.4 样例 1 输入

```
2 2
```

3.5 样例 1 输出

```
2 0
```

3.6 样例解释

第一圈后摘下果实的树的情况：（数字表示被摘下的时间）

```
* 3 * 2 *
```

```
4  *  *  *  1
*  *  0  *  *
5  *  *  *  8
*  6  *  7  *
```

整个果树林被摘下的情况：

```
12 3  18 2  10
4  11 17 9  1
22 21 0  23 24
5  13 19 15 8
14 6  20 7  16
```

所以最后一棵被摘下的果树为 $(2, 0)$

3.7 样例 2 输入

```
10 4
```

3.8 样例 2 输出

```
-8 -6
```

3.9 样例 3

见下发文件 ex_garden3.in/ans。

3.10 数据范围和约定

对于 10% 的数据，$N \leq 8$

对于 30% 的数据，$N \leq 500$

对于另外 20% 的数据，$K = 2$

对于 100% 的数据，$1 \leq N, K \leq 10^5$