DP 与搜索的极致结合，线段树，字符串 Translate:USACO/picture

Picture 矩形周长

译 by Felicia Crazy

描述

N(N<5000) 张矩形的海报，照片和其他同样形状的图片贴在墙上。它们的边都是垂直的或水平的。每个矩形可以部分或者全部覆盖其他矩形。所有的矩形组成的集合的轮廓称为周长。写一个程序计算周长。

图 1 是一个有 7 个矩形的例子：

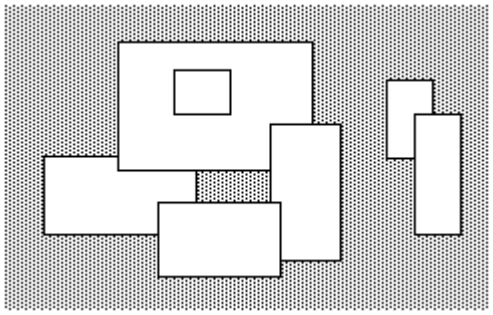


图 1.一个 7 个矩形的集合

对应的轮廓为图 2 所示的所有线段的集合：

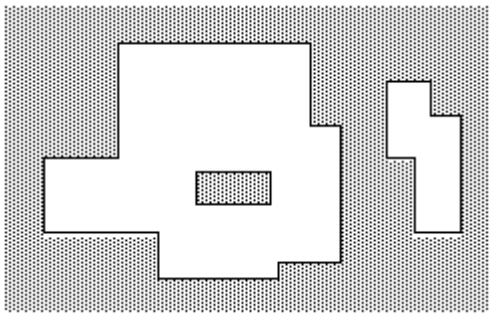


图 2. 矩形集合的轮廓

所有矩形的顶点坐标均为整数。所有的坐标都在 [-10000，10000] 的范围内，并且任何一个矩形面积都为整数。结果的值可能需要 32 位有符号整数表示。

格式

PROGRAM NAME: picture

INPUT FORMAT:

(file picture.in)

第 1 行: N，张贴在墙上的矩形的数目。 第 2..N+1 行 接下来的 N 行中，每行都有两个点的坐标，分别是矩形的左下角坐标和右上角坐标。每一个坐标由 X 坐标和 Y 坐标组成。

OUTPUT FORMAT:

(file picture.out) 只有一行，为一个非负整数，表示输入数据中所有矩形集合的轮廓长度。

SAMPLE INPUT

```
7
-15 0 5 10
-5 8 20 25
15 -4 24 14
0 -6 16 4
2 15 10 22
30 10 36 20
34 0 40 16
```

SAMPLE OUTPUT

```
228
```

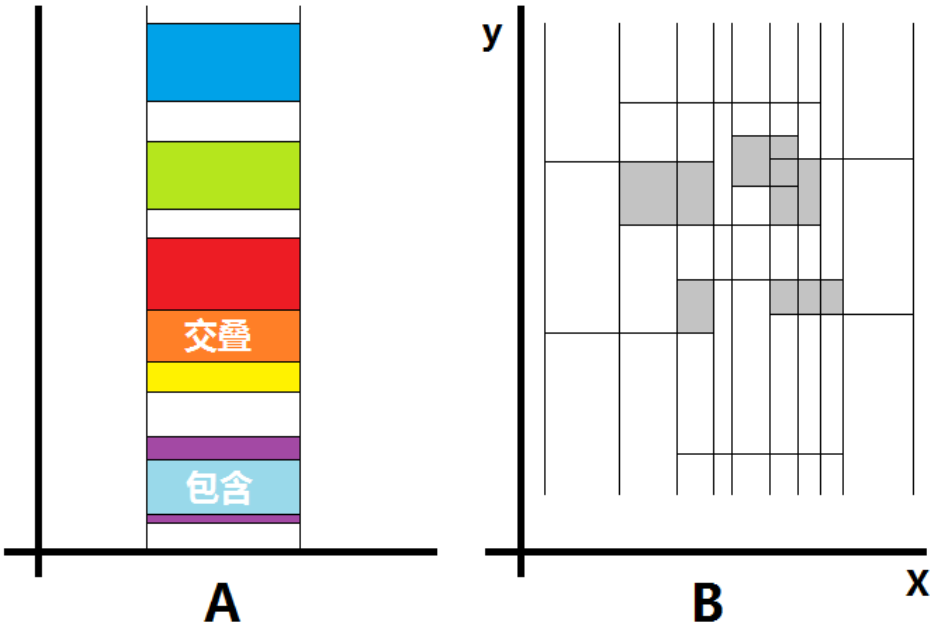
解题思路：

其实边界需要统计的是上下左右 4 个方向没有被覆盖到的边长的总和。这个问题的难点是怎样使得需要判断检索重叠的复杂度尽量小，我们先简化问题，假设所有矩形都是 A 情况的，也就是宽度都一样并且垂直排列，这样的话我们就将矩形的情况分为不相交，交叠，包含几种情况。先从横向的边开始看，如图所示，如果只考虑蓝色和浅绿色这两个矩形因为没有任何与其相交的其他矩形，直接统计横向边长，然而交叠和包含要怎么办呢。如图红色和黄色的矩形有橙色的部分，那么也就是说我们只用统计红色的上边长和黄色的下边长了。而浅蓝色和紫色的关系也是类似只用统计紫色的两个边。

从这里我们得到一些启发，假设所有边长都是 1 个单位长度，显然我们只是统计一个覆盖域的上下两个边界，那么这从抽象意义上很类似括号匹配的问题，下面的这些矩形块可以抽象成()(())((()))(()())注意观察它们的特点，可以发现跟真实的配对没有关系，也就是()()(())((())，我们用一个栈依次压入括号，每次出现()的配对就出栈。然后要统计这些，当压入后(且栈只有这一个元素)时计数器加一，当弹出元素后栈空时计数器加一。这样我们就相当于统计出了所有的该纳入计算的边界，由这个思考点出发，我们要对所有的矩形边进行离散化，同时要区别上下边界，抽象成“左右括号”。这个时候有个细节，对于横向边我们按纵标排序，如果纵标一样谁在前？其实这也是交叠的一种情况，如果纵标一样，先要把左括号排在前，也就是如果我们按纵标由小到大排，下边界应在前，不然会使得周长变大。

具体实现时我们看 B 图，因为配对不只是一种情况，而且坐标很大，这时候我们可以使用线段树来辅助数据结构的部分，需要横着纵着处理两次。也就是对于类似括号匹配中栈的操作，真正实现用的是线段树。

这样本问题就解决了，算法是**离散化+线段树**。这个算法的正确性是我们已经排好了序，而且每个方向的边都是成对出现的。



Translate:USACO/hidden

Hidden Password 隐藏口令

译 by KD

描述

有时候程序员有很奇怪的方法来隐藏他们的口令。Billy"Hacker"Geits 会选择一个字符串 S（由 L 个小写字母组成，5<=L<=100,000），然后他把 S 顺时针绕成一个圈，每次取一个做开头字母并顺时针依次取字母而组成一个字符串。这样将得到一些字符串，他把它们排序后取出第一个字符串。把这个字符串的第一个字母在原字符串中的位置做为口令。

第一个字母所在的位置是 0

如字符串 alabala，按操作得到 7 个字符串，排序后得：

aalabal abalaal alaalab alabala balaala laalaba labalaa

第一个字符串为 aalabal，这个 a 在原字符串位置为 6，则 6 为口令。当字符串相同时，输出编号最小的（即对于 aaaa 输出 0）

格式

PROGRAM NAME: hidden

INPUT FORMAT:

(file hidden.in)

INPUT FORMAT 第一行：一个数：L

第二行：字符串：S(每行 72 个字符，最后一行可能不够）

OUTPUT FORMAT:

(file hidden.out) 一行，为得到的口令

SAMPLE INPUT

```
7
alabala
```

SAMPLE OUTPUT

```
6
```

解题报告：

本题目的方法很朴素，直接暴力。因为字符串长度是 N 所以最多只有 N 个不同的串分别以第 N 个位置的字母开头，我们同时比较两个串也就是一开始从位置 i=0 和位置 j=1 开始比较，S[i]表示以第 i 个字符开头的字符串前缀，长度从 1 到 N 递增。如果 S[i]等于 S[j]，则长度加 1 继续比较；如果 S[i] > S[j]则令 i=j，j=j+1；如果 S[i]<S[j]则令 j+=前缀长度，i 不变。
解释如下:
bcbcabcbck 这里面抱哈两个 bcbc 子串，开始 S[i]='b',S[j]='c'因为'c'>'b'所以 j 加等于前缀长度 1。下一次变动是 S[i]='bcb',S[j]='bca',这时因为 S[j]<S[i]所以 i=j，j=j+1，再然后是 S[i]='b',S[j]='a'。如此往复交替，最后当 j>n 时的 i 就是最终答案，因为这时候已经完成一轮匹配了，后面不需要重复作业了。

Translate:USACO/twofive

Two Five 贰五语言

译 未知,来自 OIBH

描述

有一种奇怪的语言叫做“贰五语言”。它的每个单词都由 A～Y 这 25 个字母各一个组成。但是，并不是任何一种排列都是一个合法的贰五语言单词。贰五

语言的单词必须满足这样一个条件：把它的 25 个字母排成一个 5*5 的矩阵，它的每一行和每一列都必须是递增的。比如单词 ACEPTBDHQUFJMRWGKNSXILOVY，它排成的矩阵如下所示：

```
A C E P T
B D H Q U
F J M R W
G K N S X
I L O V Y
```

因为它的每行每列都是递增的，所以它是一个合法的单词。而单词 YXWVUTSRQPONMLKJIHGFEDCBA 则显然不合法。 由于单词太长存储不便，需要给每一个单词编一个码。编码方法如下：从左到右，再从上倒下，可以由一个矩阵的得到一个单词，再把单词按照字典顺序排序。比如，单词 ABCDEFGHIJKLMNOPQRSTUVWXY 的编码为 1，而单词 ABCDEFGHIJKLMNOPQRSUTVWXY 的编码为 2。

现在，你需要编一个程序，完成单词与编码间的转换。

格式

PROGRAM NAME: twofive

INPUT FORMAT:

(file twofive.in)

INPUT FORMAT 第一行为一个字母 N 或 W。N 表示把编码转换为单词，W 表示把单词转换为编码。 若第一行为 N，则第二行为一个整数，表示单词的编码。若第一行为 W，则第二行为一个合法的单词。

OUTPUT FORMAT:

(file twofive.out) 每行一个整数或单词。

SAMPLE INPUT

INPUT1

```
N
2
```

INPUT2

```
W
ABCDEFGHIJKLMNOPQRSUTVWXY
```

SAMPLE OUTPUT

OUTPUT 1

```
ABCDEFGHIJKLMNOPQRSUTVWXY
```

OUTPUT 2

```
2
```

解题报告：

本质思想很像 3.2.2 的 kimbits。但是分析起来难多了，首先使用 DFS 的解题策略很好想，也容易编写，但是会超时。因此我们将分析 kimbits 的思路用在这道题目上，求出 DP 方程的值并且使用 DP 构造出解。现在的难点是如果这道题目使用 DP 那么状态转移方程要怎么定义。因为如果解决了这个核心问题，构造出解就只是程序实现模块的事情了。
根据题目要求，单词一定要合法，不是合法的单词不考虑。首先是求出字符串的排位，字符串要按照字典序排位，我们可以先求出不同字符为前缀的累加值。例如以 ACB 开头的单词的个数必定是 0，而如果我们要求单词的排位，单词是 ACF.....那么我们首先要统计所有 AB 开头的，ACD 开头的，ACE 开头的.....最后加 1，如此我们就可以计算出 ACF 的排位是多少。那如果是给出排位 n 求字符串，还是要先计算累加值，先计算以 AB 开头的。假如当前的累加值 TOTAL >= n，说明第二个字母就是 B(第一个字母永远是 A)，否则继续累加 AC 开头的.....直到 TOTAL >= n 为止。然后减去确定当前字母后的最后累加的值(即 TOTAL -=

TOTAL',TOTAL'是确定字母后最后被加入 TOTAL 的那个数字),然后继续用这个办法迭代出第 3，4...个字母直到结束。
以上，我们就有了大体的思路。现在要如何去计算这个累加值呢。或者说给定字符串前缀后，求以此开头的字符串共有多少个。设状态转移方程
DP[x1][x2][x3][x4][x5](其中 5>=x1>=x2>=x3>=x4>=x5>=0)表示对于 ABCDEFGHI.....等合法字符串取出前(x1+x2+x3+x4+x5)个字符分别在第 i 行放置 xi 个，并且以这个矩阵为开头共有多少个结果。显然这个开头的最右边连成的边界斜率非负(如下示例),因为如果我们以这个为开头必须保证矩阵满足题目给出的性质，因为我们是按照顺序提取字符的，所以，斜率是负则很可能会出现填入新的字符后，这个新填入的字符比他下方的要大。然后当前要填入的是(x1+x2+x3+x4+x5+1)个字符，它显然只能够贴着已经放过的位置放，也就是它的上方和左方都已经有了单词，也就是示例中划横线的部分（默认第 0 行与第 0 列全满）。把这个位置叫做合法位置。

A B C - *
D E - * *
F - * * *
- * * * *
* * * * *

接着是用记忆化搜索解出 DP。方法是先枚举每一个前缀，递归的时候枚举所有合法位置。边界条件，DP[5][5][5][5][5]=1。因为我们每次递归都会向前**推进一个**字符，所以已经搜索过的并且以后必须要固定的字符我们记录一下。比如 ACF.....当我们搜索完 AB 开头的并且开始搜索以 ACD 开头的字符串的时候,其实字符'C'必须要固定了，以后不可以再使用这个字符了，如果递归到已固定字符则跳过。同时维护当前行和列的最大字符，因为行和列都必须是递增的，所以我们应该随时更新每一行和每一列的最值。