どんどん増えていくように区切る

問題の概要

`23342020662` のように、先頭が 1〜9 のいずれかで、 残りは 0〜9 が並んだ文字列があります。
この文字列に区切りを入れ、 **値がどんどん増えていく数列** (狭義単調増加列)にすることを考えます。
上記の例の場合、たとえば
「2, 33, 420, 20662」 、 「2, 3, 3420, 20662」 、あるいは 「2334, 2020662」
のように区切ると **値がどんどん増えていく数列** にすることができます。
このように、 **値がどんどん増えていく数列** になるように区切る方法はたくさんあることが多いんですが、このなかで要素数が最も多いものを探します。
この例の場合、 「2, 33, 420, 20662」 や 「2, 3, 3420, 20662」 などが、要素数 4 で最大となります。
さて。
入力文字列から、出来る限り要素数の多い **値がどんどん増えていく数列** を作ると、その要素数がいくつになるのかを計算するプログラムを書いて下さい。

詳細な仕様

入力文字列は

- 先頭は `1` 〜 `9` のいずれかの文字。
- 先頭以外は `0` 〜 `9` のいずれかの文字。

という形式になっています。
つくるべき **値がどんどん増えていく数列** は、

- どの要素も、前の要素よりも大きい値である。
- どの要素も、先頭の文字が「0」であってはいけない。
- 入力文字列の順序どおりに過不足なく数字が現れること。順序を変えたり余らせたりしてはいけない。

というルールを守らなくてはいけません。

出力

出力は、最長の **値がどんどん増えていく数列** の要素数を 10進数で表記したものです。
例えば、入力が `23342020662` の場合、数列の項数を最大にするような区切り方は、例えば
「2, 33, 420, 20662」
なので、要素数の `4` を出力します。

解答形式

同梱されている data.txt の各行には空白区切りで

```
データ番号 入力データ 出力の期待値
```

が書かれています。出力の期待値が間違っているデータが数件(10件以下)含まれています。
出力の期待値が間違っているデータのデータ番号を、昇順に並べたものをフォームに書いて下さい。

補足

- 不正な入力に対処する必要はありません。
- 区切られた後の値が 64bit整数の最大値を超えることもあるかもしれません。オーバーフローにご注意下さい。
- 要素数が 1 でも **値がどんどん増えていく数列** に該当します。

サンプルデータ

以下のサンプルデータについては「出力の期待値」は全て正しい値になっています。

#	入力データ	出力の期待値	要素数が最大になる区切り方の例
0	23342020662	4	2, 3, 3420, 20662
1	23342121662	5	2, 3, 34, 212, 1662

2	2334200200662	5	2, 3, 34, 200, 200662
3	1	1	1
4	123	3	1, 2, 3
5	321	2	3, 21
6	11	1	11
7	111	2	1, 11
8	1111	2	1, 111
9	223043	3	22, 30, 43
10	33616382	4	33, 61, 63, 82
11	43727677	4	43, 72, 76, 77
12	123456789	9	1, 2, 3, 4, 5, 6, 7, 8, 9
13	987654321	3	9, 87, 654321
14	9876543210	4	9, 87, 654, 3210
15	1023456789	5	10, 23, 45, 67, 89
16	2000010000	1	2000010000
17	1000020000	2	10000, 20000
18	9080706050	2	9080, 706050
19	1222222221	4	1, 2, 222, 22221
20	1222222223	5	1, 2, 22, 222, 223
21	5463728191	5	54, 63, 72, 81, 91
22	223138576279	6	22, 31, 38, 57, 62, 79
23	9119291321331341	6	9, 119, 291, 321, 331, 341
24	123456789101112131415	15	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
25	10345678910111213141 5	7	10, 34, 56, 78, 910, 11121, 31415
26	100000000000000000000012345678912345678912345678 9	2	10000000000000000000012, 345678912345678912345678 9
27	162086	4	1, 6, 20, 86
28	234883	5	2, 3, 4, 8, 83
29	993485	3	9, 93, 485
30	4315388	4	4, 31, 53, 88
31	12277590	5	1, 2, 27, 75, 90
32	17368287	5	1, 7, 36, 82, 87
33	96937745	3	9, 693, 7745
34	400002770	1	400002770
35	2229316195	5	22, 29, 31, 61, 95
36	2539406179	6	2, 5, 39, 40, 61, 79
37	4198822419	4	4, 19, 88, 22419
38	74562578447	4	7, 45, 625, 78447
39	326589614864	5	3, 26, 58, 96, 14864
40	850805290204	4	8, 50, 805, 290204
41	2934238371377	6	2, 9, 34, 238, 371, 377
42	37706514606512	5	3, 7, 70, 651, 4606512
43	118197252504975	6	1, 18, 197, 252, 504, 975
44	220009653057100	3	2, 200096, 53057100
45	6656969229814531	6	6, 65, 69, 692, 2981, 4531
46	45124306072830842	6	4, 5, 12, 430, 607, 2830842
47	149247630076000000	7	1, 4, 9, 24, 76, 300, 76000000
48	391924415882839299	10	3, 9, 19, 24, 41, 58, 82, 83, 92, 99
49	424427430624645940	7	4, 24, 42, 74, 306, 2464, 5940
50	437368244825898219	6	4, 37, 368, 2448, 2589, 8219
51	78628139325152842 77	7	7, 8, 62, 81, 393, 2515, 284277
52	796002960670390347 53	6	7, 9, 600, 2960, 6703, 9034753
53	986706576404440034080	5	9, 86, 7065, 7640, 4440034080