2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Analysis

NumList:

"Given positive integers n and k, return the kth integer in the lexically ordered list of integers 1 through n inclusive."

At first glance it seems an easy task. Sort the integers 1 through n lexically and return the kth item in the list, right? So why did this apparently straight-forward problem stump three out of four finalists at the Collegiate Challenge? Why did dmwright code and test for 30 minutes before submitting the solution which clinched his victory?

The difficulty? 1 <= n <= 100000000.

This large limit on n makes any algorithm with running time proportional to n infeasible given the execution time constraints. Sorting requires a running time in proportion to n * log n. Also gigabytes of memory are needed to represent the integers 1 through 100000000 as strings, so any algorithm for NumList must avoid explicitly enumerating integers 1 through n.

One observation which leads to an efficient algorithm is the following. Count the integers between 1 and n that begin with the digit 1. (Exercise: compute this count efficiently.) If this count is less than k, then we know that the kth integer does not begin with the digit 1. Repeating this with digits 2 through 9, we can determine the first digit of the kth integer.
Once we know the first digit of the kth integer, we can solve for the second digit, the third digit, etc., until all digits of the kth integer are known.

The algorithm in the sample solution is best seen in the worked examples below, where a* represents the integers between 1 and n with prefix a, in lexical order. For example for n=1729, 17* represents the integers 17,170-179,1700-1729 in lexical order. We avoid explicit enumeration by implicitly enumerating the first k integers as a sequence of prefix lists.
Example: n=8293, k=1234, getKth returns 2108

Size	Kth	List (of first Size integers)
1111		1*
2222		1*,2* k exceeded
1112	2	1*, 2
1223		1*, 2, 20*
1334		1*, 2, 20*, 21* k exceeded
1224	21	1*, 2, 20*, 21
1235		1*, 2, 20*, 21, 210* k exceeded
1225	210	1*, 2, 20*, 21, 210
1226		1*, 2, 20*, 21, 210, 2100*
1227		1*, 2, 20*, 21, 210, 2100*, 2101*
1228		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*
1229		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*
1230		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*
1231		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*, 2105*
1232		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*, 2105*, 2106*
1233		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*, 2105*, 2106*, 2107*
1234		1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*, 2105*, 2106*, 2107*, 2108* k exceeded
1234	2108	1*, 2, 20*, 21, 210, 2100*, 2101*, 2102*, 2103*, 2104*, 2105*, 2106*, 2107*, 2108

Example: n=293857, k=219872, getKth returns 3341

Size	Kth	List (of first Size integers)
111111		1*
216080		1*, 2*
227191		1*, 2*, 3* k exceeded
216081	3	1*, 2*, 3
217192		1*, 2*, 3, 30*
218303		1*, 2*, 3, 30*, 31*
219414		1*, 2*, 3, 30*, 31*, 32*
220525		1*, 2*, 3, 30*, 31*, 32*, 33* k exceeded
219415	33	1*, 2*, 3, 30*, 31*, 32*, 33
219526		1*, 2*, 3, 30*, 31*, 32*, 33, 330*
219637		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*
219748		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*
219859		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*
219970		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*, 334* k exceeded
219860	334	1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*, 334
219871		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*, 334, 3340*
219882		1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*, 334, 3340*, 3341* k exceeded
219872	3341	1*, 2*, 3, 30*, 31*, 32*, 33, 330*, 331*, 332*, 333*, 334, 3340*, 3341

Additional Questions

1. What is the worst-case running time of the above algorithm as a function of n? Can you do better? (Either an algorithm showing you can or a proof that you can't.)

2. If you used the algorithm to sort the integers from 1 to n lexically, i.e. getKth(n,1), getKth(n,2) ... getKth(n,n), what would the running time of this sorting algorithm be? Can you do better?

Advice to Other Problem Writers

See ZandMaster's quote.

See next page for code sample

NumList:

```
public class NumList
{
   public int getKth(int  n, int  k)
   {
      int prefix  = 0;
      for (int  total  = 0; total<k;  )
      {
         prefix *= 10;
         for (int  count  =  count(n,prefix);  total+count<k;  prefix++,  count  = count(n,prefix))
            total  += count;
         if (total<k)  total++;  //  count  prefix
      }
      return  prefix;
   }

   int count(int  n, int  prefix)
   {
      int count  = 0;
      for (int  a = prefix,  powerOf10  = 1; 0<a && a<=n; a *= 10,  powerOf10  *=  10)
         count += ( a+powerOf10<=n  ? powerOf10  : n-a+1);
      return  count;
   }
}
```