## https://codeforces.com/problemset/problem/1765/N

```
1 对一串数字,
2 删掉若干个数字。
3 得到一个没有前缀0的最小数字:
```

## 20mins

- 这样删问题转化为什么? 什么问题?
- 某种贪心思路,追求前面的每一个数字为最大值。
  - 一旦出现最大值就立刻选择当前数字为首位。
    - 当前数字为首位置之后。就更新剩下的情况。
    - 如果当前位置为0不计入更新。
  - 如果当前数字没有出现更新了,说明当前的数字全部为0.
  - 如果这一段全部为0那么最优就是全部删除。
  - 。 复杂度爆炸, 搞个鬼。
- 考虑对当前的贪心进行优化。
- 使用st表。快速查询一个最小值的情况。但是也很麻烦。

## **SOLVE**

- 想办法记录下所有的数字的位置。
  - 。 在上述资源的前提下,上述的贪心思路。
    - 根据当前的剩余操作数,确定当前的可以任意选择的值的范围。(剩余操作数,上一个选择的数字的位置等。)
    - 从小到大进行选择数字。看一个数字是否落在当前对的范围之内。

```
1 #include <bits/stdc++.h>
2 using namespace std;
   typedef long long 11;
   const int maxn = 2e5 + 10;
5
6
  void solve()
7
8
      int n;
9
       // cin >> n;
10
      string s;
11
      cin >> s;
12
      cin >> n;
      //记录下每一个元素的位置。
13
       //然后就这一个资源上进行一些操作。
14
15
       //但是问题是终止步条件是什么?就是当k等于0的时候。
```

```
16
       vector<vector<int>> rec(10);
17
        for (int i = 0; i < s.size(); i++)
18
           rec[s[i] - '0'].push back(i); //记录下所有的情况。
19
        for (int i = 0; i < 10; i++)
           reverse(rec[i].begin(), rec[i].end());
20
        int len = s.size() - n;//这一个表示的是,将要构造的答案的长度。
21
        int now = 0; //当前已经在处理第几位了。
22
23
       string ans;
        for (int i = 0; i < len; i++)
24
25
           for (int j = (i == 0); j <= 9; j++) //然后这里选择最小的。
26
27
                while (rec[j].empty() == 0 && rec[j].back() < now).
28
                                   rec[j].pop back(); //边界点应该在哪里?
29
               if (!rec[j].empty() && rec[j].back() - now <= n)</pre>
30
31
                   ans += j + '0';
32
                   n = rec[j].back() - now;
33
                   now = rec[j].back() + 1;
34
                   break;
35
                }
36
37
       cout << ans << '\n';
38
39
   int main()
40
       ios::sync with stdio(false);
41
42
       cin.tie(nullptr), cout.tie(nullptr);
43
       int t;
44
       cin >> t;
45
       while (t--)
46
          solve();
47 }
```

## 做完题目之后的一些生长思考:

- 下面的这份代码为什么tle。
  - 猜测出现了死循环。
  - 否则常数上是不会相差那么大的。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const int maxn = 2e5 + 10;
5
6 void solve()
7 {
8  int n;
9  // cin >> n;
10 string s;
```

```
11
       cin >> s;
12
       cin >> n;
       //记录下每一个元素的位置。
13
       //然后就这一个资源上进行一些操作。
14
       //但是问题是终止步条件是什么?就是当k等于0的时候。
15
       vector<vector<int>> rec(10);
16
       for (int i = 0; i < s.size(); i++)
17
18
           rec[s[i] - '0'].push back(i); //记录下所有的情况。
       for (int i = 0; i < 10; i++)
19
20
           reverse(rec[i].begin(), rec[i].end());
21
       int len = s.size() - n;
       int now = 0; //当前已经在处理第几位了。
22
23
       string ans;
       for (int i = 0; i < len; i++)
24
25
           for (int j = (i == 0); j <= 9; j++) //然后这里选择最小的。
26
           {
27
               while (rec[j].empty() == 0 \&\& rec[j].back() < now)
28
                   rec[j].pop back();
                                                             //边界点应该
   在哪里?
29
               if (rec[j].empty() || rec[j].back() > now + n) //当前已经超
   界限;
30
                  continue;
31
               ans = ans + char(j + '0');
32
               n -= rec[j].back() - now;
33
               now = rec[j].back() + 1;
34
               break;
35
36
       cout << ans << '\n';
37
  int main()
38
39
40
       ios::sync with stdio(false);
       cin.tie(nullptr), cout.tie(nullptr);
41
       int t;
42
       cin >> t;
43
44
       while (t--)
45
          solve();
46 }
```