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

```
对一串数字，
删掉若干个数字。
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。

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 10;

void solve()
{
    int n;
    //  cin >> n;
    string s;
    cin >> s;
    cin >> n;
    //记录下每一个元素的位置。
    //然后就这一个资源上进行一些操作。
    //但是问题是终止步条件是什么？就是当k等于0的时候。
```

```
    vector<vector<int>> rec(10);
    for (int i = 0; i < s.size(); i++)
        rec[s[i] - '0'].push_back(i); //记录下所有的情况。
    for (int i = 0; i < 10; i++)
        reverse(rec[i].begin(), rec[i].end());
    int len = s.size() - n;//这一个表示的是，将要构造的答案的长度。
    int now = 0; //当前已经在处理第几位了。
    string ans;
    for (int i = 0; i < len; i++)
        for (int j = (i == 0); j <= 9; j++) //然后这里选择最小的。
        {
            while (rec[j].empty() == 0 && rec[j].back() < now).
 ..............                rec[j].pop_back(); //边界点应该在哪里？
            if (!rec[j].empty() && rec[j].back() - now <= n)
            {
                ans += j + '0';
                n -= rec[j].back() - now;
                now = rec[j].back() + 1;
                break;
            }
        }
    cout << ans << '\n';
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    int t;
    cin >> t;
    while (t--)
        solve();
}
```

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。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 10;

void solve()
{
    int n;
    //  cin >> n;
    string s;
```

```
    cin >> s;
    cin >> n;
    //记录下每一个元素的位置。
    //然后就这一个资源上进行一些操作。
    //但是问题是终止步条件是什么？就是当k等于0的时候。
    vector<vector<int>> rec(10);
    for (int i = 0; i < s.size(); i++)
        rec[s[i] - '0'].push_back(i); //记录下所有的情况。
    for (int i = 0; i < 10; i++)
        reverse(rec[i].begin(), rec[i].end());
    int len = s.size() - n;
    int now = 0; //当前已经在处理第几位了。
    string ans;
    for (int i = 0; i < len; i++)
        for (int j = (i == 0); j <= 9; j++) //然后这里选择最小的。
        {
            while (rec[j].empty() == 0 && rec[j].back() < now)
                rec[j].pop_back();                          //边界点应该在哪里？
            if (rec[j].empty() || rec[j].back() > now + n) //当前已经超界限;
                continue;
            ans = ans + char(j + '0');
            n -= rec[j].back() - now;
            now = rec[j].back() + 1;
            break;
        }
    cout << ans << '\n';
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    int t;
    cin >> t;
    while (t--)
        solve();
}
```