44、热点网站统计，考点 or 实现——字符串，数组，集合操作

题目描述

企业路由器的统计页面，有一个功能需要动态统计公司访问最多的网页URL top N。请设计一个算法，可以高效动态统计Top N的页面。

输入描述

每一行都是一个URL或一个数字，如果是URL，代表一段时间内的网页访问；
如果是一个数字N，代表本次需要输出的Top N个URL。

输入约束：

1、总访问网页数量小于5000个，单网页访问次数小于65535次；
2、网页URL仅由字母，数字和点分隔符组成，且长度小于等于127字节；
3、数字是正整数，小于等于10且小于当前总访问网页数；

输出描述

每行输入要对应一行输出，输出按访问次数排序的前N个URL，用逗号分隔。

输出要求：

1、每次输出要统计之前所有输入，不仅是本次输入；
2、如果有访问次数相等的URL，按URL的字符串 字典序 升序排列，输出排序靠前的URL；

用例

输入	news.qq.com news.sina.com.cn news.qq.com news.qq.com game.163.com game.163.com www.huawei.com www.cctv.com 3 www.huawei.com www.cctv.com www.huawei.com www.cctv.com www.huawei.com www.cctv.com www.huawei.com www.cctv.com www.huawei.com 3
输出	news.qq.com,game.163.com,news.sina.com.cn www.huawei.com,www.cctv.com,news.qq.com
说明	无

输入	news.qq.com www.cctv.com 1 www.huawei.com www.huawei.com 2 3
输出	news.qq.com www.huawei.com,news.qq.com www.huawei.com,news.qq.com,www.cctv.com
说明	无

题目解析

本题需要注意："每次输出要统计之前所有输入，不仅是本次输入"。

这句话，可以看第一个用例的第二行输出，其中news.qq.com的输出就取决于上面这句话。

另外，还有这句话："总访问网页数量小于5000个，单网页访问次数小于65535次；"

也就说，最多有不超过5000 * 65535条访问URL记录，这个规模，我们需要尽可能地优化代码时间复杂度到O(1)左右，特别是"每次输出要统计之前所有输入，不仅是本次输入"，我们最好创建缓存表，而不是重新统计。

其余地就感觉没什么需要注意的了，都是逻辑实现了。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
const cache = {};
rl.on("line", (line) => {
  lines.push(line);

  if (/^\d+$/.test(line)) {
    console.log(sortURL(lines, cache));
    lines.length = 0;
  }
});

function sortURL(lines, cache) {
  let n = parseInt(lines.pop());

  lines.forEach((url) => {
    cache[url] ? cache[url]++ : (cache[url] = 1);
  });

  let arr = [];
  for (let key in cache) {
```

```
    arr.push({
      count: cache[key],
      url: key,
    });
  }

  return arr
    .sort((a, b) => {
      let res = b.count - a.count;
      return res === 0 ? (a.url > b.url ? 1 : -1) : res;
    })
    .slice(0, n)
    .map((ele) => ele.url)
    .join(",");
}
```

Java算法源码

```
import java.util.*;
import java.util.regex.Pattern;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    // 改正则用于判断一个字符串是否为正整数
    Pattern reg = Pattern.compile("^\\d+$");
    // urls保存每次输入的url
    ArrayList<String> urls = new ArrayList<>();
    // cache记录每个url出现次数
    HashMap<String, Integer> cache = new HashMap<>();
    // ans保存输出
    ArrayList<String> ans = new ArrayList<>();

    while (sc.hasNextLine()) {
      String tmp = sc.nextLine();

      if ("".equals(tmp)) { // 由于本题没有说明输入结束条件，我这里将输入空行作为输入结束的条件，此时打印前面统计的结果
        for (String an : ans) System.out.println(an);
        sc.close();
        break;
      }
```

```
        }

        // 如果输入的是正整数
        if (reg.matcher(tmp).find()) {
          ans.add(sortURL(urls, Integer.parseInt(tmp), cache)); // 将统计结果记录到ans
          urls = new ArrayList<>(); // 清空urls，继续下次输入记录
        } else { // 输入的是url
          urls.add(tmp);
        }
      }
    }

    public static String sortURL(ArrayList<String> urls, int n, HashMap<String, Integer> cache) {
      for (String url : urls) {
        cache.put(url, cache.getOrDefault(url, 0) + 1);
      }

      StringJoiner sj = new StringJoiner(",");

      cache.entrySet().stream()
          .sorted(
              (a, b) ->
                  Objects.equals(a.getValue(), b.getValue())
                      ? a.getKey().compareTo(b.getKey())
                      : b.getValue() - a.getValue())
          .limit(n)
          .map(ele -> ele.getKey())
          .forEach(url -> sj.add(url));

      return sj.toString();
    }
}
```

Python算法源码

```python
import functools


def cmp(a, b):  # a,b是dict_item类型数据，a[0]是url，a[1]是url出现次数
    if a[1] != b[1]:
        return b[1] - a[1]
    elif a[0] != b[0]:
        return 1 if a[0] > b[0] else -1
    else:
        return 0


def sortURL(urls, cache, n):
    for url in urls:
        if cache.get(url) is None:
            cache[url] = 1
        else:
            cache[url] += 1

    urlCount = list(cache.items())
    urlCount.sort(key=functools.cmp_to_key(cmp))

    return ",".join(map(lambda x: x[0], urlCount[:n]))
```

```python


# 输入获取
urls = []  # 记录输入的url
cache = {}  # 记录每个url出现的次数
ans = []  # 记录题解

while True:
    tmp = input()

    if tmp == "":  # 由于本题没有说明输入结束的条件，因此，我将输入空行视为输入结束
        for an in ans:
            print(an)  # 打印结果
        break

    if tmp.isnumeric():  # 如果输入的是数字，则需要对前面输入的url进行排序统计
        ans.append(sortURL(urls, cache, int(tmp)))
        urls = []
    else:
        urls.append(tmp)
```