


华为OD机考算法题：告警抑制

御剑乐逍遥 于 2023-09-13 15:57:27 发布

 华为OD机考 专栏收录该内容

1 订阅 11 篇文章

题目部分

题目 华为OD机考算法题：告警抑制

难度 易

题目说明 告警抑制，是指高优先级告警抑制低优先级告警的规则。高优先级告警产生后，低优先级告警不再产生。请根据原始告警列表和告警抑制关系，给出实际产生的告警列表。
注：不会出现循环抑制的情况。
告警不会传递，比如A->B，B->C，这种情况下A不会直接抑制C，但被抑制的告警仍然可以抑制其他低优先级告警。
如果两个告警存在抑制关系，被抑制的低优先级告警无论是在高优先级告警的前面还是后面，都会被抑制。（笔者注）

输入描述 第一行为数字N，表示告警抑制关系个数， $0 \leq N \leq 120$ 。
接下来 N 行，每行是由空格分隔的两个告警ID，例如: id1 id2，表示id1抑制id2。告警ID的格式为:大写字母 + 0个或者1个数字。
最后一行为告警产生列表，列表长度[1,100]。

输出描述 真实产生的告警列表。

补充说明 告警 ID 之间以单个空格分隔。

示例

示例1

输入 2
A B
B C
A B C D E

输出 A D E

说明 A抑制了B，B抑制了C，最后实际的告警为A D E。

示例2

```
输入  4
      F G
      C B
      A G
      A0 A
      A B C D E

输出  A C D E

说明  无
```

解读与分析

题目解读：

此题要求从一个字符串中，删除被抑制的告警，只输出未被抑制的告警。输出后的告警顺序与源字符串保持一致。

注：在题目的告警列表中，如果两个告警存在抑制关系，无论被抑制的低优先级告警是在高优先级告警的前面还是后面，都会被抑制。题目中并未明说，但在示例 2 中是按照这个规则处理的。

此题在说明中存在歧义，需要通过示例 2 来澄清。个人认为，应该在题干中阐述清楚，而不是通过示例说明。

分析与思路：

先申明2个变量：

1. sourceAlarms，数组，输入的原始告警列表。
2. cutOffArray，二维数组，用以存放告警的抑制关系。数组的每个元素（设为 cutOffEle）是一个包含2个元素的抑制关系，cutOffEle[0] 是高优先级的告警，cutOffEle[1] 是被抑制的低优先级告警。
3. rmAlarmSet，集合，用以存储初始告警列表中需要删除的告警。

实现步骤如下：

1. 构建cutOffArray。逐行读取抑制关系，每行数据对应一个新的 cutOffEle。把每行数据拆分成两个告警ID，其中第一个告警 ID 放到 cutOffEle[0] 中，第二个告警 ID 放到 cutOffEle[1] 中，然后把 cutOffEle 作为元素放到数组 cutOffArray 中。
2. 构建 rmAlarmSet。遍历 cutOffArray，对于每个元素 cutOffEle，如果 cutOffEle[0] 在 sourceAlarms 中存在，则把 cutOffEle[1] 添加到 rmAlarmSet 中。
3. 遍历 sourceAlarms，如果告警 ID 不在 rmAlarmSet 中，则输出它；在 rmAlarmSet 中，则跳过。

内容来源：csdn.net

作者昵称：@Athena

原文链接：https://athena.blog.csdn.net/article/details/132856184

作者主页：https://athena.blog.csdn.net

说明：在第 2 步中，判断 cutOffEle[0] 在 sourceAlarms 中时，因为 sourceAlarms 是个数组，会让时间复杂度变大，可以先把 sourceAlarms 中所有的告警信息放到一个集合，如 sourceAlarmSet 中。

此算法的时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$ 。

代码实现

Java代码

```
1  import java.util.HashSet;
2  import java.util.Scanner;
3  import java.util.Set;
4
5  /**
6   * 告警抑制
7   * @since 2023.09.13
8   * @version 0.1
9   * @author Frank
10  *
11  */
12 public class CutOffAlarms {
13
14     public static void main(String[] args) {
15         Scanner sc = new Scanner(System.in);
16         while (sc.hasNext()) {
17             String input = sc.nextLine();
18             int count = Integer.parseInt( input );
19
20             String[][] cutOffArray = new String[count][2];
21             for( int i = 0; i < count; i ++ )
22             {
23                 input = sc.nextLine();
24                 String[] curCutOff = input.split( " " );
25                 cutOffArray[i] = curCutOff;
26             }
27
28             input = sc.nextLine();
29             String[] sourceAlarms = input.split( " " );
30             processCutOffAlarms( cutOffArray, sourceAlarms );
31         }
32     }
```

内容来源：csdn.net

作者昵称：御剑乐逍遥

原文链接：https://athena.blog.csdn.net/article/details/132856184

作者主页：https://athena.blog.csdn.net

```

33 | 34 | private static void processCutOffAlarms( String[][] cutOffArray, String sourceAlarms[] )
35 | {
36 |     Set<String> sourceAlarmSet = new HashSet<String>();
37 |     for( int i = 0; i < sourceAlarms.length; i ++ )
38 |     {
39 |         sourceAlarmSet.add( sourceAlarms[i] );
40 |     }
41 |
42 |     Set<String> rmAlarmSet = new HashSet<String>();
43 |     for( int i = 0; i < cutOffArray.length; i ++ )
44 |     {
45 |         String[] curAlarmEle = cutOffArray[i];
46 |         if( sourceAlarmSet.contains( curAlarmEle[0] ) ) {
47 |             rmAlarmSet.add( curAlarmEle[1] );
48 |         }
49 |     }
50 |
51 |     StringBuffer sb = new StringBuffer();
52 |     for( int i = 0; i < sourceAlarms.length; i ++ )
53 |     {
54 |         String eachAlarm = sourceAlarms[i];
55 |         if( rmAlarmSet.contains( eachAlarm ) )
56 |         {
57 |             continue;
58 |         }
59 |         sb.append( eachAlarm );
60 |         if( i != sourceAlarms.length - 1 )
61 |         {
62 |             sb.append( " " );
63 |         }
64 |     }
65 |     System.out.println( sb.toString() );
66 | }
67 | }

```

JavaScript代码

```

1 | const rl = require("readline").createInterface({ input: process.stdin });
2 | var iter = rl[Symbol.asyncIterator]();
3 | const readline = async () => (await iter.next()).value;

```

内容来源：csdn.net

作者昵称：御剑乐逍遥

原文链接：https://athena.blog.csdn.net/article/details/132856184

作者主页：https://athena.blog.csdn.net

```

4 void async function() { 5 | while (line = await readline()) {
6     var count = parseInt(line);
7
8     var cutOffArray = new Array();
9     for (var i = 0; i < count; i++) {
10         line = await readline();
11         var curCutOff = line.split(" ");
12         cutOffArray[i] = curCutOff;
13     }
14
15     line = await readline();
16     var sourceAlarms = line.split(" ");
17     processUnhappyKids(cutOffArray, sourceAlarms);
18 }
19 }();
20
21 function processUnhappyKids(cutOffArray, sourceAlarms) {
22     var sourceAlarmSet = new Set();
23     for (var i = 0; i < sourceAlarms.length; i++) {
24         sourceAlarmSet.add(sourceAlarms[i]);
25     }
26
27     var rmAlarmSet = new Set();
28     for (var i = 0; i < cutOffArray.length; i++) {
29         var curAlarmEle = cutOffArray[i];
30         if (sourceAlarmSet.has(curAlarmEle[0])) {
31             rmAlarmSet.add(curAlarmEle[1]);
32         }
33     }
34
35     var ret = "";
36     for (var i = 0; i < sourceAlarms.length; i++) {
37         var eachAlarm = sourceAlarms[i];
38         if (rmAlarmSet.has(eachAlarm)) {
39             continue;
40         }
41         ret += eachAlarm;
42         if (i != sourceAlarms.length - 1) {
43             ret += " ";
44         }
45     }
46

```

内容来源：csdn.net

作者昵称：御剑乐逍遥

原文链接：https://athena.blog.csdn.net/article/details/132856184

作者主页：https://athena.blog.csdn.net

(完)

 **文章知识点与官方知识档案匹配，可进一步学习相关知识**

算法技能树 > 首页 > 概览 51598 人正在系统学习中

内容来源：[csdn.net](https://athena.blog.csdn.net)

作者昵称：御剑乐逍遥

原文链接：<https://athena.blog.csdn.net/article/details/132856184>

作者主页：<https://athena.blog.csdn.net>