Stream Programming Summer 2019

Frequency Counter Algorithms — 01.04, 2019

dr inż. Wojciech Macyna *Scribe: Krzysztof Agieńczuk*

1 Overview

In the last lecture we talked about collections and streams.

In this lecture we covered: frequency counter algorithms.

2 Majority algorithm

Suppose we have a stream of 7 items. An item is majority item if it appears more than $\frac{length(stream)}{2}$ times.

2.1 Brute Force

One way to compute frequency of occurrences of an item is brute force algorithm. In this algorithm we have a map in which we store elements as key and amount of occurrences as values.
Sample pseudocode:

```
for element in stream
   if exists in map
      increment counter
   else
      add to map
```

2.2 Boyer-Moore algorithm[1]

In this algorithm we need only an item and a counter.
Firstly we take an element from a stream and set counter to 1. After that if the next element is the same we increment counter. If not we decrease counter till it is equal 0. If elements are different and counter is 0, we substitute item.

At the end item that is left is our candidate for most frequent element, so we traverse elements once again and see if it's frequency is bigger than half of the items in a stream.

2.3 Misra-Gries Algorithm[2]

In this algorithm we are looking for most frequent element such that amount of it's occurrences is bigger that $\frac{n}{k}$. Space complexity is O(k), as we need k-1 counters and k-1 items.

It works in the same fashion as Boyer-Moore algorithm with exception we have k-1 counters and k-1 items. Below you can find pseudocode for the algorithm.

```
//@param k
//setOfItems holds counters for all items being accounted for
for element in stream:
   if element is in setOfItems
      counter[element] := counter[element]+1
   else if |setOfItems| < k-1
      increase size of setOfItems
      counter[element] = 1
   else
      for item in setOfItems
         counter[item] := counter[item] - 1
         if counter[item] == 0
            remove counter from setOfItems
```

References

[1] Boyer R.S., Moore J.S. MJRTY—A Fast Majority Vote Algorithm. *Automated Reasoning. Automated Reasoning Series*, vol. 1, 1991, Springer, Dordrecht

[2] Misra J., Gries D. Finding repeated elements *Science of Computer Programming*, Volume 2, Issue 2, November 1982, Pages 143-152.