UNIVERSITÀ DI PISA

Sixth hands-on: Most frequent item in a stream

Algorithm Design (2021/2022)

Gabriele Pappalardo
Email: g.pappalardo4@studenti.unipi.it
Department of Computer Science

March 2022

1 Introduction

Suppose to have a stream of n items, so that one of them occurs $> n/2$ times in the stream. Also, the main memory is limited to keeping just $O(1)$ items and their counters (where the knowledge of the value of n is not actually required). Show how to find deterministically the most frequent item in this scenario.

Hint: the problem cannot be solved deterministically if the most frequent item occurs $\leq n/2$ times, so the fact that the frequency is $> n/2$ should be exploited.

2 Solution

A *streaming algorithm* is an algorithm that receives its input as *a stream* of data, and that proceeds by making only one pass through the data.

Assuming we have a stream of n items, it is possible to find the element with the highest frequency, knowing that it occurs $n/2$ times. The proposed solution makes use of *Boyer-Moore majority vote algorithm*, that is implemented in the Listing 1.

```
def boyer_moore_majority_vote(stream):

  m = None
  count = 0

  for i in range(len(stream)):
    if count == 0:
      m = stream[i]
      count = 1
    elif m == stream[i]:
      count += 1
    else:
      count -= 1

  return m
```

Listing 1: ”Boyer-Moore majority vote algorithm”

The algorithm uses just two local variables: one for the element and the other for the counter. Thus, the algorithm *space complexity* is $O(1)$. The algorithm will always find the element with the higher frequency while it is repeated at least $n/2$ times. When analyzing the stream the following conditions are evaluated:

1. if the counter is set to zero then we mark the element, denoted by the variable m, equal to i-th stream value, as the most frequent element;
2. if the i-th stream element is equal to the most frequent one, then we increase the counter by one;
3. otherwise, if none of the above conditions have been met, then the i-th element is different from the most frequent one. Therefore, we decrease the counter.

If m is the truly majority element, the counter will be always greater or equal than 1, because the increments will be more than the decrements.