Algorithm Design 21/22

Hands On 6 - Morris' Counters

Federico Ramacciotti

1 Problem

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

We can solve the problem using one counter and the correspondent item. The counter is initialized to the first element and 1. While we read the same element, the counter is increased, when we read a different element the counter is decreased. When the counter goes to 0, we change the element with the new one and we increment the counter to 1. It works because the element must occur more than $n/2$ times, so it occurs more times than all the other elements combined. The following is the code of the algorithm, written in Python.

```python
def occ(arr):
    maxocc = arr[1]
    count = 0
    for el in arr:
            if el == maxocc:
                    count+=1
            else:
                    count-=1
                    if count == 0:
                            maxocc = el
                            count = 1
    return maxocc
```