Deterministic data streaming

Consider a stream of n items, where items can appear more than once in the stream. The problem is to find the most frequently appearing item in the stream (where ties broken arbitrarily if more than one item satisfies the latter). Suppose that only k items can be stored, one item per memory cell, where the available storage is $k + O(1)$ memory cells. Show that the problem cannot be solved deterministically under the following rules: the algorithm can access only $O(log^c n)$ bits for each of the k items that it can store, and can read the next item of the stream; you, the adversary, have access to all the stream, and the content of the k items stored by the algorithm, and can decide what is the next item that the algorithm reads (please note that you cannot change the past, namely, the items already read by the algorithm) . Hint: it is an adversarial argument based on the k items chosen by the hypothetical deterministic streaming algorithm, and the fact that there can be a tie on $> k$ items till the last minute.

Show that the problem cannot be solved deterministically under the following rules: the algorithm can only use b bits, and read the next item of the stream, one item at a time. You, the adversary, have access to all the stream, and the content of the b bits stored by the algorithm: you cannot change those b bits and the past, namely, the items already read by the algorithm, but you can change the future, namely, the next item to be read. Since the algorithm must be correct for any input, you can use any amount of streams to be fed to the algorithm and as many distinct items as you want. [Hint: it is an adversarial argument based on the fact that, for many streams, there can be a tie on the items.]

SOLUTION

Any deterministic algorithm can be described by a deterministic finite state machine, in which every state given an input move deterministically to another state. Suppose to have a streaming of n element, k memory cell to store the frequencies, and θ are the bit necessary to store the algorithm state. Therefore we can have 2^θ possible states.

Let's suppose to input $2^\theta + 1$ times the same element to the algorithm. Then must be the case that the algorithm is going to be in the same state in two different moment (notice that the state are 2^θ). Let's call A_i and A_j the states of the algorithm after it received the element for the $i-th$ and $j-th$ times respectively (with loss of generality $i < j-1$). Notice that those two states are in the same place in two different moment.

Now let's insert $(i + 1)$-times a different element to the algorithm. Then let's proof that the algorithm reply in two different way:

Case 1: We are in A_i, therefore we have seen i times the first element, and $i + 1$ the second one. Therefore, in this case the second element win.

Case 2: We are in A_j, therefore we have seen j times the first element (with $i < j - 1$), and $i + 1$ the second one. Therefore, in this case the first element win.

Hence, we have a contradiction.