UNIVERSITÀ DI PISA

Seventh hands-on: Deterministic Data Stream

Algorithm Design (2021/2022)

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

April 2022

1 Introduction

Consider a stream of n items, where items can appear more than once. The problem is to find the most frequently appearing item in the stream (where ties are broken arbitrarily if more than one item satisfies the latter). For any fixed integer $k \geq 1$, suppose that only k items and their counters can be stored, one item per memory word; namely, only $O(k)$ memory words of space are allowed.

Show that the problem cannot be solved deterministically under the following rules: any algorithm can only use these $O(k)$ memory words, 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 these $O(k)$ memory words: you cannot change these words and the past items, namely, the items already read, but you can change the future, namely, the next item to be read. Since any algorithm must be correct for any input, you can use any amount of streams and as many distinct items as you want.

Hints:

1. This is "classical" adversarial argument based on the fact that any deterministic algorithm A using $O(k)$ memory words gives the wrong answer for a suitable stream chosen by the adversary.
2. The stream to choose as an adversary is taken from a candidate set sufficiently large: given $O(k)$ memory works, let $f(k)$ denote the maximum number of possible situations that algorithm A can discriminate. Create a set of C candidate streams, where $|C| > f(k)$: in this way there are two streams $S1$ and $S2$ that A cannot distinguish, by the pigeon principle.

2 Solution

To begin with, let us define a new function called moreFreq such that, given an alphabet of characters Σ we have:

$$\text{moreFreq} : \Sigma^* \to \Sigma$$
$$\text{moreFreq}(S) = \text{most frequent item } c \text{ in the stream } S$$

Where Σ^* it the *Kleene* closure of the alphabet used in automata theory.

As suggested by the hints, our main goal is to find two streams S_1 and S_2 such that $\text{mostFreq}(S_1) \neq \text{mostFreq}(S_2)$, but for the algorithm A we have $A(S_1) = A(S_2)$. To do so, we take an universe of elements, denoted with U, and its subsets $\Sigma_i \subseteq U$ such that they have cardinality $|\Sigma_i| = \frac{|U|}{2} + 1$ and $\forall i, j, i \neq j$ they satisfy the following properties:

1. $|\Sigma_i \cap \Sigma_j| \geq 1$: have at least one element in common;
2. $|\Sigma_i \setminus \Sigma_j| \geq 1$: at least one character differ in both sets.

How many subsets satisfy the requesting criteria? We use the binomial coefficient to discover that are:

$$\binom{|U|}{\frac{|U|}{2} + 1} \simeq 2^{|U|} = \eta \tag{1}$$

Note that if the cardinality of U is huge then the number of sets will be exponentially large!

From these subsets we can build the set of candidate streams C such that $|C| > f(k)$. Given an alphabet Σ_i, the Equation 2 shows how to build a possible stream. Be aware: the streams contain each character coming from its alphabet repeated only once (this fact will be exploited later).

$$S_i = s_1^{(i)} s_2^{(i)} s_3^{(i)} \dots s_{\frac{|U|}{2}+1}^{(i)}, \forall k, j \in [|\Sigma_i|], k \neq j . s_k \neq s_j \tag{2}$$

But, how big is the set of candidate streams? We have η alphabets which from each one we build a string as described above, and we put it in C. Therefore, the set will have a cardinality of:

$$|C| = \eta = 2^{|U|}$$

Since we are requiring that $|C| > f(k)$ we can derive that:

$$|C| > f(k) \iff 2^{|U|} > f(k) \iff |U| > \log f(k)$$

Therefore, $\exists S_i \in \Sigma_i^*, S_j \in \Sigma_j^* (S_i, S_j \in C)$ such that $S_i \neq S_j$ that those streams are indistinguishable for the algorithm A, due to the pigeonhole principle (since $|C| > f(k)$).

Now, if we pick an element $x \in \Sigma_i \setminus \Sigma_j$ and $y \in \Sigma_j \setminus \Sigma_i$, such that $x \neq y$ [1], and we concatenate them to the streams S_i, S_j obtaining two new streams $S_i xy$ and $S_j xy$. Giving these two streams to A we have $A(S_i xy) = A(S_j xy)$ because S_i and S_j are indistinguishable for A. But:

$$\text{mostFreq}(S_i xy) \neq \text{mostFreq}(S_j xy)$$

Thus, the algorithm A fails.

[1] Since we are requesting the properties (1) and (2) we can take such x and y.