Algorithm Design 21/22

Hands On 7 - Deterministic Data Streaming

Federico Ramacciotti

1 Problem

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 words, let $f(k)$ denote the maximum number of possible situations that algorithm A can discriminate. Create a set of C candidate streams, where $C > f(k)$ and $S2$ cannot distinguish, by the pigeon principle.

2 Solution

Given an algorithm A and a function $Freq$ that exactly finds the most frequent item in a string, the goal is to find two streams $\tilde{S}_1, \tilde{S}_2$ *s.t.* $Freq(\tilde{S}_1) = Freq(\tilde{S}_2) \wedge A(\tilde{S}_1) = A(\tilde{S}_2)$.

Let's take the universe U and its subsets $\Sigma_i \subseteq U$ *s.t.* $|\Sigma_i| = \left\lceil \frac{|U|}{2} \right\rceil$. With $|\Sigma_i| = \frac{|U|}{2} + 1$ the number of possible subsets is

$$\binom{|U|}{\frac{|U|}{2} + 1} \cong 2^{|U|} > f(k)$$

Select from this subsets two subsets Σ_i and Σ_j such that $|\Sigma_i \cap \Sigma_j| \leq 1 \wedge |\Sigma_i \setminus \Sigma_j| \geq 1$.

Define two streams S such that

$$\exists\ S_i \in \Sigma_i^*,\ S_j \in \Sigma_j^*\ s.t.\ S_i \neq S_j \wedge S_i \overset{A}{=} S_j$$

so that S_i and S_j are indistinguishable for the algorithm A.

Given

$$S_i = s_1^i s_2^i \dots s_{\frac{|U|}{2}}^i$$

$$S_j = s_1^j s_2^j \dots s_{\frac{|U|}{2}}^j$$

we choose two characters x, y such that $x \in \Sigma_i,\ x \notin \Sigma_j$ and $y \notin \Sigma_i,\ y \in \Sigma_j$. Thus, $A(S_i xy) = A(S_j xy)$ because $S_i \overset{A}{=} S_j$ and we can conclude that the algorithm fails, since $Freq(S_i xy) \neq Freq(S_j xy)$.

We can also see the algorithm as a deterministic finite state automaton: since the transitions are unique, when the automaton is in the same state and receives the same character, it transition

to another state. If A is in a state and we give it the same characters, it transitions to the same state, giving the same response for both streams. If we build the streams as we have shown above, A fails.

Example: consider $S_1 = abx$ and $S_2 = aby$. We have that $Freq(S_1) = A(S_1) = a$ and $Freq(S_2) = A(S_2) = a$. Now we modify the streams, creating $S_1' = abxxy$ and $S_2' = abyxy$, such that $Freq(S_1') = x$ and $Freq(S_2') = y$. For the algorithm A, S_1 and S_2 were indistinguishable, so if we add xy to both streams, it outputs the same response for both. But their most frequent elements differs, so A outputs x for both or y for both, picking either one of them wrong. A fails!