In the Name of God

Theory of Computer Science - HW1

Ali Fathi - 99204943 Instructor: Dr. Ebrahimi

Note: here in communication complexity, we consider the last sending symbol is the result of function, $f(x, y)$.

1 Streaming Algorithm for Distinct Elements (DE)

1.1 Proof of $\Omega(nl)$:

Recall the Myhill-Nerode Theorem:

Myhill-Nerode Theorem. *For a given language L, if there exists some $L, n\_distinguishing$ set $T_n \subseteq L_n = L \cap \Sigma^{\leq n}$, therefore every streaming algorithms for L are at least $\Omega(\lfloor \log(|T_n|) \rfloor)$ in binary space usage.*

First, we introduce a decision problem which leads us to bound the main DE problem. Define:

$$L \coloneqq \{w \in \Sigma^* \mid DE(w) \leq \frac{|w|}{2}\}$$

If we have any algorithm A with $O(s(nl))$ binary space usage for DE, our space usage for deciding whether $w \in L$ or not is $O(s(nl) + \log n)$; because we just have to compare the result of algorithm A (which is $DE(w)$) with $\frac{|w|}{2}$; and that is possible by using a counter to save n in $\log n$ bits, divide it by 2 and then compare these two numbers without any further space consumption. Besides, if $s(nl)$ is $\Omega(\log n)$, then $O(s(nl) + \log n) = O(s(nl))$. We will show every algorithm for L is $\Omega(nl)$ so it would be impossible to have any algorithm with $o(nl)$ space for DE, and it says that DE should use $\Omega(nl)$ space.
To do so, according to Myhill-Nerode theorem, it is enough to introduce some $L_n\_distinguishable\ set$ with $2^{\Omega(nl)}$ elements where L_n is:

$$L_n = \{w = w_1 w_2 \cdots w_n \mid DE(w) \leq \frac{n}{2}\}$$

The following set will work:

$$\mathcal{S} \coloneqq \{\{s_1, s_2, \cdots, s_{\lfloor \frac{n}{2} \rfloor}\} | s_i \in \Sigma, s_i \neq s_j\} \text{ (possible because } 2^l > n^2 \geq n)$$

$$x_{\boldsymbol{s}} \coloneqq order(\{s_1, s_2, \cdots, s_{\lfloor \frac{n}{2} \rfloor}\})$$

which $order()$ induces an unique order on $\boldsymbol{s} = \{s_1, s_2, \cdots, s_{\lfloor \frac{n}{2} \rfloor}\}$.

$$T_n \coloneqq \{x_{\boldsymbol{s}} | \boldsymbol{s} \in \mathcal{S}\}$$

The elements of T_n are $L_n\_distinguishable$:

$$\forall x_{\boldsymbol{s}}, x_{\boldsymbol{t}}, \in T_n \text{ such that } \boldsymbol{s} \neq \boldsymbol{t} :$$

$$\boldsymbol{s} \cup \boldsymbol{s} = \boldsymbol{s} \Rightarrow DE(\boldsymbol{ss}) = \lfloor \frac{n}{2} \rfloor \Rightarrow x_{\boldsymbol{ss}} \in L_n$$

$$\boldsymbol{t} \cup \boldsymbol{s} \supsetneq \boldsymbol{s} \Rightarrow DE(\boldsymbol{ts}) > \lfloor \frac{n}{2} \rfloor \Rightarrow x_{\boldsymbol{ts}} \notin L_n$$

$$(\text{and } |\boldsymbol{ss}|, |\boldsymbol{ts}| \leq n)$$

As $|\Sigma| = 2^l$, for the size of T_n we have:

$$|T_n| = \binom{2^l}{\lfloor \frac{n}{2} \rfloor} = \frac{(2^l)(2^l - 1) \cdots (2^l - \lfloor \frac{n}{2} \rfloor + 1)}{(\lfloor \frac{n}{2} \rfloor)(\lfloor \frac{n}{2} \rfloor - 1) \cdots (1)} > (\frac{2^l}{\lfloor \frac{n}{2} \rfloor})^{\lfloor \frac{n}{2} \rfloor} \geq (\frac{2 \times 2^l}{2^{\frac{l}{2}}})^{\lfloor \frac{n}{2} \rfloor} > 2^{\frac{nl}{4}}$$

Therefore, T_n is $2^{\Omega(nl)}$ and the assertion is proved. ■

1.2 Algorithm with $O(nl)$:

It is so simple! We save all n inputs, then we can do everything on them:

w_1	w_2	...	w_n

As each w_i needs l bits to be stored, totally $O(nl)$ bits are used for storing the whole stream.

---

2 Randomized Algorithm

We suppose that $h(x_i)$ has a uniform distribution on $[0, 1]$. Defining $Y_i = h(x_i)$, the CDF of Y_i is:

$$F_i(y) = \mathbb{P}(Y_i \leq y) = y$$

As assumed, $h(x_i)$s are independent, therefore Y_is are i.i.d. having the same CDF as above. Defining $Y = min_i\{Y_i\}$:

$$\begin{aligned} F(y) \coloneqq \mathbb{P}(Y \leq y) &= 1 - \mathbb{P}(Y = min\{Y_1, Y_2, \cdots, Y_K\} > y) \\ &= 1 - \mathbb{P}(Y_1 > y,\, Y_2 > y,\, \cdots,\, Y_K > y) \\ &\overset{iid}{=} 1 - \prod_{i=1}^{K} \mathbb{P}(Y_i > y) = 1 - \prod_{i=1}^{n} (1 - F_i(y)) \\ &= 1 - (1-y)^K \end{aligned}$$

Taking the derivative to get PDF:

$$f(y) = K(1-y)^{K-1} \qquad y \in [0,1]$$

After having PDF, calculating expected value is straightforward:

$$\begin{aligned} \mathbb{E}[Y] &= \int_0^1 y f(y)\,\mathrm{d}y = K \int_0^1 y(1-y)^{K-1}\,\mathrm{d}y \overset{t=1-y}{=} K \int_0^1 (1-t)t^{K-1}\,\mathrm{d}t \\ &= [t^K - \frac{K}{K+1} t^{K+1}]_0^1 = \frac{1}{K+1} \qquad \blacksquare \end{aligned}$$

---

3 Multi-Readings

3.1 2^l Readings

We only need a l-bits counter. In round t ($1 \leq t \leq 2^l$), we look for t'th element of Σ and if found, increase the counter by one (and go to next round). Final number presented by counter would be the DE.

l-bits counter:

1	0	...	1

3.2 p Readings

We break 2^l letters of Σ into $\frac{2^l}{p}$ parts (for simplicity, assume it would be an integer. It is not important as we just consider orders). In round t ($1 \leq t \leq p$), we look for elements $(t-1) + ip,\ 0 \leq i \leq \frac{2^l}{p} - 1$. Here is a schematic of memory:

	<— l bits —>			
0 to p-1	1	0	...	1
p to 2p-1	0	1	...	1
⋮	⋮			
$2^l - p$ to $2^l - 1$	1	1	...	0

Each row is responsible for counting p elements (row i counts letters $(i-1)p$ to $ip-1$). In round t, we look for t'th member of $[(i-1)p, ip-1]$ elements in each row i. When any of these t'th members were found ($\frac{2^l}{n}$ candidates each round), we add the counter of it's corresponding row by 1 (and ignore it in the rest of surveillance. This could be done by a "flag" bit in each row which is set to 0 in the beginning of each round and set to 1 when the corresponding element is found. Clearly it wouldn't affect the order of storage).
At the end, we add up the value of the rows, where the result would be the DE (we can use one of the rows to hold the result of summation without any further memory allocation). Therefor, the total used memory is size of the table, which is $\frac{2^l}{p} \times l = O(\frac{2^l}{p} l)$. ■

3.3 $2p$ Readings

To do so, we need two rows each of them with $\frac{n}{2p}$ cells (each cell is l-bits and could save a letter):

pivot row	a_1	a_2	$\cdots$	$a_{\frac{n}{2p}}$
temp row	b_1	b_2	$\cdots$	$b_{\frac{n}{2p}}$

What we do is something like a sort. In round t ($1 \leq t \leq 2p$), we try to find t'th minimum $\frac{n}{2p}$ elements of input. Pivot row, contains $\frac{n}{2p}$ minimum elements of last round $(t-1)$ and temp row is supposed to contain $\frac{n}{2p}$ minimum elements of current rount (t) by comparing a new letter of input with elements of pivot row (it must be greater than all of them) and also with elements of temp row (if it is smaller than any of them, we should swap them together). At the end of the round, if we have found $\frac{n}{2p}$ distinct minimum elements, we should add our counter by $\frac{n}{2p}$ (the counter needs only l-bits, which doesn't affect our order). A pseudo-code for these explanations is like this:

```
Result: DE = final value of counter
initialize: fill pivot row and temp row with -1;
counter ← 0 ;
for t = 1 to 2p do
    pivot row ← temp row ;
    temp row ← all -1 ;
    for i = 1 to n do
        if w[i] is greater than all elements of pivot row then
            if temp row contains any -1 then
                put w[i] instead of one of -1s ;
            else if w[i] is smaller than some element in temp row
             like b[j] then
                swap(w[i], b[j]) in temp row ;
        end
    end
    counter ← counter + #(positive elements of temp row) ;
end
```

Algorithm 1: Pseudo-code for DE

In this algorithm, all element appear in pivot row, and no element in counted twice (because elements in pivot row are strictly increasing). So the final value of counter would be exactly the DE. As at most n distinct elements could exist, number of rounds required for pivot row to see all distinct elements is $\frac{n}{size(\text{pivot row})}$ and as $size(\text{pivot row}) = \frac{n}{2p}$, it would be $2p$ rounds at most, which justifies the $2p$ in question. Definitely, Space consumed by two rows is $O(\frac{n}{p}l)$ (a bit for showing -1 in each cell and l-bits for counter are $O(\frac{n}{p})$ and $O(l)$ and don't change the order). ■

---

4 Monochromatic Rectangles

Each $(x, y) \subseteq \mathcal{X} \times \mathcal{Y}$ pair in a particular protocol P, produces a particular communication history. We call it $h_P(x, y)$. Also we call all histories made by protocol P, H_P:

$$H_P = \{h_1, h_2, \cdots, h_K\} = \{h_P(x, y) \,|\, (x, y) \subseteq \mathcal{X} \times \mathcal{Y}\}$$

(I used the same number K as we will see why.)
Some pairs have same history, i.e. $h_P(x, y) = h_P(x', y')$. We call $R_h \subseteq \mathcal{X} \times \mathcal{Y}$ as all pairs which produces the same history h under protocol P. So histories

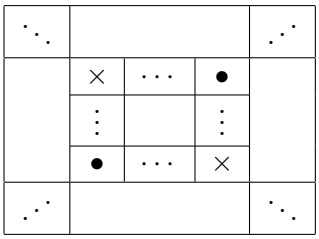
will partition all pairs as bellow:

$$R_{h_1} \uplus R_{h_2} \uplus \cdots \uplus R_{h_K} = \mathcal{X} \times \mathcal{Y}$$

We prove (in fact, recall) a lemma which leads us to make a connection between number of partitions and number of monochromatic subrectangles:

Lemma. *Every R_h represents a monochromatic subrectangle on M.*

Proof. Monochromacy is obvious, because a pair color is $f(x, y)$ and we have counted f in history. Therefore R_h members with same history have same color. So we should only prove R_h is a rectangle.
To do so, it is equivalent to prove for each two pairs such as (x, y) and (x', y') belonging to same partition like R_h, the two other corners of their subrectangle also belong to R_h (if we consider black points as (x, y) and (x', y') in matrix illustrated bellow, we should say crosses also belong to R_h).



According to "key lemma in communication complexity" proved in class, if $h_P(x, y) = h_p(x', y')$, then $h_P(x.y')$ and $h_P(x', y)$ are also the same with them. So, no need to say anythings more and that's exactly wat we wanted to prove. □

Joining this lemma with the fact that R_hs are partitions (which means they are disjoint and cover the whole matrix), we deduce that $R_{h_1}, \cdots .R_{h_K}$ define a subrectangles-covering of matrix M with size K. Therefore for minimal subrectangles-covering χ we have $\chi(f) \leq K$ (better to use K_P because K depends on protocol).
Remember K_P was the number of all communication histories produced by protocol P. So following inequality is held for K_P:

$$CC(f) = \Omega(\log_2(K_P))$$

Which is proved in class, and to give some sense of the proof, recalling $cost(P) = \max_{(x,y)}\{length(h_P(x, y))\}$, we had:

$$K_P \leq 2^{cost(P)}$$

Because all different histories in $cost(P)$ turns are at most $2^{cost(P)}$ histories which must be greater that K_P. Joining two inequalities, we get:

$$\forall P: \quad \chi(f) \leq K_P \leq 2^{cost(P)} \Rightarrow \chi(f) \leq \min_P\{2^{cost(P)}\} = 2^{CC(f)}$$

Which implies that $CC(f) \geq \log_2(\chi(f))$ and solution is done by knowing that $\chi(f)$ is just the K in question. ■
