# 21. Lower bounds for partial sums

*Lecturer: Meena Mahajan* *Scribe: Fahad Panolan*

We continue the discussion of proving lower bounds for data structure problems using communication complexity. Today we will prove a lower bound for the Partial Sums Problem.

## 21.1 The Partial sums problem

The *partial sums problem* is to maintain an array $A[1...n]$ in such a way that the following operations can be performed efficiently.

1. Update$(k, \Delta)$ : modify $A[k] \leftarrow A[k] + \Delta$
2. Sum$(k)$ : returns $\sum_{i=1}^{k} A[i]$

We assume that the value of each $A[i]$ at any time is bounded by poly$(n)$, and hence the required word size (capacity of a single cell) to store an array element is $O(\log n)$. We want to consider data structures that use poly$(n)$ cells of $O(\log n)$ bit size to store information about $A$, and study the number of cell reads per operation.

It is easy to obtain an $O(\log n)$ upper bound for the above operations, if we use a balanced binary tree where leaves contain array elements and each internal node stores the sum of elements appearing at leaves in the subtree rooted at that node. Today we will show that we really cannot do better; the amortized cost of an operation is $\Omega(\log n)$. The references for today's lecture are [PD04, PD06].

**Theorem 21.1.** *Any algorithm for the online partial sums problem that uses poly$(n)$ cells of size $O(\log n)$ bits to store information about the array has an amortized cell-probe complexity of $\Omega(\log n)$.*

## 21.2 The proof outline

Consider any algorithm for the partial sums problem using poly$(n)$ cells with $O(\log n)$ bits each. We show that it requires many cell probes on certain input sequences.

We focus on sequences of operations of a specific type. The sequences will consist of $n$ pairs of oeprations. Let $\pi$ be a permutation of $n$, and let $\Delta$ be an $n$-tuple of numbers. We consider the sequence

$$\begin{aligned} s_1 &= \langle \text{Update}(\pi(1), \Delta_1), \text{Sum}(\pi(1)) \rangle, \\ s_2 &= \langle \text{Update}(\pi(2), \Delta_2), \text{Sum}(\pi(2)) \rangle, \\ &\vdots \\ s_n &= \langle \text{Update}(\pi(n), \Delta_n), \text{Sum}(\pi(n)) \rangle. \end{aligned}$$

To prove the theorem, it is enough to show that if $\pi$ and $\Delta$ are chosen randomly, then $\mathbb{E}_{\pi,\Delta}[\#\text{ cell reads }] = \Omega(n \log n)$. In order to show this, we actually ignore many cell reads. While processing each pair $\langle \text{Update}(\pi(i), \Delta_i), \text{Sum}(\pi(i)) \rangle$, the algorithm reads many cells. Partition these cell reads based on when those cells were last modified. Further, if they were last modified while processing $s_j$ for some $j < i$, then charge this cell-read to the pair $(i, j)$; let there be $n_{ij}$ such reads. Ignore all reads of the form $(i, i)$. Thus the total number of cell-reads is at least as large as $\sum_{i=1}^{n} \sum_{j<i} n_{ij}$.

To better visualise the flow of information while the sequence of operations is being processed, conceptually construct a balanced binary tree on $n$ leaves. The $i$th leaf corresponds to an interval of time associated with the $i$th *update-sum* pair $s_i$, and *update-sum* pairs are arranged from left to right in the tree in the increasing order of time as they are executed. Now, for each $i$ and each $j < i$, there is a unique lowest common ancestor $\text{lca}(i, j)$ in this tree. Charge the $n_{ij}$ cell reads to this node. Overall, for any internal node $v$, we charge $\sum_{i>j} n_{ij}$ where the $i > j$ range over all pairs with lca $v$. Let $T$ be the set of internal noides in the tree. Then

$$\begin{aligned} \#\text{ cell reads } &\geq \sum_i \#\text{ of reads, while processing } s_i\text{, on cells which were last modified while processing } s_j \\ &= \sum_{j<i} n_{ij} \\ &= \sum_{v \in T} \#\text{ of cell reads performed while processing an instruction in the right subtree of } v\text{, where the cell read was last modified while processing an instruction in the left subtree} \end{aligned} \tag{21.2.1}$$

Let $\text{Read}_{\pi,\Delta}(v)$ denote the set of cell reads performed on input $\pi, \Delta$ while processing an instruction in the right subtree of $v$, where the cell read was last modified while processing an instruction in the left subtree. Then, as discussed above,

$$\#\text{ cell reads} \geq \sum_{v \in T} |\text{Read}_{\pi,\Delta}(v)|$$

$$\mathbb{E}_{\pi,\Delta}[\#\text{ cell reads}] \geq \sum_{v \in T} \mathbb{E}_{\pi,\Delta}[|\text{Read}_{\pi,\Delta}(v)|]$$

It therefore suffices to show a lower bound on $\mathbb{E}_{\pi,\Delta}[|\text{Read}_{\pi,\Delta}(v)|]$. We do this in two steps.

1. For a permutation $\pi$, and a node $v \in T$, we define an *interleaving factor* at $v$ due to $\pi$, denoted $\text{IL}_\pi(v)$. We show that

$$\mathbb{E}_{\pi,\Delta}[|\text{Read}_{\pi,\Delta}(v)|] \in \Omega(\mathbb{E}_\pi[\text{IL}_\pi(v)])$$

2. Let $L(v)$ denote the number of leaves in the subtree rooted at $v$. We show that

$$\mathbb{E}_\pi[\text{IL}_\pi(v)] = \Omega(L(v))$$

Putting these together, we obtain

$$\mathbb{E}_{\pi,\Delta}[\#\text{ cell reads}] \in \sum_{v \in T} \Omega(L(v)) \in \Omega\left(\sum_{v \in T} L(v)\right) \in \Omega(n \log n)$$

Since we had $2n$ instructions in our sequence, it follows that the amortized cost per instruction is $\Omega(\log n)$.

## 21.3 Proof of step 1

First, we define the interleaving factor. Let $P = \{\pi(i) | i \text{ is a leaf in the left subtree of } v\}$
$Q = \{\pi(i) | i \text{ is a leaf in the right subtree of } v\}$
The number of transitions from $P$ to $Q$ in the ordered listing of $P \cup Q$ is called the interleaving factor of $v$ between $P$ and $Q$, $\text{IL}_\pi(v)$. For instance, if $\pi$ restricted to the leaves under $v$ is `215911738`, then $P = \{1, 2, 5, 9\}$ and $Q = \{3, 7, 8, 11\}$. Sorting $P \cup Q$ gives `1 2 3 5 7 8 9 11`, and the interleaving factor is 3 (transitions at 2 3, 5 7, and 9 11).

For a fixed vertex $v$, the cell reads in the right subtree which were last modified in the left subtree can be thought of as involving communication between the left subtree and the right subtree. Consider an input $\pi, \Delta$, and Consider the situation in which we know

1. The permutation $\pi$.
2. All values of $\Delta$ outside the left subtree of $v$, say $\Delta'$.
3. The values and addresses of the cells which are read while processing the right subtree instructions.

Then all the *sum* queries in the right subtree of $v$ can be answered, even though all of $\Delta$ is not known, as follows: We execute the partial sums algorithm on $\pi, \Delta$, until we reach an update in the left subtree of $v$. Let the data structure constructed at this point be $S'$. Now we jump to the instructions in the right subtree of $v$. Whenever a cell is to be read, check whether this cell figures in the list at item 3. If so, use the value specified there, otherwise use the value of that cell in $S'$.

Actually we can say something stronger. Let $\Delta''$ denote the update values in the left subtree of $v$. The above algorithm not only reports partial sums in the right subtree of $v$, it also reports $\text{IL}_\pi(v)$ independent subset sums from $\Delta''$. For instance, as above, with $\pi =$ `2 1 5 9 11 7 3 8`, $P = \{1, 2, 5, 9\}$ and $Q = \{3, 7, 8, 11\}$, when we report Sum(11), we can also deduce $\Delta(1) + \Delta(2) + \Delta(5) + \Delta(9)$, since we know Sum(11) before and after these 4 updates. When we report Sum(7), we can deduce $\Delta(1) + \Delta(2) + \Delta(5)$. And when we report Sum(3), we can deduce $\Delta(1) + \Delta(2)$. When we report Sum(8), however, we cannot deduce any new information.

This can be formalised as follows; the proof is left as an exercise.

**Lemma 21.2.** *Given $\pi$, $\Delta'$ and the values and addresses of the cells which are read while processing the instructions in the right subtree of $v$, $IL_\pi(v)$ independent subset sums from $\Delta''$ (update values in the left subtree) can be obtained.*

Now we will construct a communication problem, which gives the required lower bound. In the communication problem, there is a permutation $\pi$ and an update sequence $\Delta$. The permutation $\pi$ and some update values, those in $\Delta'$, are public. Alice is given the remaining update values $\Delta''$ and she sends a single message $M$ to Bob. Bob reports $IL_\pi(v)$ independent subset sums of $\Delta''$.

From Lemma 21.2, it is enough that $M$ contains addresses and contents of those cells which are modified in the left subtree, and are next read in the right subtree. Assume that Alice follows this strategy. Then, for a fixed $\pi$ and $\Delta$,

$$\begin{aligned} |M| &\leq |\text{Read}_{\pi,\Delta}(v)|.(\text{Bits needed for the address of a cell} + \text{Size of a cell}) \\ &= |\text{Read}_{\pi,\Delta}(v)|.O(\log n) \end{aligned}$$

Now let $\pi$ and $\Delta$ be chosen uniformly at random. Then $\text{IL}_\pi(v)$ is itself a random variable, so is $M$. But $M$ suffices to recover $\text{IL}_\pi(v)$ independent subset sums in $\Delta''$. Specifying any one sum requires $\Theta(\log n)$ bits; so $M$ carries information about $\Theta(\text{IL}_\pi(v) \log n)$ bits. Thus

$$H(M) \in \mathbb{E}_\pi[\Omega(\text{IL}_\pi(v) \log n)] = \Omega(\mathbb{E}_\pi[\text{IL}_\pi(v)] \log n).$$

Putting the above equations together, we get, for some constant $c$,

$$\begin{aligned} c \cdot \log n \cdot \mathbb{E}_\pi[\text{IL}_\pi(v)] &\leq H(M) \\ &\leq \mathbb{E}_{\pi,\Delta}[|M|] \quad \text{(entropy cannot exceed average length)} \\ &\leq \mathbb{E}_{\pi.\Delta}[|\text{Read}_{\pi,\Delta}(v)|.O(\log n)]. \end{aligned}$$

$$\text{Hence } \mathbb{E}_{\pi.\Delta}[|\text{Read}_{\pi,\Delta}(v)|] \in \Omega(\mathbb{E}_\pi[\text{IL}_\pi(v)]).$$

## 21.4 Proving Step 2

This is left as an exercise!

**Exercise 21.3.** *Show that* $\mathbb{E}_\pi\big[IL_\pi(v)\big] = \Theta(\min\{|P|, |Q|\})$ *where* $P$ *and* $Q$ *are the sets of leaves in the left subtree of* $v$ *and right subtree of* $v$ *respectively. Hence conclude that* $\sum_v \mathbb{E}_\pi\big[IL_\pi(v)\big] = \Omega(n \log n)$.

## References


[PD04] Mihai Patrascu and Erik D. Demaine. *Tight bounds for the partial-sums problem.* In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 20–29. 2004. doi:10.1145/982792.982796.

[PD06] ———. *Logarithmic lower bounds in the cell-probe model.* SIAM J. Computing, 35(4):932–963, 2006. (Preliminary version in *36th STOC*, 2004 and *15th SODA*, 2004). arXiv:cs/0502041, doi:10.1137/S0097539705447256.