# CS60027 Parallel Algorithms, Autumn 2023–2024

**Class Test 2**

08–November–2023 | 06:30pm–07:30pm | Maximum marks: 20

---

**Roll no:** ____________________ **Name:** ____________________________________________

[*Write your answers in the question paper itself. Be brief and precise. Answer* <u>all</u> *questions.*]

Let $A = (a_1, a_2, \ldots, a_n)$ be an array (not necessarily sorted) of $n$ <u>distinct</u> integers (positive, negative, or zero). A pair $(a_i, a_j)$ satisfying the two conditions $i < j$ and $a_i < a_j$ is called *sorted within* (SW) $A$. Let $S = ((a_{i_1}, a_{j_1}), (a_{i_2}, a_{j_2}), \ldots, (a_{i_r}, a_{j_r}))$ be a set of some (not necessarily all) sorted-within pairs of $A$. Let us call such a set $S$ a *sorted-within set* (an SWS) of pairs from $A$.

Now, let $S$ be a set of pairs of elements from $A$. Each pair in $S$ is of the form $(a_i, a_j)$ with $i < j$. We do not impose the constraint $a_i < a_j$, so the pairs in $S$ may or may not be SW. Let $a_{k_1}, a_{k_2}, \ldots, a_{k_t}$ $(t \geqslant 0)$ with $k_1 < k_2 < \cdots < k_t$ be all the elements of $A$ <u>not</u> belonging to any pair in $S$ (as a first or a second component). These elements of $A$ are called *unpaired* (with respect to $S$). We call $S$ an *ascent destroyer* (AD) if we have $a_{k_1} > a_{k_2} > \cdots > a_{k_t}$.

For instance, take $A = (5, -2, 3, 8, 0, -4, 6, -1)$. The SWS $S_1 = \{(5,8), (-2,8), (-2,-1), (5,6)\}$ is an AD because the unpaired elements satisfy $3 > 0 > -4$, whereas the SWS $S_2 = \{(5,8), (-2,3), (-2,6), (5,6)\}$ is not an AD because the second of the inequalities $0 > -4 > -1$ does not hold. Finally, the set $S_3 = \{(5,8), (-2,8), (-2,-1), (8,6)\}$ is not an SWS, because it contains the non-SW pair $(8,6)$. But $S_3$ is an AD because the unpaired elements satisfy $3 > 0 > -4$ in the sequence as they appear in $A$. Finally, the set $S_4 = \{(5,8), (-2,3), (-2,6), (8,6)\}$ is neither an SWS nor an AD.

In general, the SWS $\{(a_i, a_j) \mid 1 \leqslant i < j \leqslant n \text{ and } a_i < a_j\}$ of all possible SW pairs is necessarily an AD. However, this set may contain $\Theta(n^2)$ pairs. This bound is tight, and is achieved, for example, if $A$ is sorted in the ascending order. However, a sorted array contains much smaller ADs like $\{(a_1, a_j) \mid 2 \leqslant j \leqslant n\}$ (of size $n-1$), $\{(a_i, a_n) \mid 1 \leqslant i \leqslant n-1\}$ (of size $n-1$), and $\{(a_1, a_2), (a_3, a_4), \ldots, (a_r, a_{r+1})\}$ (of size $\lfloor n/2 \rfloor$), where $r = n-1$ or $n-2$ according as whether $n$ is even or odd.

Your task is to find **any** set $S$ of pairs from a given $A = (a_1, a_2, \ldots, a_n)$, which is both an SWS and an AD. Adversarial arguments show that this problem cannot be solved using $o(n)$ operations (you do not have to prove this). You need to find one AD SWS $S$ using only $O(n)$ operations. The order of the pairs in your $S$ is not important. However, the pairs in $S$ must be distinct from one another (that is, every two pairs in $S$ must differ in at least one of the two components). You must output $S$ as a contiguous array of pairs. You may use an additional $O(n)$ space (but no more).

**1.** Propose a **sequential algorithm** to solve the problem in $O(n)$ time. Your proposal must contain a clear pseudocode, and a proof of correctness and running time. **(10)**

*Solution* We use a stack $K$ to solve the problem.

1. Initialize $K$ to an empty stack.
2. Initialize $S$ to the empty set.
3. For $j = 1, 2, \ldots, n$ (in that order), do:
   (a) If $K$ is empty, then push $a_j$ to $K$,
   (b) else, do:
   Extract the top $t$ from $K$.
   If $(a_j > t)$, then
   set $S := S \cup \{(t, a_j)\}$,
   else, do:
   Push $t$ back to $K$.
   Push $a_j$ to $K$.
4. Return $S$.

Assuming that push and pop in $K$ can be done in $\mathrm{O}(1)$ time and $S$ can be augmented by a new element in $\mathrm{O}(1)$ time, the running time of this algorithm is clearly $\mathrm{O}(n)$. For the proof of correctness, first note that $S$ consists of pairs of the form $(t, a_j)$ with $t < a_j$. Moreover, this $t$ must have been pushed as $a_i$ to $K$ for some $i < j$. So $S$ consists only of SW pairs. The elements not paired by $S$ are precisely those that remain in the stack $K$ when the for loop terminates. Whenever a new element $a_j$ is pushed on the top of $t$ in $K$, we have $a_j < t$, that is, the stack is always kept in the descending order from bottom to top. Finally, note that the elements of $K$ from bottom to top are always in the same order as they appear in $A$.

**2.** Propose an *optimal* **parallel algorithm** to solve the problem in $O(\log n)$ time. Your algorithm must meet the WT bounds on an **EREW PRAM**. The input array $A$ is supplied in the shared memory, and your algorithm should write $S$ as a contiguous array in the shared memory. Justify that your algorithm is correct and optimal, and runs in $O(\log n)$ time. **(10)**

*Solution* For each $j$, we plan to add at most one SW pair $(a_i, a_j)$. Here, $i$ may be the same for multiple values of $j$. In order to avoid concurrent read or write, we first compute the prefix minima of $A$.

1. Compute all the prefix minima of $A[\,]$ in $B[\,]$ using the BBT algorithm.
2. For $j = 1, 2, \ldots, n$, pardo:
   (a) If $(A[j] > B[j])$, then set $C[j] := 1$,
   (b) else set $C[j] := 0$.
3. Compute the prefix sums of $C[\,]$ in $D[\,]$ using the BBT algorithm.
4. For $j = 1, 2, \ldots, n$, pardo:
   (a) If $(C[j] = 1)$, then set $S[D[j]] := (B[j], A[j])$.

The prefix minima in Step 1 and the prefix sums in Step 3 can be computed in $O(\log n)$ time using $O(n)$ operations on an EREW PRAM. The remaining steps do not require any concurrent read or write, and can be done in $O(1)$ time using $O(n)$ operations. It follows that the overall running time is $O(\log n)$, and the total work done is $O(n)$ (which is optimal).

For proving the correctness, first note that $S$ consists only of pairs $(A[i], A[j])$ for which $A[i] = \min(A[1 \ldots j]) < A[j]$ (so $i < j$). Therefore $S$ is an SWS. In order to show that $S$ is an AD, take any two elements $a_k, a_l$ with $k < l$, unpaired in $S$. Since $A[l]$ is unpaired, we must have the condition $A[l] = \min(A[1 \ldots l])$. This, in particular, implies that $A[l] < A[k]$.