Homework #4

(Due: May 19)

Task 1. [200 Points] Sorting Almost All Items

Subset-Sort(A, n, d)

Input: A: array to be almost sorted at the current recursion level,
n: size of the array passed to Subset-Sort at recursion depth 0,
d: current recursion depth

Output: Almost all of A's items in sorted order

$m \leftarrow |A|$
if $d \geq \log\log\log n$ ***then***
  $A' \leftarrow$ items of A in sorted order obtained using an existing $\mathcal{O}(m \log m)$ work and $\mathcal{O}(\log m \log\log m)$ span deterministic sorting algorithm.
else
  $P \leftarrow \left\langle \sqrt{m} \log^3 n \text{ items selected uniformly at random from } A \right\rangle$.
  (The $\log^3 n$ factor in $\sqrt{m}\log^3 n$ is called the *oversampling factor*.)
  Sort P deterministically in $\mathcal{O}\left(\left(\sqrt{m}\log^3 n\right)^{1.5}\right)$ work and $\mathcal{O}(\log m + \log\log n)$ span.
  $P' \leftarrow \langle p_0, p_1, p_2, \ldots, p_{\sqrt{m}}, p_{\sqrt{m}+1} \rangle$, where $p_0 = -\infty$, $p_{\sqrt{m}+1} = +\infty$. and pivots $p_1, p_2, \ldots, p_{\sqrt{m}}$ are $\sqrt{m}$ uniformly spaced items selected from P in sorted order.
  For each p_i of P', allocate a bucket B_i of size $\Theta(\sqrt{m}\log n \log\log\log n / \log\log n)$ to place all items of A in the range (p_{i-1}, p_i), $1 \leq i \leq \sqrt{m}+1$.
  In parallel, find the bucket for each item e of A. Place e in two locations in the bucket chosen uniformly at random. Some items may be lost due to write collisions.
  Remove duplicate items of A from each bucket.
  Remove empty space between items in each bucket.
  Recursively sort every bucket $B_i \leftarrow$ Subset-Sort$(B_i, n, d+1)$.
  Remove empty space between the buckets to recover a fully sorted array A'.
  (A' will not necessarily contain all elements of A.)
endif
return A'

Figure 1: Subset-Sort$(A[1:n], n, 0)$ returns $n - o(n)$ items of A in sorted order.

Given an array $A[1:n]$ of n distinct items, n and $d = 0$ as inputs the Subset-Sort algorithm shown in Figure 1 attempts to sort the items in A by value. However, the algorithm is multithreaded[1]

[1] i.e., a parallel algorithm designed to run on a shared-memory multicore machine (i.e., on a multicore machine with a single RAM shared by all processing cores).

and allows *races*[2], and as a result some items will be lost due to write collisions. Hence, the sorted output generated by SUBSET-SORT may not contain all n items of A. Indeed, one can prove that SUBSET-SORT will return $n - o(n)$ items of A in sorted order in $\mathcal{O}(n \log n)$ work[3] and $\mathcal{O}(\log n)$ span[4] w.h.p. in n. This task asks you to write that proof.

(*a*) [**50 Points**] Prove that partitioning an array of size n into $\sqrt{n}+1$ blocks with oversampling factor $\log^3 n$ (as in Lines 5–7 of SUBSET-SORT) will produce no blocks with size falling outside of $(1 \pm \varepsilon)\sqrt{n}$ with probability at least $1 - n^{-\log n}$, where $\varepsilon = 1/\log^2 \log n$.

(*b*) [**10 Points**] Argue that part (*a*) continues to hold w.h.p. in n for arrays of size $m < n$ being partitioned into $\sqrt{m}+1$ blocks with fixed $\varepsilon = 1/\log^2 \log n$ and oversampling factor $\log^3 n$.

(*c*) [**80 Points**] Prove that attempting to place n elements in $n\alpha$ space (as in Lines 8–9 of SUBSET-SORT), where $\alpha = \Omega\left(\frac{\log n}{\log \log n}\right)$, will take $\Theta(n \log n / \log \log n)$ work and $\mathcal{O}(\log n)$ span, and result in $\Theta\left(\frac{n}{\alpha}\right)$ collisions, all w.h.p. in n. We make the following assumptions: (*i*) if multiple threads try to write to a memory location simultaneously only one (an arbitrary one) of them succeeds in writing and all others fail; and (*ii*) if k threads attempt to write to that location at the same time, the write costs each thread $\mathcal{O}(k)$ time even if the thread fails to write.

(*d*) [**20 Points**] Use part (*c*) to argue that SUBSET-SORT can recursively partition an n-element array to depth $\log \log \log n$ while keeping the size of every partition block at every depth $d \in [1, \log \log \log n]$ within a factor $(1 + 1/\log^2 \log n)^d$ of $\Theta\left(n^{2^{-d}}\right)$, w.h.p. in n.

(*e*) [**40 Points**] Prove that w.h.p. in n, the sorted version of A does not include a sequence of length $\Omega(\log n / \log \log n)$ without an element appearing in $B = \text{SUBSET-SORT}(A, n, 0)$.

(*f*) [**Optional: No Points**] Prove that the SUBSET-SORT algorithm takes $\mathcal{O}(n \log n)$ work w.h.p. in n, $\Theta(\log n)$ span w.h.p. in n, and $\Theta\left(\frac{n \log n \log \log \log n}{\log \log n}\right)$ space to sort $n - \Theta\left(\frac{n \log^2 \log n}{\log^2 n \log \log \log n}\right)$ elements of an n-element array w.h.p. in n.

[2] A *race condition* occurs if two or more threads try to access (i.e., read from or write to) the same memory location simultaneously and at least one of them attempts to write.

[3] The *work* performed by an algorithm when the input size is n, denoted by $T_1(n)$, is its running time on a serial machine (i.e., on a machine with a single processing core).

[4] The *span* of an algorithm when the input size is n, denoted by $T_\infty(n)$, is its running time on a parallel machine with an unbounded number of processing cores.