Homework #4

(Due: May 19)

GROUP NUMBER: 18

Group Members		
Name	SBU ID	% Contribution
Manish Reddy Vadala	114190006	33
Ranjith Reddy Bommidi	114241300	33
Venkata Ravi Teja Takkella	113219890	33

COLLABORATING GROUPS

Group Number	Specifics (e.g., specific group member? specific task/subtask?)
NA	NA

EXTERNAL RESOURCES USED

	Specifics (e.g., cite papers, webpages, etc. and mention why each was used)
1.	https://arxiv.org/abs/2008.13292 The work done in this paper is similar to the questions asked in the assignment.
2.	
3.	
4.	
5.	

1 (a)

Let us show the given input array in a sorted sequence format like $(a_1, ..., a_n)$, , from which $\sqrt{n}\log^3 n$ elements are sampled and out of them q = $\sqrt{n}$ pivots are chosen (with repetition) along with two pivots added in the first and last, with oversampling factor $s = \log^3 n$. The assumption behind the with repetition part is to decrease the complexity of the problem. Without repetition the probability of each element would be different and would increase the complexity of solution.

Firstly we can prove that, If one of the resulting blocks contains at least k elements, then there must be a sub-sequence $(a_j, ..., a_{j+k-1})$ from which less than $\log^3 n$ elements were sampled. We can say that because, since the initial whole input array is assumed to be a sorted sequence. For a case, where the random sampling are taken from the first $\sqrt{n}\log^3 n$ elements and after keeping the whole set of elements in the buckets, we would be getting $\log^3 n$ elements in all the buckets except the last one. Then we can clearly see that there exists a lot of blocks with at-least $\log^3 n$ elements and using this as k, any sub-sequence in the original array would give at maximum of $\log^3 n$ elements. And this value is much smaller if the initial assumption of continuous random sampling is changed to more sparse sampling.

In the similar way, we can also show that if the resulting blocks contains less than k elements, there must be a sub-sequence from which at least $\log^3 n$ samples were sampled.

$\{x_1, ..., x_{(q+1)s}\}$ be the elements sampled from A, and define the Indicator variables $X_i = 1$ if $x_i \in (a_j, ..., a_{j+k-1})$, $X_i = 0$ otherwise, and $X = \sum_{i=1}^{(q+1)s} X_i$ for some arbitrary fixed j. Here we know that $E(X_i)$ is basically expectation of a specific element being present in the sub-sequence i.e. $\frac{k}{n}$.
Now E(X) = (q + 1)sE(X_i) = $\frac{sk}{\sqrt{n}}$. Let b_i be the sizes of the blocks of the partition of A with pivots $x_s, x_{2s}, ..., x_{qs}$: we then have $P(max_i b_i \geq k) \leq P(\exists_j\ X < s)$ (resp. $P(min_i b_i < k) \leq P(\exists_j\ X \geq s)$).

We handle the upper tail by setting $k = (1+\epsilon)\sqrt{n}$ gives us $E(X) = (1+\epsilon)s$, from which we can upper bound the probability of an unexpectedly large partition block(as seen in our initial assumption to prove our statement) by using a Chernoff bound.:

$$
\begin{aligned}
P\Big(\max_i b_i \geq (1+\epsilon)\sqrt{n}\Big) &\leq P(\exists_j\ X < s) \\
&\leq nP\Big(X < (1-\epsilon/2)E(X)\Big) \qquad\qquad \Big(1-\epsilon/2 > \frac{1}{1+\epsilon}\Big) \\
&\leq nexp(-\epsilon^2 s/8) \\
&= n\ exp\Big(\frac{-\log^3 n}{8\log^4 \log n}\Big) \\
&< (1/2)n^{-\log n}\ (for\ n \geq 4)
\end{aligned}
$$

Similar to the treatment of upper tail, for the lower tail we set $k = (1-\epsilon)\sqrt{n}$, which gives $E(X) = (1-\epsilon)s$. Applying the Chernoff bound for lower tails then shows that

$$
\begin{aligned}
P\Big(\min_i B_i < (1-\epsilon)\sqrt{n}\Big) &\leq P(\exists_j\ X \geq s) \\
&\leq nP(X > (1+\epsilon)E(Y)) \qquad\qquad \Big(1+\epsilon < \frac{1}{1-\epsilon}\Big) \\
&\leq n\ exp(-\epsilon^2(1-\epsilon)s/4) \\
&\leq n\ exp\Big(\frac{-\log^3 n}{8log^4 \log n}\Big) \\
&\leq (1/2)n^{-\log n} \qquad\qquad (for\ n \geq 4)
\end{aligned}
$$

Combining the upper and lower tails immediately gives a bound on the probability of a sufficiently regular partition, $P\Big(\forall i(1-\epsilon)\sqrt{n} \leq b_i \leq (1+\epsilon)\sqrt{n}) > 1 - n^{-\log n}\ for\ n \geq 4$.

1 (b)

If the number of elements that need to be partitioned while also still using the same oversampling factor, will only give more probability to get a more uniform partition. So if it has worked w.h.p n in the above question, it should still hold if the number of elements in the array $m < n$; which reached to the algorithm at a certain depth too.

Hence Proved.

1 (c)

The probability of a given element to be part of a collision is less than $\frac{1}{m}$, and so the fact that $\Theta(n/m)$ collisions will occur w.h.p can be directly shown from a pair of Chernoff bounds, $P(X > 2E(X)) < e^{-E(X)/3}$ and $P(X < E(X)/2) < e^{-E(X)/8}$, where we take X to be the number of collisions.

We are given in the question that the write costs $O(k)$ if k threads try to write to the same location. So considering that $\Theta(n/m)$ collisions which were misplaced due to a collision: they are randomly distributed throughout the $\Theta(n)$ locations that were selected by at least one element, so with probability greater than 1-$exp^{(-\Theta(n/m^2))}$ there are $\Theta(n/m^2)$ of them that collided with one another.
By induction, there are $\Theta(n/m^d)$ locations where d elements collided, w.h.p. in n for all d such that $n/m^d = \Omega(logn)$. The remaining $O(\log(n))$ locations where $\Omega(log_m n)$ elements collided contribute, at most,$O(log^2 n)$ work and $O(logn)$ span.

The span is therefore dominated by the time taken to spawn n processes,$T_\infty(n) = \Theta(logn)$,and work is given by $T_1(n) = \Theta(n+n\sum_{d=1}^{\log_n m} \frac{d^2}{m^d})$, where we have used the identity$\sum_{j=1}^{\infty} j^2 x^j = \frac{x(1+x)}{(1x)^3}\, x$ for x small. So if we write our equation above for the work again, $T_1(n) = \Theta(n+nm) = \Theta(nm) = \Theta(nlogn/loglogn)$

1 (d)

Let n be the size of the array passed to Subset-Sort in the initial call, and n_c be the size of the array passed at some point in the recursion tree. We terminate recursion at depth $\log\log\log n$, at which point the base condition sorting algorithm is applied, taking $O(n_c \log n_c)$ work. All higher levels of recursion sample $\sqrt{n_c}$ pivots with oversampling factor $\log^3 n$,which are sorted using our rest of the algorithm, after which Subset-Sort is called on the $\sqrt{n_c}$ partition blocks, each of which will be no larger than $(1 + 1/\log^2 \log n)\sqrt{n_c}$. For every call to Subset-Sort there is $O\Big(n_c \frac{\log n \log\log\log\log n}{\log\log n}\Big)$ work done on the binary searches by which elements find which partition block to be placed in. Thus the work is $T_1(n_c, n) = O(n_c \log n_c)$ when depth is greater than $\log\log\log n$, and $T_1(n_c, n) = O\Big((\sqrt{n_c}\log^3 n)^{\frac{3}{2}} + n_c \frac{\log n \log\log\log\log n}{\log\log n} + n_c \log n_c\Big) + \sqrt{n_c} T_1((1 + \frac{1}{\log^2 \log n})\sqrt{n_c}, n)$ otherwise. The Subset-Sort is called $O(n \log\log\log n)$ times, and the likelihood of a suitably uniform partition during any particular call of Subset-Sort is bounded below by $1 - n^{-\log n}$, so all partitions will be suitably uniform with high probability in n.

And so can upper bound w.h.p. the size of arrays n_c passed to depth d with $(1 + \frac{1}{\log^2 \log n})^d n^{2^{-d}}$. Therefore in order to be able to apply to the array partitioning which occurs at depth d, we can use SUBSET-SORT and by keeping the size of every partition block within a factor of $(1 + \frac{1}{\log^2 \log n})^d$ of $\theta(n^{2^{-d}})$

1 (e)

We know with high probability that the bins at all stages of Subset-Sort's divide and conquer process are filled no more than within a $(1 + \frac{1}{log^2 logn})^d$ factor above or below expectation (from part d). We can get more simpler bound by $(1 + \frac{1}{log^2 logn})^d n^{2^{-d}} \leq (1 + \frac{1}{\log^2 \log n})^{\log_{3/2} \log n} n^{2^{-d}} \leq exp(1/\log\log n) n^{2^{-d}} \sim (1 + \frac{1}{\log\log n}) n^{2^{-d}}$

And this gives us an upper bound on the likelihood that any given element $a_j \in A$ fails to be included in B.

$$P(a_j \notin B) \leq 1 - \Big(1 - \frac{1 + 1/\log\log n}{\frac{\log^2 n \log^2 \log\log n}{\log^2 \log n}}\Big)^{\log\log\log n} \sim \frac{\log^2 \log n}{\log^2 n \log\log\log n} \equiv \delta$$

This result also follows from our high probability bound on the number of collisions which occur during Subset-Sort.

Now we can repeat the argument which we did in part a, interpreting B as having been uniformly randomly sampled from A, and thereby bounding the likelihood that a subsequence of length exists in A from which no elements are included in B.

We define the indicator variables Y_i to be 1 if $b_i \in a_j, ..., a_{j+l1}$ and 0 otherwise, and $Y \equiv \sum_i Y_i$, from which we have $P(Y = 0) \sim \prod_i P(a_{j+i} \notin B) \leq \delta^l$. The probability of a subsequence of length being contained in A but completely absent from B is then bounded with $P(\exists j Y = 0) \leq nP(Y = 0) \leq n\delta^l$, so we find that there is a polynomially small chance of there being a subsequence of length $l = \frac{\log n}{\log \frac{1}{\delta}} = \Theta\big(\frac{\log n}{\log\log n + \log\log\log\log n \log\log\log n}\big)$ which is entirely absent from B.

We can approximate this length to $\Omega\big(\frac{\log n}{\log\log n}\big)$