Homework 4

(Due: Dec 14)

Group Number: 54

Group Members		
Name	SBU ID	% Contribution
Sharad Sridhar	111492675	33
Nihal Harish	111498545	33
Raunak Shah	111511721	33

Collaborating Groups

Group Number	Specifics (e.g., specific group member? specific task/subtask?)
17	Help was taken for TASK 1C
27	Question 2 was done in collaboration
35, 17	Question 3 was done in collaboration

External Resources Used

	Specifics (e.g., cite papers, webpages, etc. and mention why each was used)
1.	Lecture Slides used for study
2.	
3.	
4.	
5.	

Task 1

Solution for part a

We can parallelize the given algorithm in the following way:

Select($A[q:r], k$)

1. $n \leftarrow r - q + 1$

2. if $n \leq 140$ then

3. sort$A[q:r]$ and return $A[q+k-1]$

4. else

5. divide A$[q:r]$ into block B_i's each containing 5 consecutive elements
   (last block may contain fewer than 5 elements)

6. PARALLEL FOR $i \leftarrow 1$ *to* $\lceil n/5 \rceil$ do

7. $M[i] \leftarrow$ median of B_i using sorting

8. $x \leftarrow$ $Select(M[1:\lceil n/5 \rceil], \lfloor(\lceil n/5 \rfloor + 1)/2 \rceil)$ (median of medians)

9. $t \leftarrow$ Parallel-Partition($A[q:r], x$) (partition around x which ends up at $A[t]$)

10. if $k = t - q + 1$, then return $A[t]$

11. else if $k < t - q + 1$, then return Select($A[q:t-1, k]$)

12. else return Select($A[t+1:r], k - t + q - 1$)

We can analyze the Work and Span of this algorithm by looking at the bounds during certain steps in the program.

Work can be represented as $T(n)$.

Using the already derived (in class) recurrence for a serialized deterministic select algorithm, we have (using the Akra Bazzi method) the result as $\theta(nlogn)$

The span can be calculated as follows:

Now, the parallel for loop in line 6 will take $\theta(logn)$ time, and the line 9 parallel-partition will take $\theta(log^2 n)$, which leads us to the fact that at each level of the recursion tree.

The recurrence relation leads to an observation that the problem of size n is divided into two sub-problems of size $(n - \alpha_x)$ at each level. The depth of the recursion tree is $\theta(logn$.

This leads to a parallel runtime of $\theta(log^3 n)$

Now, the parallelism is $\frac{work}{span} = \frac{\theta(nlogn)}{\theta(log^3 n)} = \theta(\frac{n}{log^2 n})$

Solution for part b

The parallel algorithm for returning the k-th smallest number is as follows:

Par-Randomized-K-Select($A[q:r], k$)

1. $n \leftarrow r - q + 1$

2.if $n \leq 30$, then

3. sort$A[q:r]$ and return $A[q+k-1]$

4. else

5. select a random x from $A[q:r]$

6. $t \leftarrow$ Par-Partition($A[q:r], x$)

7. if $k = t - q + 1$, then return $A[t]$

8. else if $k < t - q + 1$, then return Par-Randomized-K-Select($A[q:t-1, k]$)

9. else return Par-Randomized-K-Select($A[t+1:r], k - t + q - 1$)

Now, similar to the previous question ,we see that for WORK, we have line 6 that does $\theta(n)work$. The depth of the recursion tree for parallel partition is $\theta(logn)$. Thus, same as before we have a work $= \theta(nlogn)$.

Now, for Parallelism,

Same as the previous question, we have line 6 doing $\theta(log^2 n)$ work. The depth of the recursion tree is $\theta(logn)$. This leads us to a SPAN of $\theta(log^3 n)$

Now, the parallelism is $\frac{work}{span} = \frac{\theta(nlogn)}{\theta(log^3 n)} = \theta(\frac{n}{log^2 n})$

Solution for part c

For this problem, we consider (2k+1) numbers at a time and compare them to the original n numbers. We can do this in parallel. Each processor compares the 2k+1 numbers with the given set of n distinct numbers. Based on the comparison, the appropriate cell is updated with 0 or 1.

Par-k-Smallest($A[q:r], k$)

1. $n \leftarrow r - q + 1$

2. if $n \leq 140$ then

3. sort$A[q:r]$ and return $A[q+k-1]$

4. PARALLEL FOR $i \leftarrow 1$ to n do

5. PARALLEL FOR $j \leftarrow 1$ to $2k+1$ do

6. IF $A[q+i-1] > A[q+j-1]$

7. $COMP[i][j] = 1$

8. ELSE

9. $COMP[i][j] = 0$

10. PARALLEL FOR $i \leftarrow 1$ to n do

11. $C[i] = PREFIX - SUM(COMP[i][],' +')$

12. PARALLEL FOR $i \leftarrow n$ do

13. IF $(C[i] < k)$

14. $final - result[C[i]] = A[i]$

15. return final-result

The extra space is taken as $\theta(nk)$ as is shown by the COMP array.

In serial case, the maximum time is taken by the two nested for loops, one which runs for k times and one which runs for n times. The complexity here is bounded by this which is $\theta(nk)$.

Now, when we analyze the span:

The two parallel for loops take $\theta(logn)time.$

Now in line 10, we have a parallel for loop which will take another $\theta(logn)time.$ From the class discussions we have the complexity of PREFIX SUM as $\theta(logn)$. Now this then gives a bounding complexity of $\theta(log^2n)$, which is the required result.

Parallelism can be computed as : $\frac{\theta(nk)}{\theta(log^2n)}$