CMPE 300 - Analysis of Algorithms
Fall 2015
Assignment 2 Solutions

Question 1

Consider a list of n integers $X = \{x_1, x_2, \ldots, x_n\}$ where $x_1 = x_2 = \ldots x_i \neq x_{i+1} = x_{i+2} = \ldots x_n$.

a) Write an EREW PRAM algorithm in pseudocode for finding the index i where the number of processeors is p. Perform an analysis and express the time complexity.

Solution:

We should write an EREW algorithm to find index i. First divide the list into p blocks of size $\frac{n}{p}$. Each processor performs sequential search on this $\frac{n}{p}$ numbers to find the index. Since the control results in a positive answer only for one of the indices i, We don't have any concurrent writes. Note that the indices on the are not checked. $p-1$ processors can perform this check in $O(1)$ time.

There are also different algorithms with similar ideas but you should be careful about different processors accessing the same index since this is an executive read algorithm. That is why we don't check overlapping indices in parallel in the first place.

The complexity of the algorithm is $O(\frac{n}{p})$. If the sequential search is implemented as binary search then the complexity is reduced to $O(\log(\frac{n}{p}))$ which is the optimal solution for this problem.

b) Suppose that any EREW PRAM algorithm requires $(\log n - \log p)$ time for solving the problem. Prove that CREW PRAM is more powerful than EREW PRAM.

Solution:

In order to prove that CREW PRAM is more powerful than EREW PRAM for this problem, we need to find an algorithm which has time complexity better than $(\log n - \log p)$. The idea is as follows.

Divide the list into group of p elements where each group consists of $\frac{n}{p}$ elements. Processor i checks if the first and the last elements are different in the i'th group. This step requires $O(1)$ time. The list is again divided into p and the same procedure is applied until p elements are left. Note that the algorithm requires $\log_p n$ steps. After that, each index can be compared with the adjacent index parallely (conccurent read) by p processors in $O(1)$ time. The algorithm has $O\frac{\log_n}{\log_p}$ time complexity in total.

Question 2

Consider a permutation of the list of the integers in Question 1 and call the new list $Y = \{y_1, y_2, \ldots, y_n\}$. Write a Monte Carlo algorithm in pseudocode to find the integer which appears most in the list. Perform complexity analysis. Does your algorithm always give the correct answer? You will get more points or no points at all depending on the complexity of your algorithm.

Solution:

Notice that the list contains only two distinct elements, say i and j. Let the number of occurences of i and j be n_i and n_j respectively. Note that $n_i + n_j = n$ and $n_i > \frac{n}{2}$ if i is the majority element and $n_j > \frac{n}{2}$ if j is the majority element. Therefore, if you select a random integer r, the probability that $L[r]$ is the majority element is more than $\frac{1}{2}$. So, the algorithm is to randomly select an integer r and return r which has $O(1)$ complexity. The algorithm does not always return the correct answer. The correcness of the algorithm is $\frac{x}{n}$ where x is the number of occurences of the majority element. Correctness of the algorithm can be increased by repeating the same procedure for constant number of times.