INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (Mid Semester)									**SEMESTER (Autumn)**			
Roll Number									**Section**		**Name**	
Subject Number	C	S	6	0	0	2	6	**Subject Name**		*Parallel and Distributed Algorithms*		
Department / Center of the Student									**Additional sheets**			

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

CS60026 Parallel and Distributed Algorithms, Autumn 2017–2018

Mid-Semester Test

19–September–2017 CSE-107/108/119, 09:00–11:00am Maximum marks: 50

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions. If you use any algorithm/result/formula covered in the class, just mention it, do not elaborate.*]

1. We want to compute the dot product of two n-dimensional vectors $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, defined as $\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$.

(a) Let $m = \sqrt{n}$ be an integer. You are given an $m \times m$ mesh with the processors numbered $P_1, P_2, \ldots, P_n$ in the row-major order. Suppose that the processor P_i is initially given the values u_i and v_i, and the final result is to be read from processor P_n. Design an $O(\sqrt{n})$-time algorithm to solve the problem in this setting. **(10)**

Solution **Step 1:** Each processor P_i computes its contribution $z_i = u_i v_i$ in parallel. This takes $O(1)$ time.

Step 2: In this step, the rows run in parallel. Each node sends its partial sum to its right neighbor. At the end of $m-1$ transfers of partial sums, the rightmost node in the k-th row stores $W_k = z_{(m-1)k+1} + z_{(m-1)k+2} + \cdots + z_{mk}$. This step takes $O(m) = O(\sqrt{n})$ running time.

Step 3: Now, the partial sums based on $W_1, W_2, \ldots, W_m$ are transferred vertically downward by the rightmost nodes in the m rows. After $O(m) = O(\sqrt{n})$ time, the processor P_n computes the value $\mathbf{u} \cdot \mathbf{v} = W_1 + W_2 + \cdots + W_m$.

(b) Recall that the cost of a parallel algorithm is the product of its parallel running time with the number of processors used. The cost of the algorithm of Part (a) is $\mathrm{O}(\sqrt{n} \times m^2) = \mathrm{O}(n^{\frac{3}{2}})$, which is not cost-optimal, since an optimal sequential algorithm requires only $\mathrm{O}(n)$ operations. Establish that the dot product $\mathbf{u} \cdot \mathbf{v}$ can be computed by a cost-optimal parallel algorithm in $\mathrm{O}(n^{\frac{1}{3}})$ time on an $n^{\frac{1}{3}} \times n^{\frac{1}{3}}$ mesh. **(10)**

Solution For simplicity, let $l = n^{\frac{1}{3}}$ be an integer. The l^2 processors in the $l \times l$ mesh are numbered in the row-major order. Each processor P_i is given l pairs (u_j, v_j) for $j = (i-1)l+1, (i-1)l+2, \ldots, il$. The processors run in parallel, with each processor P_i sequentially computing the l-fold sum $Z_i = \sum_{j=(i-1)l+1}^{il} u_j v_j$ for the (u_j, v_j) pairs assigned to it. This takes $\mathrm{O}(l)$ running time.

After this, the processors send the partial sums, first row-wise, and then in the last column as in Steps 2 and 3 of Part (a). Since the mesh is of dimension $l \times l$, this transfer culminates in the final computation of $\mathbf{u} \cdot \mathbf{v}$ in P_{l^2} in $\mathrm{O}(l)$ time.

Thus, the cost of this algorithm is $\mathrm{O}(l \times l^2) = \mathrm{O}(l^3) = \mathrm{O}(n)$, which is optimal.

2. Let $A = (a_1, a_2, \ldots, a_n)$ be a sorted array that may contain duplicate entries. Your task is to prepare an output sorted array B composed of the elements of A but with all the duplicates removed. For example, if $A = (1,2,2,6,8,8,8,12,15,15,20)$, the output should be $B = (1,2,6,8,12,15,20)$. Develop an $O(\log n)$-time $O(n)$-work PRAM algorithm to solve this problem. What PRAM type does you algorithm use? **(8+2)**

Solution **Step 1:** We use an index array *IDX*, first to note the positions of the duplicate entries.

> **for** $i = 1,2,\ldots,n$ **pardo {**
>   **if** $(i = 1)$**, set** $IDX[i] = 1$
>   **else if** $(a_i \neq a_{i-1})$**, set** $IDX[i] = 1$
>   **else set** $IDX[i] = 0$
> **}**

This step takes $O(1)$ time and does $O(n)$ work.

Step 2: We do a parallel prefix computation on the array *IDX*. Suppose that the prefixes are stored in the array *IDX* itself. This step can be finished in $O(\log n)$ time using $O(n)$ work.

Step 3: If we use a CREW PRAM, the write-back to B may proceed as follows.

> **for** $i = 1,2,\ldots,n$ **pardo {**
>   **if** $(i = 1)$ **or** $(a_i \neq a_{i-1})$**, set** $B[IDX[i]] = a_i$
> **}**

On a common CRCW PRAM, the write-back can proceed unconditionally.

> **for** $i = 1,2,\ldots,n$ **pardo {**
>   **set** $B[IDX[i]] = a_i$
> **}**

In either case, Step 3 uses $O(1)$ parallel running time and $O(n)$ work.

3. You are given a polynomial $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, and a value of x. Your problem is to evaluate f at the given point x. Assume that you have a p-processor CREW PRAM with $p \leqslant n$. Develop a parallel algorithm to solve the problem in $\mathrm{O}(\frac{n}{p} + \log n)$ time on your PRAM. (**Hint:** Use Horner's rule and the WT scheduling principle.) **(10)**

Solution Use Horner's rule to write the polynomial expression as

$$f(x) = (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \cdots + a_1)x + a_0.$$

This computation can be expressed as a binary arithmetic expression tree E, in which each internal node stands for either a multiplication or an addition operation. The total number of nodes in E is $N = 4n + 1 = \Theta(n)$.

The parallel algorithm based on raking can evaluate the tree in $T(N) = \mathrm{O}(\log N)$ time using $W(N) = \mathrm{O}(N)$ work (operations).

By the WT scheduling principle, this algorithm can be scheduled to $p \leqslant n < N$ processors to run in time $\mathrm{O}(\frac{W(N)}{p} + T(n))$. Since $N = \Theta(n)$, this running time is $\mathrm{O}(\frac{n}{p} + \log n)$.

4. Let $A = (a_1, a_2, \ldots, a_n)$ be an array of n integers, among which most are zero. Your task is to locate the first index k for which $a_k \neq 0$ (assume that such a k exists). You are given a common CRCW PRAM. Propose an $O(1)$-time $O(n)$-work algorithm for solving this problem on your PRAM. (**Hint:** First solve the problem for arrays of size $\sqrt{n}$.) **(10)**

Solution First, let us solve this problem for an array X of size $m = \sqrt{n}$ (assumed to be an integer). Let $X = (x_1, x_2, \ldots, x_m)$ be an integer array with at least one non-zero entry. The following subroutine identifies the index of the first non-zero entry in X.

Subroutine S

```
if (x_1 ≠ 0) return 1
for i = 2,3,4,...,m and for j = 1,2,3,...,i−1 pardo {
    if (x_i ≠ 0) and (x_j = 0), set y_{i,j} = 1, else set y_{i,j} = 0.
}
for i = 2,3,4,...,m, initialize z_i = 0
for i = 2,3,4,...,m and for j = 1,2,3,...,i−1 {
    concurrently write y_{i,j} to z_i
}
```

The concurrent write of 1 succeeds only at the leftmost index k of a non-zero entry in X. The running time of this subroutine is $O(1)$, and the work done is $O(m^2) = O(n)$.

Let us now solve the original problem for A. We break A into m blocks, each of size m.

```
for i = 1,2,3,...,m, pardo x_i = 1.
for i = 1,2,3,...,m and for j = 1,2,3,...,m {
    Concurrently write to x_i the value 0 if a_{(i−1)m+j} = 0 or the value 1 if a_{(i−1)m+j} ≠ 0.
}
Call Subroutine S on X to get the leftmost k with x_k ≠ 0.
Call Subroutine S on A[(k−1)m+1...km] to get the leftmost l with a_{(k−1)m+l} ≠ 0.
return (k−1)m+l
```

If the i-th block of A of size m contains only zeros, the concurrent write of 0 at x_i succeeds, otherwise x_i continues to store 1. The first subroutine call identifies the block of A which contains the leftmost non-zero entry. We then search for the leftmost non-zero entry in that block of A.

The initialization of X and the concurrent write in it take $O(1)$ time and uses $O(n)$ work. The two calls of the subroutine S are on arrays of size $\sqrt{n}$, so each finishes in $O(1)$ time after doing $O(n)$ work.

Use this space for leftover answers and rough work