In-Class Midterm

(2:25 PM – 3:40 PM : 75 Minutes)

- This exam will account for either 10% or 20% of your overall grade depending on your relative performance in the midterm and the final. The higher of the two scores (midterm and final) will be worth 20% of your grade, and the lower one 10%.
- There are four (4) questions, worth 80 points in total. Please answer all of them in the spaces provided.
- There are 14 pages including two (2) blank pages. Please use the blank pages if you need additional space for your answers.
- Page 14 contains some useful bounds. No additional cheatsheets are allowed.
- Assume that the span of a parallel ***for*** loop with n iterations is $\Theta(\log n) + k$, where k is the maximum span of one iteration.

GOOD LUCK!

Question	Score	Maximum
1. Leftmost One		25
2. Prefix Sums		25
3. Balancing Resource Usage		25
4. Tighter Bound for the Greedy Scheduler		5
Total		80

NAME: ______________________________

Question 1. [**25 Points**] **Leftmost One.** We have already looked at the following problem in the class under a different name.

<u>Leftmost One</u>

> **Input.** A 0-1 bit array $A[1:n]$.
>
> **Output.** Smallest $k \in [1, n]$ such that $A[k] = 1$.

1(a) [**6 Points**] Find the work and span of the following agorithm for solving the Leftmost One probem.

```
Par-Leftmost-One(A)
n ← |A|
array B[1 : n]                              {B[i] will be set to 1 if A[i] is the leftmost 1}
parallel for i ← 1 to n do B[i] ← A[i]      {initially assume that each 1 is the leftmost 1}
parallel for i ← 1 to n do
    parallel for j ← 1 to i − 1 do          {compare A[i] with all A[j], j < i}
        if A[j] = 1 then B[i] ← 0  {if A[j] = 1 for some j < i, then A[i] is not the leftmost 1}
k ← 0
parallel for i ← 1 to n do                  {only for the leftmost A[i] = 1 we still have B[i] = 1}
    if B[i] = 1 then k ← i
return k                                    {return index of the leftmost 1}
```

1(*b*) [**10 Points**] Design an algorithm for solving the LEFTMOST ONE problem in $\Theta(n)$ work and $\Theta(\log n)$ depth (span) using the algorithm from part 1(*a*) as a subroutine. Provide pseudocode, and analysis of work and span.

[Hint: Split A into $\sqrt{n}$ segments.]

1(c) [**9 Points**] Given an array of n numbers each of which is an integer between 1 and n (not necessarily distinct) design an algorithm for finding the minimum number (value only) in $\Theta(n)$ work and $\Theta(\log n)$ depth (span) using your algorithm from part 1(b) as a subroutine. Provide pseudocode, and analysis of work and span.

Use this page if you need additional space for your answers.

Question 2. [**25 Points**] **Prefix Sums.** Consider the following problem covered in the class.

<u>Prefix Sums</u>

> **Input.** An array $A[1:n]$ of n elements with a binary associative operation $\oplus$.
>
> **Output.** An array $S[1:n]$, where $S[i] = A[1] \oplus A[2] \oplus \ldots \oplus A[i]$ for $i \in [1, n]$.

2(a) [**8 Points**] The following algorithm solves Prefix Sums when called as Par-Prefix-Sums$(A, 1, n, \oplus, S)$. Write down the recurrence relations for work and span of the algorithm, and solve them.

```
Par-Prefix-Sums(A, q, r, ⊕, S)
    if q = r then S[q] ← A[q]
    else
        m ← ⌊(q+r)/2⌋                                   {split the array into two halves}
        parallel : Par-Prefix-Sums(A, q, m, ⊕, S)         {find prefix sums for the left half}
                   Par-Prefix-Sums(A, m + 1, r, ⊕, S)     {find prefix sums for the right half}
        parallel for i ← m + 1 to r do
            S[i] ← S[i] ⊕ S[m]                      {update right half with the sum of the left half}
```

2(b) [**10 Points**] Design a work-optimal algorithm for PREFIX SUMS using PAR-PREFIX-SUMS from part 2(a) as a subroutine. Provide pseudocode, and analysis of work and span.

[Hint: Contract array A.]

2(c) [**7 Points**] Design a work-optimal parallel algorithm to evaluate the following polynomial of degree $n-1$, where $a_0, a_1, \ldots, a_{n-1}$ are given constants.

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

Provide pseudocode, and analysis of work and span.

[Hint: Use your work-optimal parallel prefix algorithm from part 2(b).]

Use this page if you need additional space for your answers.

Question 3. [**25 Points**] **Balancing Resource Usage.** Suppose we have 2 processors (X and Y), n jobs and n resources. Job i ($1 \leq i \leq n$) is specified as a vector $\langle a_{i,1}, a_{i,2}, \ldots, a_{i,n} \rangle$, where,

$$a_{i,j} = \begin{cases} 1, & \text{if job } i \text{ uses resource } j, \\ 0, & \text{otherwise.} \end{cases}$$

Each job must be assigned to either processor X or processor Y, and these assignment are given by the vector $\langle b_1, b_2, \ldots, b_n \rangle$, where,

$$b_i = \begin{cases} +1, & \text{if job } i \text{ assigned to processor } X, \\ -1, & \text{otherwise.} \end{cases}$$

Our goal is to find a vector b that balances the workload between X and Y by minimizing the maximum imbalance in the usage of any resource, that is, by minimizing $\Delta = \max_{1 \leq i \leq n} |c_i|$, where,

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ \ldots \\ b_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \ldots \\ \ldots \\ c_n \end{bmatrix}$$

Observe that each c_i ($= \sum_{j=1}^{n} a_{i,j} b_j$) is the sum of n terms, each of which is either 0, +1 or −1. Let

$$\begin{aligned} X_i &= \text{number of terms with value } +1 \text{ in } c_i, \\ Y_i &= \text{number of terms with value } -1 \text{ in } c_i, \\ k_i &= \text{number of 1's among } a_{i,1}, a_{i,2}, \ldots, a_{i,n}, \text{ and} \\ \beta &= \sqrt{12 n \ln n}. \end{aligned}$$

Then clearly, $X_i + Y_i = k_i$, $X_i - Y_i = c_i$, and $|c_i| \leq k_i$.

We will show that good load balancing (i.e., $\Delta < \beta$) can be achieved even if we choose the entries of b independently and uniformly at random, that is, with $Pr\left[b_i = +1\right] = Pr\left[b_i = -1\right] = \frac{1}{2}$.

3(a) [**6 Points**] Show that if $|c_i| \leq \beta$ then $\frac{k_i}{2}\left(1 - \frac{\beta}{k_i}\right) \leq X_i \leq \frac{k_i}{2}\left(1 + \frac{\beta}{k_i}\right)$.

3(b) [**4 Points**] Show that $E[X_i] = \frac{k_i}{2}$.

3(c) [**10 Points**] Clearly, $k_i \leq \beta \Rightarrow |c_i| \leq \beta$. Prove that even for $k_i > \beta$, $Pr\left[|c_i| \geq \beta\right] \leq \frac{2}{n^2}$.

3(d) [**5 Points**] Show that w.h.p. $\Delta \leq \beta$.

Question 4. [**5 Points**] **Tighter Bound for the Greedy Scheduler.** We proved in the class that on an ideal parallel computer with p processing elements, a gready scheduler executes a multithreaded computation with work T_1 and span T_∞ in time $T_p \leq \frac{T_1}{p} + T_\infty$. We came up with this bound by showing that the number of complete steps (where all p processors have work to do) is at most $\frac{T_1}{p}$, and the number of incomplete steps (where some processors are idle, but at least one has work to do) is at most T_∞, and by observing that $T_p \leq$ #complete steps + #incompete steps.

4(a) [**5 Points**] Argue that the bound above can be improved to $T_p \leq \frac{T_1 - T_\infty}{p} + T_\infty$.

Some Useful Bounds

Master Theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise,} \end{cases}$$

where, $\frac{n}{b}$ is interpreted to mean either $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$. Then $T(n)$ has the following bounds:

Case 1: If $f(n) = \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$ for some constant $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.

Case 2: If $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ for some constant $k \geq 0$, then $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$.

Case 3: If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some constant $\epsilon > 0$, and $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta\left(f(n)\right)$.

Markov's Inequality. Let X be a random variable that assumes only nonnegative values. Then for all $\delta > 0$, $Pr\left[X \geq \delta\right] \leq \frac{E[X]}{\delta}$.

Chebyshev's Inequality. Let X be a random variable with a finite mean $E[X]$ and a finite variance $Var[X]$. Then for any $\delta > 0$, $Pr\left[|X - E[X]| \geq \delta\right] \leq \frac{Var[X]}{\delta^2}$.

Chernoff Bounds. Let $X_1, \ldots, X_n$ be independent Poisson trials, that is, each X_i is a 0-1 random variable with $Pr[X_i = 1] = p_i$ for some p_i. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then the following bounds hold.

(1) For any $\delta > 0$, $Pr\left[X \geq (1+\delta)\mu\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$.

(2) For $0 < \delta < 1$, $Pr\left[X \geq (1+\delta)\mu\right] \leq e^{-\frac{\mu\delta^2}{3}}$.

(3) For $0 < \gamma < \mu$, $Pr\left[X \geq \mu + \gamma\right] \leq e^{-\frac{\gamma^2}{3\mu}}$.

(4) For $0 < \delta < 1$, $Pr\left[X \leq (1-\delta)\mu\right] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu}$.

(5) For $0 < \delta < 1$, $Pr\left[X \leq (1-\delta)\mu\right] \leq e^{-\frac{\mu\delta^2}{2}}$.

(6) For $0 < \gamma < \mu$, $Pr\left[X \leq \mu - \gamma\right] \leq e^{-\frac{\gamma^2}{2\mu}}$.