In-Class Final Exam (Solution Ideas)

(11:35 AM – 12:50 PM : 75 Minutes)

- This exam will account for either 15% or 30% 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 30% of your grade, and the lower one 15%.
- There are four (4) questions, worth 75 points in total. Please answer all of them in the spaces provided.
- There are 14 pages including four (4) blank pages. Please use the blank pages if you need additional space for your answers.
- The exam is *open slides*. So you can consult the lecture slides during the exam. No additional cheatsheets are allowed.

GOOD LUCK!

Question	Score	Maximum
1. Binary Addition		25
2. Nuts and Bolts		15
3. Partial Sums		10
4. Sampling		25
Total		75

NAME: ____

Question 1. [25 Points] Binary Addition. Function Standard-Binary-Addition is the standard grade school algorithm for adding two n-bit binary numbers, and Rec-Binary-Addition is a recursive divide-and-conquer algorithm for the same task. In this problem we will try to parallelize the two algorithms and analyze the resulting parallel algorithms.

```
Standard-Binary-Addition( $x_n \dots x_1,\ y_n \dots y_1$ )
(Inputs are two $n$-bit binary numbers $X = x_n \dots x_1$ and $Y = y_n \dots y_1$, where $n \geq 1$. Output is an $(n+1)$-bit binary
number $Z = z_{n+1} \dots z_1$, where $Z$ is the sum of $X$ and $Y$ (i.e., $Z^c = X + Y$).)
1. $c \leftarrow 0$
2. for $i \leftarrow 1$ to $n$ do
3.     $s \leftarrow x_i + y_i + c$
4.     $z_i \leftarrow s$ mod $2$,  $c \leftarrow s$ div $2$
5. $z_{n+1} \leftarrow c$
6. return $Z$
```

```
Rec-Binary-Addition( $x_n \dots x_1,\ y_n \dots y_1$ )
(Inputs are two $n$-bit binary numbers $X = x_n \dots x_1$ and $Y = y_n \dots y_1$, where $n \geq 1$ is assumed to be a power of 2.
Outputs are two $(n+1)$-bit binary numbers $Z^c = z^c_{n+1} \dots z^c_1$ and $Z = z_{n+1} \dots z_1$, where $Z^c$ is the sum of $X$ and $Y$
assuming an initial carry (i.e., $Z^c = X + Y + 1$), and $Z$ is the same sum without an initial carry (i.e., $Z = X + Y$).)
1.  if $n = 1$ then
2.      if $x_1 = y_1 = 0$ then return $\langle\ 01,\ 00\ \rangle$
3.      elif $x_1 = y_1 = 1$ then return $\langle\ 11,\ 10\ \rangle$
4.      else return $\langle\ 10,\ 01\ \rangle$
5.  else
6.      let $X_h = x_n \dots x_{\frac{n}{2}+1}$ and $X_l = x_{\frac{n}{2}} \dots x_1$        {split $X$ at the midpoint}
7.      let $Y_h = y_n \dots y_{\frac{n}{2}+1}$ and $Y_l = y_{\frac{n}{2}} \dots y_1$        {split $Y$ at the midpoint}
8.      $\langle\ L^c,\ L\ \rangle \leftarrow$ Rec-Binary-Addition ( $X_l,\ Y_l$ )   {where $L^c = l^c_{\frac{n}{2}+1} l^c_{\frac{n}{2}} \dots l^c_1$ and $L = l_{\frac{n}{2}+1} l_{\frac{n}{2}} \dots l_1$}
9.      $\langle\ H^c,\ H\ \rangle \leftarrow$ Rec-Binary-Addition ( $X_h,\ Y_h$ )   {where $H^c = h^c_{\frac{n}{2}+1} h^c_{\frac{n}{2}} \dots h^c_1$ and $H = h_{\frac{n}{2}+1} h_{\frac{n}{2}} \dots h_1$}
10.     if $l_{\frac{n}{2}+1} = 1$ then Copy( $Z,\ H^c,\ l_{\frac{n}{2}} \dots l_1$ )   {(carry from position $\frac{n}{2}$) $\Rightarrow Z = h^c_{\frac{n}{2}+1} h^c_{\frac{n}{2}} \dots h^c_1 l_{\frac{n}{2}} \dots l_1$}
11.     else Copy( $Z,\ H,\ l_{\frac{n}{2}} \dots l_1$ )   {(no carry from position $\frac{n}{2}$) $\Rightarrow Z = h_{\frac{n}{2}+1} h_{\frac{n}{2}} \dots h_1 l_{\frac{n}{2}} \dots l_1$}
12.     if $l^c_{\frac{n}{2}+1} = 1$ then Copy( $Z^c,\ H^c,\ l^c_{\frac{n}{2}} \dots l^c_1$ )   {(carry from position $\frac{n}{2}$) $\Rightarrow Z^c = h^c_{\frac{n}{2}+1} h^c_{\frac{n}{2}} \dots h^c_1 l^c_{\frac{n}{2}} \dots l^c_1$}
13.     else Copy( $Z^c,\ H,\ l^c_{\frac{n}{2}} \dots l^c_1$ )   {(no carry from position $\frac{n}{2}$) $\Rightarrow Z^c = h_{\frac{n}{2}+1} h_{\frac{n}{2}} \dots h_1 l^c_{\frac{n}{2}} \dots l^c_1$}
14.     return $\langle\ Z^c,\ Z\ \rangle$
```

```
Copy( $z_{2n+1} \dots z_1,\ h_{n+1} \dots h_1,\ l_n \dots l_1$ )
1. for $i \leftarrow 1$ to $n$ do
2.     $z_i \leftarrow l_i$
3.     $z_{n+i} \leftarrow h_i$
4. $z_{2n+1} \leftarrow h_{n+1}$
```

1(a) [**3 Points**] Can you replace the ***for*** loop in lines 2–4 of STANDARD-BINARY-ADDITION with a ***parallel for*** loop (without changing anything else)? Justify your answer.

Solution. Observe that for each $i \in (1, n]$, the value of c in interation i of the ***for*** loop depends on all iterations preceding iteration i. So, we cannot replace the serial ***for*** loop with a ***parallel for*** loop.

1(b) [**6 Points**] Which parts of REC-BINARY-ADDITION can be executed in parallel? Justify your answer.

Solution. The two recursive calls in lines 8 and 9 of REC-BINARY-ADDITION can be executed in parallel because they do not depend on each other (the input values passed to the two recursive calls do not change over time, and the output locations are disjoint). The same is true for the two ***if*** conditions in lines 10–13, and so lines 10–11 can be excuted in parallel of lines 12–13. Also each iteration of the ***for*** loop in lines 1–3 of COPY is independent of all other iterations of the loop. Hence, that ***for*** loop can be replaced with a ***parallel for*** loop.

1(c) [**10 Points**] Write down the recurrence relations for *work* and *span* for your parallel version of REC-BINARY-ADDITION from part 1(b), and solve them. 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.

Solution. The recurrence relations are as follows.

$$\text{Work}, T_1(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 2T_1\left(\frac{n}{2}\right) + \Theta(n) & \text{otherwise.} \end{cases}$$

$$\text{Span}, T_\infty(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ T_\infty\left(\frac{n}{2}\right) + \Theta(\log n) & \text{otherwise.} \end{cases}$$

Applying Master Theorem case 2: $T_1(n) = \Theta(n \log n)$ and $T_\infty(n) = \Theta\left(\log^2 n\right)$.

1(d) [**4 Points**] Find the parallel running time (i.e., T_p) and the parallelism of the parallel REC-BINARY-ADDITION from part 1(b).

Solution.
Parallel running time, $T_p\left(n\right) = \Theta\left(\frac{T_1(n)}{p} + T_\infty\left(n\right)\right) = \Theta\left(\frac{n \log n}{p} + \log^2 n\right)$.

Parallelism, $P = \frac{T_1(n)}{T_\infty(n)} = \Theta\left(\frac{n}{\log n}\right)$.

1(e) [**2 Points**] Is your parallel REC-BINARY-ADDITION work-optimal? Justify your answer.

Solution. Since the size of the input is $\Theta\left(n\right)$ no serial algorithm can add two n-bit binary numbers in $o\left(n\right)$ time. But clearly STANDARD-BINARY-ADDITION solves the problem in $\mathcal{O}\left(n\right)$ time. Hence, the running time of the optimal serial algorithm is $T_s\left(n\right) = \Theta\left(n\right)$.

But $p \cdot T_p\left(n\right) = \Theta\left(n \log n + p \log^2 n\right) = \omega\left(T_s\left(n\right)\right)$.

Hence, the algorithm is not work-optimal.

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

Question 2. [**15 Points**] **Nuts and Bolts.** Suppose you are given n nuts and n bolts. Each nut has a different size and it matches exactly one bolt and vice versa. Your job is to find the matching bolt for every nut in the set. When you solve this problem you can only compare a nut with a bolt and conclude that either the nut exactly fits the bolt or it is too small for the bolt or it is too large. You are not allowed to compare one nut with another nut or a bolt with another bolt.

2(a) [**5 Points**] Suppose you have chosen a nut randomly from the given set. Show that in $\Theta(n)$ time you can find the matching bolt, and divide the remaining nuts and bolts into two sets: one containing all nuts and bolts smaller than the chosen pair and the other containing everything larger.

Solution. We first compare each bolt with the chosen nut in order to find the one that fits. This sequence of comparisons also divides the remaining $n-1$ bolts into two sets $\mathcal{B}_l$ and $\mathcal{B}_r$, where $\mathcal{B}_l$ contains all bolts smaller than the matching bolt and $\mathcal{B}_r$ contains all larger bolts. Then we compare each of the remaining $n-1$ nuts with the bolt that fits the chosen nut, and divide them into two sets $\mathcal{N}_l$ and $\mathcal{N}_r$ with $\mathcal{N}_l$ containing all nuts smaller than the chosen nut and $\mathcal{N}_r$ containing all larger nuts. Clearly, all these operations take a total of $\Theta(n)$ time.

Observe that each nut in $\mathcal{N}_l$ (resp. $\mathcal{N}_r$) must have its matching bolt in $\mathcal{B}_l$ (resp. $\mathcal{B}_r$), and vice versa.

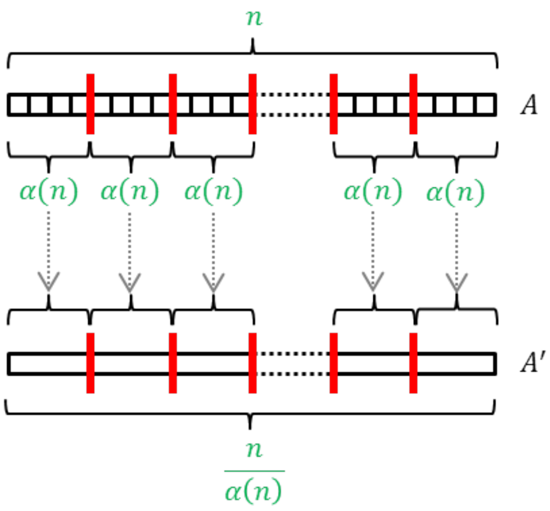
2(b) [**10 Points**] Prove that using the partition algorithm in part 2(a) you can solve the nuts and bolts problem in $\mathcal{O}\left(n\log n\right)$ time with high probability. You do not necessarily need to prove the bound explicitly. You can simply show that your algorithm is structurally exactly similar to an algorithm you have already seen in the class, and so the bound proved for that algorithm applies.

Solution. We can use an algorithm structurally exactly similar to the randomized quicksort algorithm we saw in the class. As in that quicksort algorithm the partition algorithm chooses a random pivot (a nut), and divides the problem into two disjoint subproblems in $\Theta\left(n\right)$ time. One subproblem is to match all nuts and bolts smaller than the pivot nut/bolt, and the other one is to match all nuts and bolts larger than the pivot. Both subproblems can be solved recursively. Hence, the complexity analysis of that algorithm also applies here, and our proposed nuts and bolts matching algorithm runs in $\mathcal{O}\left(n\log n\right)$ time with high probability.

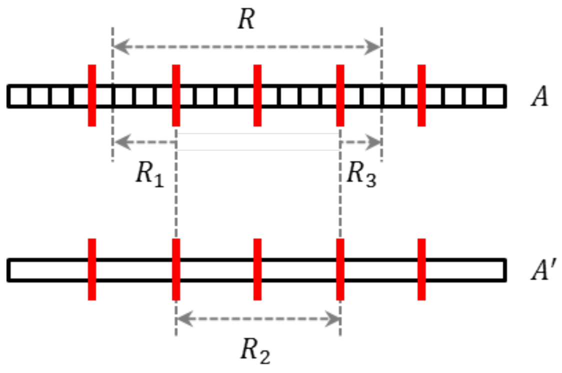
Use this page if you need additional space for your answers.

Question 3. [**10 Points**] **Partial Sums.** In the class we have shown that the *partial semigroup sums* problem can be solved using $\Theta(n\alpha(n))$ space with $\mathcal{O}(\alpha(n))$ applications of the associative operator per query. How do you reduce the space complexity to $\Theta(n)$ without changing the query complexity asymptotically?

Solution. Let A be the input array of length n. We divide A into $\frac{n}{\alpha(n)}$ blocks of length $\alpha(n)$ each, and construct another array A' such that $A'[i]$ simply stores the sum of all entries in the i-th block of A, where $1 \leq i \leq \frac{n}{\alpha(n)}$. We now construct a partial sums data structure on A' which will use $\Theta\left(\left(\frac{n}{\alpha(n)}\right)\alpha\left(\frac{n}{\alpha(n)}\right)\right) = \mathcal{O}(n)$ space, and will support queries using $\mathcal{O}\left(\alpha\left(\frac{n}{\alpha(n)}\right)\right) = \mathcal{O}(\alpha(n))$ applications of the associative operator per query. Thus total space used by A and A' is $\Theta(n)$.



Now any query range R in A can be decomposed into at most 3 disjoint (possibly empty) query subranges: R_1, R_2 and R_3. Among those each of R_1 and R_3 is a query inside a block of size $\alpha(n)$ in array A, can be answered naïvely simply summing up all entries in the block corresponding to the subrange. Since the size of the block is $\alpha(n)$ each of those two queries will require $\mathcal{O}(\alpha(n))$ applications of the operator. The R_2 subrange is a query in array A', and so can be answered using only $\mathcal{O}(\alpha(n))$ applications of the operator. Hence, query complexity of R remains $\mathcal{O}(\alpha(n))$.



Question 4. [**25 Points**] **Sampling.** In this problem we will analyze the properties of two different sampling of elements from an array.

4(*a*) [**10 Points**] One can show that given an array A of n distinct numbers no deterministic algorithm can find an element among the smallest $\frac{n}{k}$ members of A using fewer than $\left(\frac{k-1}{k}\right)n$ comparisons. But consider the function given below that runs in $\Theta\left(\log_{\left(\frac{k}{k-1}\right)} n\right)$ time. Prove that with high probability in n the element it returns is among the smallest $\frac{n}{k}$ elements of A.

```
TOP( $A[\,1:n\,]$, $k$ )
(Inputs are an array $A[\,1:n\,]$ of $n$ distinct numbers, and an integer $k \in [1,n]$. Output is
an element $x$ of $A$.)
  1. $x \leftarrow A[\,\text{RANDOM}(\,1,n\,)\,]$            {choose a number from A uniformly at random}
  2. for $i \leftarrow 1$ to $\left\lceil \log_{\frac{k}{k-1}} n \right\rceil$ do
  3.     $y \leftarrow A[\,\text{RANDOM}(\,1,n\,)\,]$ {choose another number from A uniformly at random}
  4.     if $y < x$ then $x \leftarrow y$                      {keep the smaller number}
  5. return $x$
```

Solution. Let x_0 be the element chosen in line 1, and for $1 \le i \le \lceil m \rceil$, let x_i be the element chosen in line 3 in iteration i of the ***for*** loop in lines 2–4, where $m = \log_{\left(\frac{k}{k-1}\right)}(n)$. Let $rank_{(A)}(x)$ denote the number of elements in A not larger than x.

Now clearly, for $0 \le i \le \lceil m \rceil$, $Pr\left(rank_{(A)}(x_i) > \frac{n}{k}\right) \ge 1 - \frac{1}{k}$.

$\therefore Pr\left(rank_{(A)}\left(\min_{i=0}^{\lceil m \rceil}\{x_i\}\right) > \frac{n}{k}\right) \le \left(\frac{k-1}{k}\right)^{\lceil m \rceil + 1} < \left(\frac{k-1}{k}\right)^{m} = \frac{1}{n}$.

Hence, $Pr\left(rank_{(A)}\left(\min_{i=0}^{\lceil m \rceil}\{x_i\}\right) \le \frac{n}{k}\right) > 1 - \frac{1}{n}$.

4(b) [**15 Points**] Consider the function given below, and prove that with high probability in n no $\left\lceil \frac{2k}{k-1} \right\rceil$ numbers in $B[\ 1 : m\ ]$ are equal.

```
SAMPLE( $A[\ 1:n\ ],\ k$ )
(Inputs are an array $A[\ 1:n\ ]$ of $n$ distinct numbers, and a real number $k > 1$. Output is
an array $B[\ 1:m\ ]$ of $m = \left\lfloor n^{\frac{1}{k}} \right\rfloor$ numbers sampled uniformly at random from $A$.)
    1. array $B[\ 1:m\ ]$
    2. for $i \leftarrow 1$ to $m$ do
    3.     $j \leftarrow$ RANDOM( $1, n$ )        {choose an integer uniformly at random from $[\ 1, n\ ]$}
    4.     $B[\ i\ ] \leftarrow A[\ j\ ]$
    5. return $B$
```

Solution. Let us fix a specific index t of A, and define:

$$X_i = \begin{cases} 1 & \text{if in the } i\text{-th iteration of the } \textbf{\textit{for}} \text{ loop } j = t, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $Pr\left(X_i = 1\right) = \frac{1}{n}$, and $E\left[X_i\right] = 1 \times Pr\left(X_i = 1\right) = \frac{1}{n}$.

Now if X is the total number of times $A[t]$ is picked, then $X = \sum_{i=1}^{m} X_i$.

So, $\mu = E\left[X\right] = \sum_{i=1}^{m} E\left[X_i\right] = \frac{m}{n} \le \frac{n^{\frac{1}{k}}}{n} = n^{\frac{1-k}{k}}$.

We will use the following Chernoff bound: $Pr\left(X \ge (1+\delta)\mu\right) \le \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$, where $\delta > 0$.

Let $l = \frac{2k}{k-1}$. We put $(1+\delta)\mu = l$ in the bound above. Then $\delta = \left(\frac{2k}{k-1}\right) \times \frac{1}{\mu} - 1 \ge= \left(\frac{2k}{k-1}\right) \times n^{\frac{k-1}{k}} - 1$, and clearly, $\delta > 0$.

Then we have: $Pr\left(X \ge l\right) \le \frac{e^{\delta\mu}}{(1+\delta)^{(1+\delta)\mu}} = \frac{e^{l-\mu}}{\left(\frac{l}{\mu}\right)^{l}} = \left(\frac{e}{l}\right)^{l} \times \frac{1}{e^{\mu}} \times \mu^{l}$.

But $\mu \ge 0 \Rightarrow e^{\mu} \ge 1$, and since $l = \frac{2k}{k-1}$ is a constant, so is $c = \left(\frac{e}{l}\right)^{l}$. Hence, $Pr\left(X \ge \left\lceil \frac{2k}{k-1} \right\rceil\right) \le Pr\left(X \ge \frac{2k}{k-1}\right) \le c \times \mu^{l} \le \frac{c}{n^{\left(\frac{k-1}{k}\right) \times l}} = \frac{c}{n^2}$. In other words, the probability that $A[t]$ appears $\left\lceil \frac{2k}{k-1} \right\rceil$ times or more in B is at most $\frac{c}{n^2}$.

$\therefore$ The probability that any of the n entries of A appears $\left\lceil \frac{2k}{k-1} \right\rceil$ times or more in B is at most $n \times \frac{c}{n^2} = \frac{c}{n}$. Hence, the probability that no entry of A appears $\left\lceil \frac{2k}{k-1} \right\rceil$ times or more in B is at least $1 - \frac{c}{n}$. In other words, w.h.p. in n no $\left\lceil \frac{2k}{k-1} \right\rceil$ entries of B are equal.

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

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