CSE548, AMS542: Analysis of Algorithms, Fall 2019 **Date:** Dec 2, 2019

Final In-Class Exam

(7:10 PM – 9:10 PM : 120 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 three (3) questions, worth 75 points in total. Please answer all of them in the spaces provided.
- There are 20 pages including six (6) blank pages and one (1) page of appendix. Please use the blank pages if you need additional space for your answers.
- The exam is *open slides* and *open notes*. But *no books* and *no electronic devices* (e.g., computers, calculators, cell phones, etc.).

Good Luck!

Question	Pages	Score	Maximum
1. A Queue from Three Stacks	2–6		25
2. Copying Arrays	9–12		30
3. Parallelize a Recursive Function	15–17		20
Total			75

Name: _____

```
INIT( )                                                  {initialize FIFO (First In, First Out) queue}
    1. S1 ← ∅,  S2 ← ∅,  S3 ← ∅                          {three stacks which are emptied initially}

TRANSFER( Sin, Sout)                                     {transfer contents of stack Sin to stack Sout}
    1. while Sin ≠ ∅ do                                  {transfer all items of Sin}
    2.     x ← Sin.POP( )                                {pop an item from Sin}
    3.     Sout.PUSH( x )                                {push it to Sout}

ENQ( x )                                                 {enqueue key x into the queue}
    1. if S2 = ∅ and S3 = ∅ then                         {if the queue is empty}
    2.     S2.PUSH( x )                                  {push x into stack S2 to maintain invariant}
    3. else
    4.     S1.PUSH( x )                                  {otherwise push into S1}
    5.     if S1.SIZE( ) > S2.SIZE( ) + S3.SIZE( ) then  {invariant violated}
    6.         TRANSFER( S2, S3 )                        {move all items from S2 to S3}
    7.         TRANSFER( S1, S2 )                        {move all items from S1 to S2}

DEQ( )                                                   {dequeue the oldest key from the queue}
    1. if S2 = ∅ and S3 = ∅ then return NIL              {queue is empty}
    2. TRANSFER( S3, S2 )                                {move the items from S3 back to S2}
    3. x ← S2.POP( )                                     {S2 must be nonempty by invariant}
    4. if S1.SIZE( ) > S2.SIZE( ) then                   {invariant violated}
    5.     TRANSFER( S2, S3 )                            {move all items from S2 to S3}
    6.     TRANSFER( S1, S2 )                            {move all items from S1 to S2}
    7. return x
```

Figure 1: A FIFO queue is implemented using three LIFO stacks S_1, S_2 and S_3.

QUESTION 1. [25 Points] A Queue from Three Stacks. A *queue* is a dynamic collection of items that supports two operations: ENQ and DEQ. An ENQ(x) operation inserts the item x into the queue, while a DEQ( ) operation removes the current oldest item from the collection. Thus a queue works on a "First In, First Out" or FIFO principle.

A *stack*, on the other hand, is a collection of items that also supports two operations: PUSH and POP. A PUSH(x) operation inserts the item x into the stack, while a POP( ) operation removes the newest item (i.e., the item that has been in the set for the shortest time) from the collection. Thus a stack works on a "Last In, First Out" or LIFO principle.

Figure 1 shows how to implement a queue using three stacks S_1, S_2 and S_3. Figure 2 shows the state of the data structure after each of a sequence of ENQ and DEQ operations performed on it.

Assuming that both PUSH and POP operations can be performed on the given stacks in $\Theta(1)$ worst-case cost per operation, this task asks you to determine the worst-case and amortized costs of ENQ and DEQ operations on the queue as implemented in Figure 1.

The queue always maintains the following **invariant**: $S_1.\text{SIZE}( ) \leq S_2.\text{SIZE}( ) + S_3.\text{SIZE}( )$.

As soon as the invariant is violated, it is fixed first by moving all items from S_2 to S_3, and then emptying S_1 by moving all its items to S_2.

Before performing a DEQ operation we first move the items (if any) from S_3 back to S_2, and then POP an item from S_2 to return.

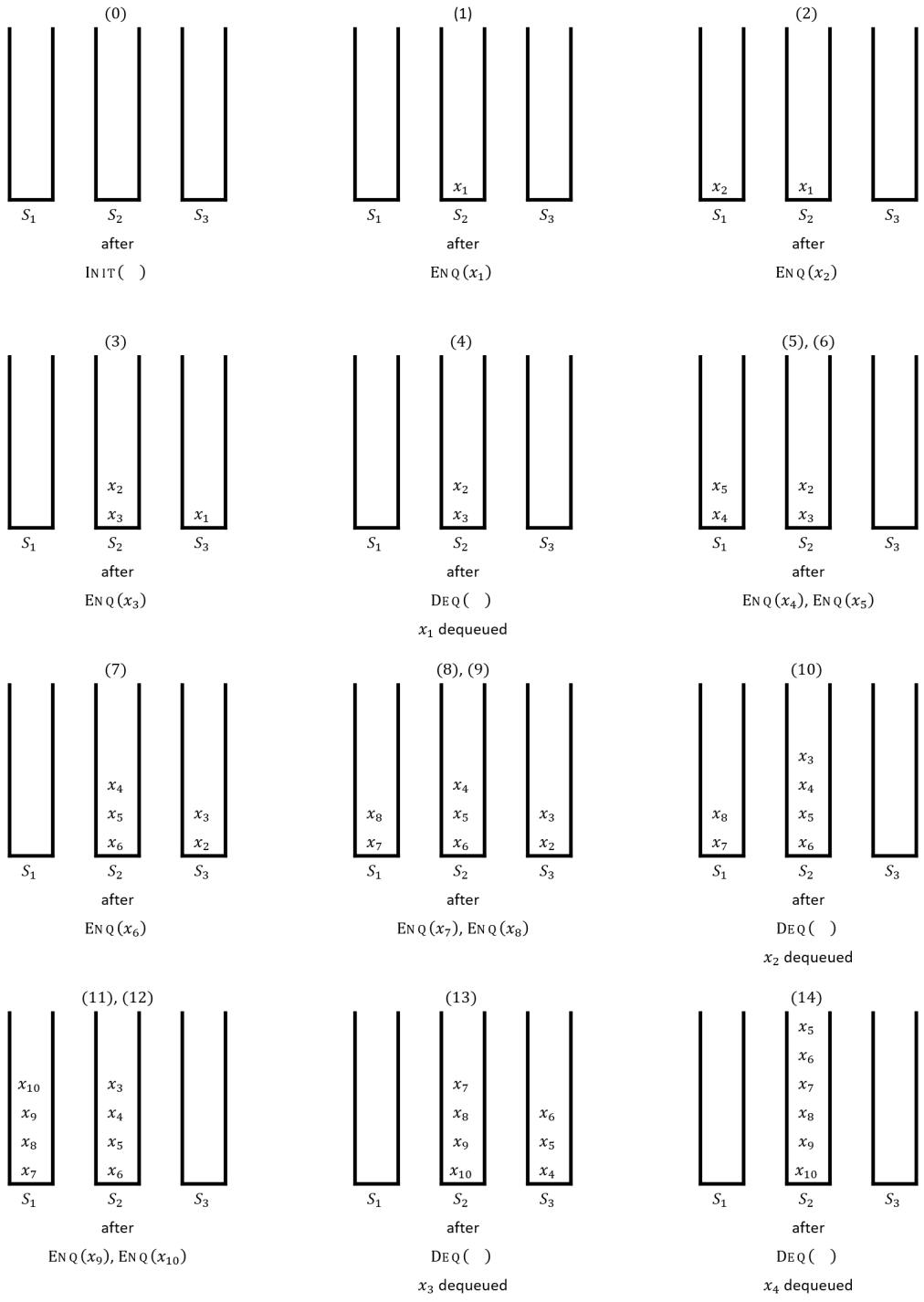

Figure 2: State of the queue after each of a sequence of fourteen ENQ and DEQ operations.

1(a) [**5 Points**] What is the worst-case cost of each of the following operations when the queue contains n items: (i) ENQ(x) and (ii) DEQ( )? Justify your answers.

1(b) [**15 Points**] Use either the accounting method or the potential method to show that the amortized cost of each of the following operations is $\Theta(1)$: (i) ENQ(x) and (ii) DEQ( ).

1(c) [**5 Points**] Suppose after executing INIT we perform an intermixed sequence of n ENQ and DEQ operations on the queue as implemented in Figure 1. What is the total worst-case cost of performing these n operations based on your results from part 1(a)? What is the total worst-case cost based on your results from part 1(b)?

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

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

Question 2. [30 Points] Copying Arrays. In this task you are asked to analyze the performance of two functions that copy the contents of one array to random locations of another.

```
RANDOM-COPY-ONE( X[ 1 : n ], Y[ 1 : 2n ] )
(Copy the contents of array X[1 : n] to n random locations of array Y[1 : 2n], and set each of the remaining n locations of Y[1 : 2n] to NIL.)
1. for i ← 1 to 2n do Y[i] ← NIL
2. i ← 1
3. while i ≤ n do
4.     j ← an integer chosen uniformly at random from [1, 2n]
5.     if j < n + i and Y[j] = NIL then
6.         Y[j] ← X[i]
7.         i ← i + 1
```

Figure 3: Copy the n items of array $X[1 : n]$ to n random locations of array $Y[1 : 2n]$.

$2(a)$ **[5 Points]** Figure 3 shows a function RANDOM-COPY-ONE that copies the n items of array $X[\ 1 : n\ ]$ to n random locations of array $Y[\ 1 : 2n\ ]$. Argue that in each iteration of the ***while*** loop of lines 3–7, the assignment in line 6 is executed with probability $\frac{1}{2}$.

2(b) [**5 Points**] Argue that the expected number of times the body of the ***while*** loop of lines 3–7 of RANDOM-COPY-ONE is executed is $2n$.

.

2(c) [**10 Points**] Prove that with high probability in n the body of the ***while*** loop of lines 3–7 of RANDOM-COPY-ONE will not be executed more than $4n$ times.

.

```
RANDOM-COPY-TWO( X[ 1 : n ], Y[ 1 : n ] )
(Copy the contents of array X[1 : n] to random locations of array Y[1 : n].)
 1. for i ← 1 to n do Y[i] ← NIL
 2. i ← 1
 3. while i ≤ n do
 4.     j ← an integer chosen uniformly at random from [1, n]
 5.     if Y[j] = NIL then
 6.         Y[j] ← X[i]
 7.         i ← i + 1
```

Figure 4: Copy the n items of array $X[1:n]$ to n random locations of array $Y[1:n]$.

2(d) [**10 Points**] Now consider the function RANDOM-COPY-TWO given above which copies each item of $X[1:n]$ to a random location of $Y[1:n]$. Prove that with high probability in n the body of the ***while*** loop of lines 3–7 of RANDOM-COPY-2 will not be executed more than $2n \ln n$ times.

.

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

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

Question 3. [20 Points] Parallelize a Recursive Function. The recursive function Update($X[\,1:n\,],\ Y[\,1:n\,],\ Z[\,1:n\,]$) shown in Figure 5 updates array $Z[1:n]$ based on values stored in $X[1:n]$ and $Y[1:n]$. All three input arrays are assumed to be mutually disjoint. Assume for simplicity that $n = 2^k$ for some integer $k \geq 0$.

Your task will be to parallelize Update.

Update($X[\,1:n\,],\ Y[\,1:n\,],\ Z[\,1:n\,]$)

(Arrays $X[1:n]$, $Y[1:n]$ and $Z[1:n]$ are mutually disjoint. Array $Z[1:n]$ is updated based on the values (positive integers) stored in $X[1:n]$ and $Y[1:n]$. This function only reads from $X[1:n]$ and $Y[1:n]$ – it never writes to them. A function $f(\cdot,\cdot)$ is used in the base case which accepts two positive integers as inputs and returns a positive integer as the output. We assume $n > 0$ to be an integral power of 2.)

1. ***if*** $n = 1$ ***then*** {*base case*}
2. $\quad Z[1] \leftarrow \max\{\ Z[1],\ f(\ X[1],\ Y[1]\ )\ \}$
3. ***else***
4. $\quad$ let X_L and X_R denote $X[\,1:\frac{n}{2}\,]$ and $X[\,\frac{n}{2}+1:n\,]$, respectively. Similarly for Y and Z.
5. $\qquad$ Update($X_L,\ Y_L,\ Z_L$)
6. $\qquad$ Update($X_L,\ Y_R,\ Z_R$)
7. $\qquad$ Update($X_R,\ Y_L,\ Z_L$)
8. $\qquad$ Update($X_R,\ Y_R,\ Z_R$)
9. $\qquad$ Update($Z_L,\ Y_R,\ Z_R$)

Figure 5: Updates array $Z[1:n]$ based on values stored in $X[1:n]$ and $Y[1:n]$. Assumes that each entry of $Z[1:n]$ has already been initialized to 0 before making the initial function call.

3(a) [**10 Points**] Show how you will parallelize UPDATE by writing down ***spawn*** and ***sync*** keywords at the right places in Figure 5. Justify your choices.

Write down the recurrences for the work and span of your parallel version of UPDATE and solve them. What is its parallelism? What is its running time on p processors under a greedy scheduler?

3(b) [**10 Points**] Explain how you will improve the parallelism of UPDATE. Write down the pseudocode for your improved algorithm.

Write down the recurrences for the work and span of your new algorithm and solve them. What is its parallelism? What is its running time on p processors under a greedy scheduler?

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

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

Appendix I: Useful Tail Bounds

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

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

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 $\mu = E[X]$. Following bounds hold:

Lower Tail:

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

Upper Tail:

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

Appendix II: The 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 $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$. 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 $T(n) = \Theta(f(n))$.