Data Structures and Algorithms
Homework Assignment 3

Given: January 28, 2018 **Due:** February 5, 2018

Note: The homework is due **electronically on Gradescope and Canvas** on Monday, February 5 by 11:59 pm EST. For late submissions, please refer to the Late Submission Policy on the course webpage. You may use a maximum of 2 late days on this homework.

A. **Gradescope**: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Failing to do so will get you points off, which cannot be argued against after the fact. Gradescope may prompt you with a warning to select your cover page, please ignore this warning.

B. **LaTeX**: You must use the `hw121.cls` LaTeX template provided on the course website, or a harsh penalty will be incurred. Handwritten solutions or solutions not typset in LaTeXwill not be accepted.

C. **Solutions**: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements.

D. **Algorithms**: Whenever you present an algorithm, your answer must include 3 seperate sections:

1. A precise description of your algorithm in English. No pseudocode, no code.
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

E. **Collaboration**: You are allowed to discuss **ideas** for solving homework problems in groups of up to 3 people but *you must write your solutions independently.* Also, you must write on your homework the names of the people with whom you discussed. For a clarification on the collaboration policy, please see Piazza @547

F. **Outside Resources**: Finally, you are not allowed to use *any* material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you're unsure if something violates our policy, please ask.

Note the following: $\lg n$ means $\log_2 n$.

---

[5] 1. Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. Prove that

$$\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$$

[5] 2. For each of the parts below, list the functions in increasing asymptotic order. In some cases functions may be asymptotically equivalent (that is $f(n)$ is $\Theta(g(n))$). In such cases indicate this by writing $f(n) = g(n)$. When one is asymptotically strictly less than the other (that is, $f(n)$ is $O(g(n))$ but $f(n)$ is not $\Theta(g(n))$), express this as $f(n) < g(n)$. For example, given the set:

$$n^2 \qquad n \log n \qquad 3n + n \log n,$$

the first function is $\Theta(n^2)$ and the other two are $\Theta(n \log n)$, and therefore the answer would be

$$n \log n \;=\; 3n + n \log n \;<\; n^2.$$

You are *not* required to show your work; just writing your answer suffices.

(a)	$(3/2)^n$	$3^{(n/2)}$	$2^{(n/3)}$
(b)	$\lg n$	$\ln n$	$\lg(n^2)$
(c)	$n^{\lg 4}$	$2^{\lg n}$	$2^{(2 \lg n)}$
(d)	$\max(50n^2, n^3)$	$50n^2 + n^3$	$\min(50n^2, n^3)$
(e)	$\lceil n^2/20 \rceil$	$\lfloor n^2/20 \rfloor$	$n^2/20$

[10] 3. Consider the following IsPrime algorithm for determining if a given input, n where $n > 1$, is a prime.

```
IsPrime(n)
for (i = 2; i ≤ √n; i = i + 1) do
    if n mod i = 0 then
        return NO
return YES
```

a. Prove that the algorithm correctly determines if the input integer $n > 1$ is a prime or not.

b. Let $T(n)$ be the worst-case running time of the algorithm in terms of the input integer n. Determine $T(n)$.

c. Let b denote the *size* of the input to IsPrime, i.e., b is the number of bits needed to represent n. Express the worst-case running time of the algorithm as a function of b.

d. Is IsPrime a polynomial-time algorithm in the size of the input b?

[15] 4. For each of the code fragments given below, give a bound of the form $\Theta(f(n))$ on its running time on an input of size n. Justify your answer.

(a)

```
for (i = 1; i ≤ n; i = i * 2) do
    for (j = i; j ≤ 8 * i + 1; j = j + 1) do
        print ("CIS 121 is fun!")
```

(b)

```
for (i = 1; i ≤ n; i = i + 1) do
    for (j = 1; j ≤ i; j = j * 2) do
        print ("hi")
```

(c)

```
sum = 0
for (i = 1; i < n; i = i * 2) do
    for (j = i; j < n; j = j + 1) do
        sum = sum + 1
```

[15] 5. During Shirali's bonding activity at the 121 and 160 TA hike, Robin wandered away from the group and stumbled upon a treasure chest in an abandoned cave. Next to the treasure chest, he discovers a cryptic note describing the contents of the chest. Robin learns that there are n doubloons in the chest, some of which are gold and some of which are silver. Additionally, he learns that either the silver doubloons or gold doubloons make up at least 95% of the treasure chest.

Robin wants to know how valuable the treasure chest is, so his objective is to determine whether the treasure chest contains mostly gold doubloons or mostly silver doubloons.

a. Give a deterministic algorithm for the problem such that Robin only removes a small fraction of the doubloons from the treasure chest. In other words, he is only able to examine strictly less than n doubloons. Prove the correctness of your algorithm, and provide its running time. Use the Θ notation to express your answer and provide a justification.

b. Consider the following randomized algorithm for the problem, in which the function `randomDoubloon` chooses a random doubloon from the treasure chest, where each doubloon is equally likely to be chosen. The doubloon is then returned into the treasure chest. The function `isGold` returns 1 if the doubloon is gold and 0 if the doubloon is silver. What is the running time of the algorithm? You may assume that the functions `randomDoubloon` and `isGold` both take $O(1)$ time.

```
i ← randomDoubloon
j ← randomDoubloon
k ← randomDoubloon
if isGold(i) + isGold(j) + isGold(k)≤ 1 then
  return ‘‘mostly silver doubloons’’
else
  return ‘‘mostly gold doubloons’’
```

c. The algorithm can output an incorrect answer. Explain how this can happen.

d. Suppose the treasure chest contains mostly silver doubloons. Give a non-trivial upper-bound on the probability that the randomized algorithm will output "`mostly gold doubloons`." Justify your answer.

[6] 6. Solve the following recurrences using the method of expansion (iteration), giving your answer in Θ notation. For all recurrences, assume that $T(n) = 1$, for all $n \leq 2$ and n is an exact power of 2.

a. $T(n) = 2T(n/2) + 2n$

b. $T(n) = T(\sqrt{n}) + 1$

[9] 7. Solve the following recurrences using the method of expansion (iteration), giving your answer in Θ notation. For all recurrences, assume that $T(n) = 1$ for $n \leq 10$ and n is an exact power of 2.

a. $T(n) = 7T(n - 2)$

b. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

[15] 8. Consider the following basic problem. You're given an array A consisting of n integers $A[1], A[2], \ldots, A[n]$. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ – that is, the sum $A[i] + A[i + 1] + \cdots + A[j]$. (The value of the array entry $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is the output of these values. You should ignore these values when summing the array entries.)

Here is a simple algorithm to solve this problem.

```
for (i=1; i <=  n; i++)
  for (j=i+1; j <= n; j++)
    Add up array entries A[i] through A[j]
    Store the result in B[i,j]
```

a. For the above code fragment give a bound of the form $\Theta(f(n))$ on its running time on an input of size n by giving both a big-oh and big-omega bound. That is, show that the runtime is both $O(f(n))$ and $\Omega(f(n))$ for some function $f(n)$. Justify your answer.

b. Although the algorithm you analyzed in part (a) is the most natural way to solve the problem – after all, it just iterates through the relevant entries of the array B, filling in a value for each – it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.

[20] 9. Sadat and Steven meet up once more for a lucrative real-estate project. They designed their dream city, which only consists of buildings and two-way ski-lifts. We say that two buildings b_1 and b_2 are connected if and only if there is a ski-lift that connects them. To minimize cost, they decided that no more than one ski-lift will be used to make any two buildings connected.

Sadat will own some of the buildings, and Steven will own the rest (if any are still left). However, they want to be able to communicate between each other as much as possible. To do this, they want to assign ownership to the buildings such that the number of ski-lifts between Sadat's buildings and Steven's buildings is maximized.

Sadat does not think that there is a very efficient algorithm for doing this, but Steven came up with the following algorithm:

> 1. Randomly and independently color each building red or blue with probability $\frac{1}{2}$ each.
> 2. Output the grouping defined by the red/blue split of the buildings. Sadat will own the red buildings, Steven will own the blue ones.

Let the random variable X denote the number of ski-lifts between the two groups outputted by the algorithm.

a. Compute $\mathbf{E}[X]$ as a function of the number of pairs of connected buildings, and deduce that $\mathbf{E}[X] \geq \frac{\text{OPT}}{2}$, where OPT is the maximum number of ski-lifts between Sadat's buildings and Steven's buildings.

b. Let p be the probability that the number of ski-lifts between Sadat's buildings and Steven's buildings output by the algorithm has size at least $\frac{49}{100}\text{OPT}$. Using Markov's inequality show that $p \geq \frac{1}{51}$.
(Markov's inequality states that for any non-negative random variable X and any for any $a > 0$, $\Pr(X \geq a) \leq \mathbf{E}[X]/a$)

c. Compute the variance $\text{Var}[X]$.

d. Let p be the probability defined in part (b). Use Chebyshev's inequality together with part (c) to show that $p = 1 - O(|E|^{-1})$, where E is the set consisting of all pairs of

connected buildings.
(Chebyshev's inequality states that for a random variable X and for any $a > 0$,

$$\Pr\left(|X - \mathbf{E}[X]| \geq a\right) \leq \frac{\mathrm{Var}[X]}{a^2}$$

Note how Chebyshev's inequality gives a much sharper bound here than does Markov's inequality.)

e. How would you modify the algorithm so that it *always* finds the number of ski-lifts between Sadat's buildings and Steven's buildings to be at least $\frac{49}{100}\mathrm{OPT}$ but has only expected linear running time?