Assignment: Analysis of Algorithms and Recursion
1 Analysis of algorithms

1. Fill in the following table with one of the three: $O$, $\Omega$, $\Theta$.

Remark: If $f = \Theta(g)$ then $f = O(g)$ and $f = \Omega(g)$ are wrong answers.

<table>
<thead>
<tr>
<th></th>
<th>$f(n)$</th>
<th>$g(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$n$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>b</td>
<td>$n^2$</td>
<td>$n$</td>
</tr>
<tr>
<td>c</td>
<td>$2n$</td>
<td>$5n$</td>
</tr>
<tr>
<td>d</td>
<td>$1000000n$</td>
<td>$(1/100000)n$</td>
</tr>
<tr>
<td>e</td>
<td>$\log_2(n)$</td>
<td>$\log_2(n)$</td>
</tr>
<tr>
<td>f</td>
<td>$\log_2(n)$</td>
<td>$\log_{10}(n)$</td>
</tr>
<tr>
<td>g</td>
<td>$n \log_2(n)$</td>
<td>$n/ \log_2(n)$</td>
</tr>
<tr>
<td>h</td>
<td>$2^n$</td>
<td>$n^{100}$</td>
</tr>
<tr>
<td>i</td>
<td>$2^n$</td>
<td>$3^n$</td>
</tr>
<tr>
<td>j</td>
<td>$2^n$</td>
<td>$n!$</td>
</tr>
</tbody>
</table>

2. Which of the following 10 functions are $O(n)$? Which are $\Omega(n)$? Which are $\Theta(n)$?

Remark: If $f = \Theta(n)$ then $f = O(n)$ and $f = \Omega(n)$ are wrong answers.

<table>
<thead>
<tr>
<th></th>
<th>$f(n)$</th>
<th>$g(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$2^n$</td>
<td>[2^n]</td>
</tr>
<tr>
<td>b</td>
<td>$n^2$</td>
<td>[2^n]</td>
</tr>
<tr>
<td>c</td>
<td>$2n$</td>
<td>[2^n]</td>
</tr>
<tr>
<td>d</td>
<td>$n/ \log_2(n)$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>e</td>
<td>$\log_2(n)$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>f</td>
<td>$100 \log_2(n) \log_2(n)$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>g</td>
<td>$n \log_2(n)$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>h</td>
<td>$10^{100}n/100^{100}$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>i</td>
<td>$n^{\pi}$</td>
<td>[\log_2(n)]</td>
</tr>
<tr>
<td>j</td>
<td>$n!$</td>
<td>[\log_2(n)]</td>
</tr>
</tbody>
</table>

3. Match the following 8 functions as 4 pairs: if $f(n)$ is paired with $g(n)$ then $f(n) = \Theta(g(n))$.

Justify your pairings.

$2^{n+1} : n : \log_2(n^2) : n^2 : 2^n : \log_2(n) : 100n^2 - 500n : \log_2(2^n)$
4. Let $P$ be a problem whose input is an array of size $n$ for $n \geq 1$. Order the following ten algorithms from the most efficient to the least efficient. Justify your answer.

- Algorithm $A$ solves $P$ with complexity $\Theta(n)$.
- Algorithm $B$ solves $P$ with complexity $\Theta(\log(n))$.
- Algorithm $C$ solves $P$ with complexity $\Theta(n \log(n))$.
- Algorithm $D$ solves $P$ with complexity $\Theta(n^2)$.
- Algorithm $E$ solves $P$ with complexity $\Theta(1)$.
- Algorithm $F$ solves $P$ with complexity $\Theta(n!)$.
- Algorithm $G$ solves $P$ with complexity $\Theta(n^n)$.
- Algorithm $H$ solves $P$ with complexity $\Theta(n^{100})$.
- Algorithm $I$ solves $P$ with complexity $\Theta(\sqrt{n})$.
- Algorithm $J$ solves $P$ with complexity $\Theta(2^n)$.

5. For each of the following four parts, give an example of a function that satisfies the criteria or state that none exist. Justify your answers.

(a) A function that is $O(n/2)$ and also $\Omega(2^n)$.
(b) A function that is both $\Omega(10^n)$ and $O(n^2/100)$.
(c) A function that is $O(5n)$ but not $\Theta(n/3)$.
(d) A function that is $\Omega(2^n)$ but not $\Theta(2^n)$.

6. A problem $P$ has an upper bound complexity $O(n^2)$ and a lower bound complexity $\Omega(n)$. Justify your answers to the following three questions.

(a) Could someone design an algorithm that solves the problem whose complexity is $n^3$?
(b) Could someone design an algorithm that solves the problem whose complexity is $0.5n$?
(c) Could someone design an algorithm that solves the problem whose complexity is $100\log(n)$?

7. Express the value of $c$ when each of the following procedures terminates with the $\Theta$-notation. Try to find the exact value of $c$ when each of the following procedures terminates. Justify your answers.

(a) $f(n)$ (* $n = k^2$ is a positive square integer *)

$$c = 0$$

for $i = 1$ to $n$ do
    if $i$ is a square number
    then $c := c + 1$

(b) $f(n)$ (* $n > 1000$ is a power of 2 *)

$$c = 0$$

while $n > 512$ do
    $n := n/2$
    $c := c + 1$

8. Consider the following procedure:

$f(x, y)$ (* a positive multiple of 3 integer $x$ and a positive integer $y$ *)

$$c = 0$$

for $i = 1$ to $x/3$ do
    for $j = 1$ to $6y^2$ do
        then $c := c + 1$

(a) As a function of $x$ and $y$, what is the exact value of $c$ when the program terminates?
(b) Define $x$ and $y$ as functions of $n$ such that $c = \Theta(n^3)$ when the program terminates.
2 Recursion

1. For each one of the following three closed-form expressions, define a recursive formula with an initial value such that the solution to the recursive formula is the closed-form expression. That is, for each expression, define \( T(n) \) as a function of \( T(n - 1) \) and define \( T(1) \).

   Justify your answers.

   (a) \( T(n) = 2n \) for an integer \( n \geq 1 \).
   (b) \( T(n) = 2^n \) for an integer \( n \geq 1 \).
   (c) \( T(n) = n! \) for an integer \( n \geq 1 \).

2. Match each one of the following 5 recursive formulas with one of the possible 5 solutions:

   \(-n \ ; \ 2n \ ; \ n^2 \ ; \ 2^n \ ; \ n!\)

   Justify your answers with few sentences.

   (a) \( T(0) = 1 \ ; \ T(n) = 2T(n - 1) \) for \( n > 0 \)
   (b) \( T(0) = 0 \ ; \ T(n) = T(n - 1) - 1 \) for \( n > 0 \)
   (c) \( T(0) = 0 \ ; \ T(n) = T(n - 1) + (2n - 1) \) for \( n > 0 \)
   (d) \( T(0) = 1 \ ; \ T(n) = nT(n - 1) \) for \( n > 0 \)
   (e) \( T(0) = 0 \ ; \ T(n) = T(n - 1) + 2 \) for \( n > 0 \)

3. Solve the following three recurrences and prove that your solutions are correct.

   (a)
   \[
   T(n) = \begin{cases} 
   2 & \text{for } n = 1 \\
   T(n - 1) + 7 & \text{for } n \geq 2 
   \end{cases}
   
   (b)
   \[
   T(n) = \begin{cases} 
   3 & \text{for } n = 1 \\
   2T(n - 1) & \text{for } n \geq 2 
   \end{cases}
   
   (c)
   \[
   T(n) = \begin{cases} 
   2 & \text{for } n = 1 \\
   (n + 1)T(n - 1) & \text{for } n \geq 2 
   \end{cases}
   
4. In the following two parts, the top-down evaluation and the bottom-up evaluation are not considered as proofs.

   (a) Prove that for \( n \geq 1 \), the solution to the following recurrence is \( M(n) = 3(2^n - 1) \).

   \[
   M(n) = \begin{cases} 
   3 & \text{for } n = 1 \\
   2M(n - 1) + 3 & \text{for } n \geq 2 
   \end{cases}
   
   (b) Prove that for \( n \geq 1 \), the solution to the following recurrence is \( M(n) = \frac{3^n + 1}{2} \).

   \[
   M(n) = \begin{cases} 
   2 & \text{for } n = 1 \\
   3M(n - 1) - 1 & \text{for } n \geq 2 
   \end{cases}
   
4
5. Solve the following two recurrences by expressing $T(n)$ as a function of $n$ for all integers $n \geq 1$. Prove that your solutions are correct.

(a) 
$$T(n) = \begin{cases} 
1 & \text{for } n = 0 \\
2T(n-1) + 1 & \text{for } n \geq 1 
\end{cases}$$

(b) 
$$T(n) = \begin{cases} 
1 & \text{for } n = 1 \\
T(n-1) + (2n - 1) & \text{for } n \geq 2 
\end{cases}$$

6. Solve the following double recursive formulas. Find the closed-forms for both $T(n)$ and $S(n)$ as a function of $n$. Assume that $n$ is a positive integer.

Explain how you found your solution and prove your answer.

**Hint:** One way to solve these double recursion is to express $T(n)$ as a function of $T(n-1)$ and $S(n)$ as a function of $S(n-1)$ establishing two independent recursive formulas. Then “guess” the closed-forms (possibly using a bottom-up evaluation) and prove their correctness with induction.

$$T(n) = \begin{cases} 
2 & \text{for } n = 1 \\
S(n-1) + 1 & \text{for } n > 1 
\end{cases} \\
S(n) = \begin{cases} 
3 & \text{for } n = 1 \\
T(n) + 1 & \text{for } n > 1 
\end{cases}$$

7. Solve the following recurrence and prove that your solution is correct.

$$T_n = \begin{cases} 
1 & \text{for } n = 1 \\
2 & \text{for } n = 2 \\
T_{n-2} + n & \text{for } n \geq 2 
\end{cases}$$

**Hint:** There are different solutions for even $n$ and odd $n$ that can be combined to one solution using the floor and ceiling operations.

8. Solve the following recurrence and prove that your solution is correct.

$$P_n = \begin{cases} 
1 & \text{for } n = 0 \\
2 & \text{for } n = 1 \\
5P_{n-1} - 6P_{n-2} & \text{for } n \geq 2 
\end{cases}$$

**Guide:** Do the bottom-up evaluation, guess the solution, and prove by induction the correctness of your guess.


3 Fibonacci numbers

1. The Fibonacci sequence $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$ is defined as follows

$$F_n = \begin{cases} 
0 & \text{ for } n = 0 \\
1 & \text{ for } n = 1 \\
F_{n-1} + F_{n-2} & \text{ for } n \geq 2 
\end{cases}$$

(a) What is the smallest $n$ for which $F_n > 100$? What is the smallest $n$ for which $F_n > 1000$?
(b) Let $A_n = (F_1 + F_2 + \cdots + F_n)/n$ be the average of the first $n$ Fibonacci numbers. What is the smallest $n$ for which $A_n > 10$?
(c) Find all $n$ for which $F_n = n$. Explain why these are the only cases.
(d) Find all $n$ for which $F_n = n^2$. Explain why these are the only cases.

2. Prove the following identity for $n \geq 2$:

$$F_{n+1} + F_{n-1} = F_{n+2} - F_{n-2}$$

**Hint:** There exists a simple proof without induction that is based on the recursive definition of the Fibonacci numbers.

3. Prove the following identity for $n \geq 1$:

$$F_{2n+1} = F_{n+1}^2 + F_n^2$$

4. Define the following (almost Fibonacci) recurrence

$$G_n = \begin{cases} 
0 & \text{ for } n = 0 \\
1 & \text{ for } n = 1 \\
G_{n-1} + G_{n-2} + 1 & \text{ for } n \geq 2 
\end{cases}$$

(a) Find the values of $G_0, G_1, \ldots, G_{10}$.
(b) Express $G_n$ as a function of Fibonacci numbers.
(c) Prove that your expression for $G_n$ is correct for all $n \geq 0$.

5. A binary string is 1-even if the length of any sequence of consecutive 1’s in the string is even.

For example $(0110)$, $(11011)$, $(00000)$, and $(001110)$ are 1-even binary strings while $(010)$, $(11100)$, and $(1100111)$ are not 1-even binary strings.

(a) List all the 1-even strings of length 2, 3, and 4.
(b) For $k \geq 1$, as a function of $k$, how many 1-even binary strings of length $k$ exist? Justify your answer.

**Hint:** Think of Fibonacci numbers and their recursive definition. Then consider the two cases in which the first bit in the string is 1 or 0.

6. A binary string is 1-lonely if it does not contain consecutive 1s.

For example $(0010)$ and $(01001)$ are 1-lonely binary strings while $(11000)$, $(01110)$, and $(010110)$ are not 1-lonely binary strings.

(a) List all the 1-lonely binary strings of length 2, 3, and 4.
(b) For $k \geq 1$, as a function of $k$, how many 1-lonely binary strings of length $k$ exist? Justify your answer.

**Hint:** Think of Fibonacci numbers and their recursive definition. Then consider the two cases in which the first bit in the string is 1 or 0.
7. For $n \geq 1$, in how many out of the $n!$ permutations $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of the numbers $\{1, 2, \ldots, n\}$ the value of $\pi(i)$ is either $i - 1$, or $i$, or $i + 1$ for all $1 \leq i \leq n$?

**Example:** The permutation $(21354)$ follows the rules while the permutation $(21534)$ does not because $\pi(3) = 5$.

**Hint:** Find the answer for small $n$ by checking all the permutations and then find the recursive formula depending on the possible values for $\pi(n)$.

4 Other problems

1. For $n \geq 2$, let $A = A[1], \ldots, A[n]$ be an array of $n$ positive integers. Let the sum of all the integers in the array be $M = A[1] + \cdots + A[n]$. For $1 \leq i \leq n$, let $S[i]$ be the sum of all the numbers in the array except $A[i]$.


**Example:** Let $A = [16, 2, 128, 64, 1, 8, 32, 4]$. Then $M = 255$ and $S = [239, 253, 127, 191, 254, 247, 223, 251]$. Design a linear time algorithm ($\Theta(n)$) to compute $S[1], \ldots, S[n]$ only with plus operations (you are not allowed to use minus operations).

What is the exact number of plus operations used by your algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.


Describe an efficient algorithm that finds two integers from the array whose sum is even.

What is the complexity of your algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.