Algorithms

Assignment Solutions: Analysis of Algorithms and Recursion
# 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>???</th>
<th>$g(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$n$</td>
<td>$O$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>b</td>
<td>$n^2$</td>
<td>$\Omega$</td>
<td>$n$</td>
</tr>
<tr>
<td>c</td>
<td>$2n$</td>
<td>$\Theta$</td>
<td>$5n$</td>
</tr>
<tr>
<td>d</td>
<td>$1000000n$</td>
<td>$\Theta$</td>
<td>$(1/100000)n$</td>
</tr>
<tr>
<td>e</td>
<td>$\log_2(n)$</td>
<td>$O$</td>
<td>$\log_2(n)$</td>
</tr>
<tr>
<td>f</td>
<td>$\log_2(n)$</td>
<td>$\Theta$</td>
<td>$\log_{10}(n)$</td>
</tr>
<tr>
<td>g</td>
<td>$n \log_2(n)$</td>
<td>$\Omega$</td>
<td>$n/\log_2(n)$</td>
</tr>
<tr>
<td>h</td>
<td>$2^n$</td>
<td>$\Omega$</td>
<td>$n^{100}$</td>
</tr>
<tr>
<td>i</td>
<td>$2^n$</td>
<td>$O$</td>
<td>$3^n$</td>
</tr>
<tr>
<td>j</td>
<td>$2^n$</td>
<td>$O$</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>???</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$2^n$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>b</td>
<td>$n^2$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>c</td>
<td>$2n$</td>
<td>$\Theta$</td>
</tr>
<tr>
<td>d</td>
<td>$n/\log_2(n)$</td>
<td>$O$</td>
</tr>
<tr>
<td>e</td>
<td>$\log_2(n)$</td>
<td>$O$</td>
</tr>
<tr>
<td>f</td>
<td>$100 \log_2(n) \log_2(n)$</td>
<td>$O$</td>
</tr>
<tr>
<td>g</td>
<td>$n \log_2(n)$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>h</td>
<td>$10^{10} n/100^{100}$</td>
<td>$\Theta$</td>
</tr>
<tr>
<td>i</td>
<td>$n^\pi$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>j</td>
<td>$n!$</td>
<td>$\Omega$</td>
</tr>
</tbody>
</table>

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

   $2^{n+1}$ ; $n$ ; $\log_2(n^2)$ ; $n^2$ ; $2^n$ ; $\log_2(n)$ ; $100n^2 - 500n$ ; $\log_2(2^n)$

   **Answer:**

   $\log_2(n^2) = 2 \log_2(n)$ \quad $\Rightarrow$ \quad $\log_2(n^2) = \Theta(\log_2(n))$

   $\log_2(2^n) = \log_2(n) \log_2(2) = n$ \quad $\Rightarrow$ \quad $\log_2(2^n) = \Theta(n)$

   $100n^2 - 500n = \Theta(100n^2)$ \quad $\Rightarrow$ \quad $100n^2 - 500n = \Theta(n^2)$

   $2^{n+1} = 2 \cdot 2^n$ \quad $\Rightarrow$ \quad $2^{n+1} = \Theta(2^n)$

2
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.

- 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 $\Theta(2^n)$.

**Answer:** The following is the hierarchy among the ten functions:

$$1 = o(\log(n)) = o(\sqrt{n}) = o(n) = o(n \log(n)) = o(n^2) = O(n^{100}) = o(2^n) = o(n!) = o(n^n)$$

Therefore, using $X < Y$ to indicate that algorithm $X$ is more efficient than algorithm $Y$, it follows that the order among the ten algorithms from the most efficient one to the least efficient one is as follows:

$$E < B < I < A < C < D < H < J < F < G$$

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

(a) A function that is $O(n/2)$ and also $\Omega(2n)$.

**Answer:** Both $n/2 = \Theta(n)$ and $2n = \Theta(n)$. Therefore, $n = O(n/2)$ and $n = \Omega(2n)$.

(b) A function that is both $\Omega(10n)$ and $O(n^2/100)$.

**Answer:** Observe that $10n = o(n^2/100)$, $10n = \Theta(n)$, and $n^2/100 = \Theta(n^2)$. Therefore, any function that is $\Omega(n)$ and $O(n^2)$ is a correct answer. In particular, both $n$ and $n^2$ are correct answers. But also $n \log_2(n)$, $\sqrt{n}$, and $n/\log_2(n)$ are correct answers. In fact, more accurately, the latter three functions are $\omega(2n)$ and $o(n/2)$.

(c) A function that is $O(5n)$ but not $\Theta(n/3)$.

**Answer:** Such a function must be $O(n)$ but not $\Theta(n)$ and as a result it must be $o(n)$. The functions $\sqrt{n}$ and $\log(n)$ are two examples.

(d) A function that is $\Omega(2^n)$ but not $\Theta(2^n)$.

**Answer:** Such a function must be $\omega(n)$. The functions $3^n$, $n!$, and $n^n$ are three examples.
6. A problem \( P \) has an **upper bound** complexity \( O(n^2) \) and a **lower bound** complexity \( \Omega(n) \).

(a) Could someone design an algorithm that solves the problem whose complexity is \( n^3 \)?

**Answer:** Yes because \( n^3 = \Omega(n) \) and it is always possible to design inefficient algorithms.

(b) Could someone design an algorithm that solves the problem whose complexity is \( 0.5n \)?

**Answer:** Yes because \( 0.5n = \Theta(n) \) and therefore \( 0.5n = O(n^2) \) and \( 0.5n = \Omega(n) \).

(c) Could someone design an algorithm that solves the problem whose complexity is \( 100\log(n) \)?

**Answer:** No because \( 100\log(n) = o(n) \) and as such \( 100\log(n) \neq \Omega(n) \).

7. Express the value of \( c \) when each of the following procedures terminates with the \( \Theta \)-notation. Try to find then exact value of \( c \) when each of the following procedures terminates.

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

\[
c = 0
\]
\[
\text{for } i = 1 \text{ to } n \text{ do}
\]
\[
\text{if } i \text{ is a square number}
\]
\[
\text{then } c := c + 1
\]

**Answer:** \( c = \sqrt{n} = \Theta(n^{1/2}) \).

**Explanation:** \( c \) is incremented only when \( i \) is a square number. That is, for \( n = k^2 \), \( c \) is incremented when \( i = 1, 4, 9, \ldots, (k - 1)^2, k^2 \). The final value of \( c \) is \( k = \sqrt{n} \) because there are exactly \( k \) square integers between 1 and \( k^2 \).

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

\[
c = 0
\]
\[
\text{while } n > 512 \text{ do}
\]
\[
n := n/2
\]
\[
c := c + 1
\]

**Answer:** \( \log_2 n - 9 = \Theta(\log n) \).

**Explanation:** Assume \( n = 2^k \). Since \( n > 1000 \) it follows that \( k \geq 10 \). Each time \( c \) is incremented by 1, \( n \) is divided by 2. Therefore, the values of \( n \) are: \( 2^k, 2^{k-1}, \ldots, 2^9 \). Once \( n = 2^9 \) the while loop stops because \( 512 = 2^9 \). Therefore, \( c \) is incremented \( k - 9 \) times. The answer is \( \log_2 n - 9 \) because \( k = \log_2 n \).

8. Consider the following procedure:

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

\[
c = 0
\]
\[
\text{for } i = 1 \text{ to } x/3 \text{ do}
\]
\[
\text{for } j = 1 \text{ to } 6y^2 \text{ do}
\]
\[
\text{then } c := c + 1
\]

(a) As a function of \( x \) and \( y \), what is the **exact** value of \( c \) when the program terminates?

**Answer:** The variable \( c \) is incremented \( (x/3)(6y^2) \) times. Therefore, when the program terminated \( c = 2xy^2 \).

(b) Define \( x \) and \( y \) as functions of \( n \) such that \( c = \Theta(n^3) \) when the program terminates.

**Answer:** When \( n = 2x \) and therefore \( x = n/2 \) and \( y = n \), by part (a) the program terminates with \( c = 2xy^2 = 2(n/2)(n^2) = n^3 \). Trivially, \( n^3 = \Theta(n^3) \). This can be generalized for any \( x = \Theta(n) \) and \( y = \Theta(n) \).

There are infinitely many other answers. For example, \( x = \Theta(n^2) \) and \( y = \Theta(\sqrt{n}) \).
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 \( T(1) \).

(a) \( T(n) = 2n \) for an integer \( n \geq 1 \).

**Answer:**
\[
T(n) = \begin{cases} 
  2 & \text{for } n = 1 \\
  T(n - 1) + 2 & \text{for } n \geq 1 
\end{cases}
\]

**Proof by induction** \( T(n) = 2n \) for \( n \geq 1 \):
- **Induction base.** \( T(1) = 2 \cdot 1 = 2 \) for \( n = 1 \).
- **Induction hypothesis.** Assume that \( T(n - 1) = 2(n - 1) \) for \( n > 1 \).
- **Inductive step.** Prove that \( T(n) = 2n \) for \( n > 1 \):
\[
T(n) = T(n - 1) + 2 \\
= 2(n - 1) + 2 \\
= 2n - 2 + 2 \\
= 2n
\]

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

**Answer:**
\[
T(n) = \begin{cases} 
  2 & \text{for } n = 1 \\
  2T(n - 1) & \text{for } n > 1 
\end{cases}
\]

**Proof by induction that** \( T(n) = 2^n \) for \( n \geq 1 \):
- **Induction base.** \( T(1) = 2^1 = 2 \) for \( n = 1 \).
- **Induction hypothesis.** Assume that \( T(n - 1) = 2^{n-1} \) for \( n > 1 \).
- **Inductive step.** Prove that \( T(n) = 2^n \) for \( n > 1 \):
\[
T(n) = 2T(n - 1) \\
= 2 \cdot 2^{n-1} \\
= 2^n
\]

(c) \( T(n) = n! \) for an integer \( n \geq 1 \).

**Answer:**
\[
T(n) = \begin{cases} 
  1 & \text{for } n = 1 \\
  nT(n - 1) & \text{for } n > 1 
\end{cases}
\]

**Proof by induction that** \( T(n) = n! \) for \( n \geq 1 \):
- **Induction base.** \( T(1) = 1! = 1 \) for \( n = 1 \).
- **Induction hypothesis.** Assume that \( T(n - 1) = (n - 1)! \) for \( n > 1 \).
- **Inductive step.** Prove that \( T(n) = n! \) for \( n > 1 \):
\[
T(n) = nT(n - 1) \\
= n \cdot (n - 1)! \\
= n!
\]
2. Match each one of the following 5 recursive formulas with one of the possible 5 solutions:
\[-n \quad 2n \quad n^2 \quad 2^n \quad n!\]

(a) \(T(0) = 1 \quad \text{; } T(n) = 2T(n-1) \) for \( n > 0 \)

Solution: \( T(n) = 2^n \).

Intuition: In every recursive step the value is doubled by two.

Proof:
- \textit{Induction base.} \( T(0) = 1 = 2^0 \) for \( n = 0 \).
- \textit{Induction hypothesis.} Assume that \( T(n-1) = 2^{n-1} \) for \( n > 0 \).
- \textit{Inductive step.} Prove that \( T(n) = 2^n \) for \( n > 0 \):
  \[
  T(n) = 2T(n-1) = 2 \cdot 2^{n-1} = 2^n
  \]

(b) \(T(0) = 0 \quad \text{; } T(n) = T(n-1) - 1 \) for \( n > 0 \)

Solution: \( T(n) = -n \).

Intuition: In every recursive step the value is decremented by one.

Proof:
- \textit{Induction base.} \( T(0) = 0 = -0 \) for \( n = 0 \).
- \textit{Induction hypothesis.} Assume that \( T(n-1) = -(n-1) \) for \( n > 0 \).
- \textit{Inductive step.} Prove that \( T(n) = -n \) for \( n > 0 \):
  \[
  T(n) = T(n-1) - 1 = -(n-1) - 1 = -n + 1 - 1 = -n
  \]

(c) \(T(0) = 0 \quad \text{; } T(n) = T(n-1) + (2n-1) \) for \( n > 0 \)

Solution: \( T(n) = n^2 \).

Intuition: In the \( n \)th recursive step the value is incremented by \( 2n-1 \) which is the difference between \( n^2 \) and \( (n - 1)^2 \).

Proof:
- \textit{Induction base.} \( T(0) = 0 = 0^2 \) for \( n = 0 \).
- \textit{Induction hypothesis.} Assume that \( T(n-1) = (n-1)^2 \) for \( n > 0 \).
- \textit{Inductive step.} Prove that \( T(n) = n^2 \) for \( n > 0 \):
  \[
  T(n) = T(n-1) + (2n-1) = (n-1)^2 + (2n-1) = n^2 - 2n + 1 + 2n - 1 = n^2
  \]

(d) \(T(0) = 1 \quad \text{; } T(n) = nT(n-1) \) for \( n > 0 \)

Solution: \( T(n) = n! \).

Intuition: In the \( n \)th recursive step the value is multiplied by \( n \).

Proof:
- \textit{Induction base.} \( T(0) = 1 = 0! \) for \( n = 0 \).
- \textit{Induction hypothesis.} Assume that \( T(n-1) = (n-1)! \) for \( n > 0 \).
- \textit{Inductive step.} Prove that \( T(n) = n! \) for \( n > 0 \):
  \[
  T(n) = n \cdot T(n-1) = n \cdot (n-1)! = n!
  \]

(e) \(T(0) = 0 \quad \text{; } T(n) = T(n-1) + 2 \) for \( n > 0 \)

Solution: \( T(n) = 2n \).

Intuition: In every recursive step the value is incremented by 2.

Proof:
- \textit{Induction base.} \( T(0) = 0 = 2 \cdot 0 \) for \( n = 0 \).
- \textit{Induction hypothesis.} Assume that \( T(n-1) = 2(n-1) \) for \( n > 0 \).
- \textit{Inductive step.} Prove that \( T(n) = 2n \) for \( n > 0 \):
  \[
  T(n) = T(n-1) + 2 = 2(n-1) + 2 = 2n - 2 + 2 = 2n
  \]
3. Solve the following three recurrences and prove that your solutions are correct.

**Recurrence (a):**

\[
T(n) = \begin{cases} 
2 & \text{for } n = 1 \\
T(n - 1) + 7 & \text{for } n \geq 2 
\end{cases}
\]

**Bottom-Up evaluation:**

\[
\begin{align*}
T(1) &= 2 = 7 \cdot 1 - 5 \\
T(2) &= T(1) + 7 = 9 = 7 \cdot 2 - 5 \\
T(3) &= T(2) + 7 = 16 = 7 \cdot 3 - 5 \\
T(4) &= T(3) + 7 = 23 = 7 \cdot 4 - 5 \\
&\vdots \\
T(n) &= \quad = 7 \cdot n - 5
\end{align*}
\]

**Top-Down evaluation:**

\[
\begin{align*}
T(n) &= T(n - 1) + 7 \\
&= (T(n - 2) + 7) + 1 \cdot 7 = T(n - 2) + 2 \cdot 7 \\
&= (T(n - 3) + 7) + 2 \cdot 7 = T(n - 3) + 3 \cdot 7 \\
&= (T(n - 4) + 7) + 3 \cdot 7 = T(n - 4) + 4 \cdot 7 \\
&\vdots \\
&\vdots \\
&= T(n - (n - 1)) + (n - 1)7 \\
&= T(1) + 7n - 7 \\
&= 2 + 7n - 7 \\
&= 7n - 5
\end{align*}
\]

**Solution:** \( T(n) = 7n - 5 \) for \( n \geq 1 \).

**Proof by induction:**

- **Induction base.** \( T(1) = 2 = 7 \cdot 1 - 5 \).
- **Induction hypothesis.** \( T(n - 1) = 7(n - 1) - 5 = 7n - 12 \) for \( n > 1 \).
- **Inductive step.** For \( n > 1 \),

\[
\begin{align*}
T(n) &= T(n - 1) + 7 \quad (* \text{definition of } T(n) *) \\
&= (7n - 12) + 7 \quad (* \text{induction hypothesis } *) \\
&= 7n - 5 \quad (* \text{rearranging terms } *)
\end{align*}
\]
Recurrence (b):

\[ T(n) = \begin{cases} 
3 & \text{for } n = 1 \\
2T(n-1) & \text{for } n \geq 2 
\end{cases} \]

Bottom-Up evaluation:

\[
\begin{align*}
T(1) &= 3 = 3 \cdot 1 = 3 \cdot 2^0 \\
T(2) &= 2T(1) = 6 = 3 \cdot 2 = 3 \cdot 2^1 \\
T(3) &= 2T(2) = 12 = 3 \cdot 4 = 3 \cdot 2^2 \\
T(4) &= 2T(3) = 24 = 3 \cdot 8 = 3 \cdot 2^3 \\
& \vdots \\
T(n) &= 3 \cdot 2^{n-1}
\end{align*}
\]

Top-Down evaluation:

\[
\begin{align*}
T(n) &= 2 \cdot T(n-1) = 2^1 \cdot T(n-1) \\
&= 2^1 \cdot (2 \cdot T(n-2)) = 2^2 \cdot T(n-2) \\
&= 2^2 \cdot (2 \cdot T(n-3)) = 2^3 \cdot T(n-3) \\
&= 2^3 \cdot (2 \cdot T(n-2)) = 2^4 \cdot T(n-4) \\
& \vdots \\
&= 2^i \cdot T(n-i) \\
& \vdots \\
&= 2^{n-1} \cdot T(n-(n-1)) \\
&= 2^{n-1} \cdot T(1) \\
&= 2^{n-1} \cdot 3 \\
&= 3 \cdot 2^{n-1}
\end{align*}
\]

Solution: \( T(n) = 3 \cdot 2^{n-1} \) for \( n \geq 1 \).

Proof by induction:

- **Induction base.** \( T(1) = 3 = 3 \cdot 1 = 3 \cdot 2^0 = 3 \cdot 2^{1-1} \).
- **Induction hypothesis.** \( T(n-1) = 3 \cdot 2^{n-2} \) for \( n > 1 \).
- **Inductive step.** For \( n > 1 \),

\[
\begin{align*}
T(n) &= 2 \cdot T(n-1) \quad (* \text{definition of } T(n) *) \\
&= 2 \cdot (3 \cdot 2^{n-2}) \quad (* \text{induction hypothesis } *) \\
&= 3 \cdot (2 \cdot 2^{n-2}) \quad (* \text{rearranging terms } *) \\
&= 3 \cdot 2^{n-1} \quad (* \text{definition of the power function } *)
\end{align*}
\]
Recurrence (c):

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

Bottom-Up evaluation:

\[
\begin{align*}
T(1) &= 2 = 2 \\
T(2) &= 3T(1) = 6 = 3 \cdot 2 \cdot 1 \\
T(3) &= 4T(2) = 24 = 4 \cdot 3 \cdot 2 \cdot 1 \\
T(4) &= 5T(3) = 120 = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \\
& \vdots \\
T(n) &= (n + 1) \cdot (n - 1) \cdots 2 \cdot 1 \\
T(n) &= (n + 1)!
\end{align*}
\]

Top-Down evaluation:

\[
\begin{align*}
T(n) &= (n + 1) \cdot T(n - 1) \\
&= (n + 1) \cdot (n \cdot T(n - 2)) \\
&= (n + 1) \cdot n \cdot (n - 1) \cdot T(n - 3) \\
&= (n + 1) \cdot n \cdot (n - 1) \cdot (n - 2) \cdot T(n - 4) \\
& \vdots \\
&= (n + 1) \cdot n \cdot (n - 1) \cdots 3 \cdot T(1) \\
&= (n + 1) \cdot n \cdot (n - 1) \cdots 3 \cdot 2 \\
&= (n + 1) \cdot n \cdot (n - 1) \cdots 3 \cdot 2 \cdot 1 \\
&= (n + 1)!
\end{align*}
\]

Solution: \( T(n) = (n + 1)! \) for \( n \geq 1 \).

Proof by induction:

- Induction base. \( T(1) = 2 = 2 \cdot 1 = 2! = (1 + 1)! \).
- Induction hypothesis. \( T(n - 1) = n! \) for \( n > 1 \).
- Inductive step. For \( n \geq 1 \),

\[
\begin{align*}
T(n) &= (n + 1)T(n - 1) \quad (\ast \text{ definition of } T(n) \ast) \\
&= (n + 1)n! \quad (\ast \text{ induction hypothesis } \ast) \\
&= (n + 1)! \quad (\ast \text{ definition of factorial } \ast)
\end{align*}
\]
4. (a) Prove that for $n \geq 1$, the solution to the following recurrence is $M(n) = 3(2^n - 1)$.

\[ M(n) = \begin{cases} 
3 & \text{for } n = 1 \\
2M(n-1) + 3 & \text{for } n \geq 2 
\end{cases} \]

**Proof by induction:**
- **Induction base.** $M(1) = 3(2^1 - 1) = 3(2 - 1) = 3 \cdot 1 = 3$ for $n = 1$.
- **Induction hypothesis.** Assume that $M(n - 1) = 3(2^{n-1} - 1)$ for $n > 1$.
- **Inductive step.** Prove that $M(n) = 3(2^n - 1)$ for $n > 1$.

\[
M(n) = 2M(n-1) + 3 \\
= 2(3(2^{n-1} - 1)) + 3 \\
= 3(2^{n-1} - 1) + 3 \\
= 3(2^{n-1} - 1 + 1) \\
= 3(2^n - 2 + 1) \\
= 3(2^n - 1)
\]

(b) Prove that for $n \geq 1$, the solution to the following recurrence is $M(n) = \frac{3^{n+1}}{2}$.

\[ M(n) = \begin{cases} 
2 & \text{for } n = 1 \\
3M(n-1) - 1 & \text{for } n \geq 2 
\end{cases} \]

**Proof by induction:**
- **Induction base.** $M(1) = \frac{3^1 + 1}{2} = \frac{4}{2} = 2$ for $n = 1$.
- **Induction hypothesis.** Assume that $M(n - 1) = \frac{3^{n-1} + 1}{2}$ for $n > 1$.
- **Inductive step.** Prove that $M(n) = \frac{3^{n+1}}{2}$ for $n > 1$.

\[
M(n) = 3M(n-1) - 1 \\
= 3 \cdot \frac{3^{n-1} + 1}{2} - 1 \\
= \frac{3(3^{n-1} + 1)}{2} - 2 \\
= \frac{3^n + 3 - 2}{2} \\
= \frac{3^n + 1}{2}
\]
5. Solve the following two recurrences by expressing $T(n)$ as a function of $n$ for all integers $n \geq 1$.

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

Small values of $n$:
- $n = 0$: $T(0) = 1 = 2^1 - 1$.
- $n = 1$: $T(1) = 2T(0) + 1 = 2 \cdot 1 + 1 = 3 = 2^2 - 1$.
- $n = 2$: $T(2) = 2T(1) + 1 = 2 \cdot 3 + 1 = 7 = 2^3 - 1$.
- $n = 3$: $T(3) = 2T(2) + 1 = 2 \cdot 7 + 1 = 15 = 2^4 - 1$.
- $n = 4$: $T(4) = 2T(3) + 1 = 2 \cdot 15 + 1 = 31 = 2^5 - 1$.

Answer: $T(n) = 2^{n+1} - 1$.

Proof by induction:
- Induction base. $T(0) = 2^{0+1} - 1 = 1$ for $n = 0$.
- Induction hypothesis. Assume that $T(n-1) = 2^n - 1$ for $n > 0$.
- Inductive step. Prove that $T(n) = 2^{n+1} - 1$ for $n > 0$:
\[
T(n) = 2T(n-1) + 1 \\
= 2(2^n - 1) + 1 \\
= 2^{n+1} - 2 + 1 \\
= 2^{n+1} - 1
\]

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

Small values of $n$:
- $n = 1$: $T(1) = 1 = 1^2$.
- $n = 2$: $T(2) = T(1) + (2 \cdot 2 - 1) = 1 + 3 = 4 = 2^2$.
- $n = 3$: $T(3) = T(2) + (2 \cdot 3 - 1) = 4 + 5 = 9 = 3^2$.
- $n = 4$: $T(4) = T(3) + (2 \cdot 4 - 1) = 9 + 7 = 16 = 4^2$.

Answer: $T(n) = n^2$.

Proof by induction:
- Induction base. $T(1) = 1^2 = 1$ for $n = 1$.
- Induction hypothesis. Assume that $T(n-1) = (n-1)^2$ for $n > 0$.
- Inductive step. Prove that $T(n) = n^2$ for $n > 0$:
\[
T(n) = T(n-1) + (2n-1) \\
= (n-1)^2 + (2n-1) \\
= n^2 - 2n + 1 - 2n - 1 \\
= n^2
\]
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.

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

**Answer:** \( T(n) = 2n \) and \( S(n) = 2n + 1 \)

**Proof:** First transform the given double recursive formulas in which \( T(n) \) is defined by \( S(n - 1) \) and \( S(n) \) is defined by \( T(n) \) to a standard single recursive formula in which \( T(n) \) is defined by \( T(n - 1) \) for \( n > 1 \).

\[
T(n) = S(n - 1) + 1 = (T(n - 1) + 1) + 1 = T(n - 1) + 2
\]

Next compute \( T(n) \) with a bottom-up evaluation.

\[
\begin{align*}
T(1) &= 2 = 2 \cdot 1 \\
T(2) &= T(1) + 2 = 4 = 2 \cdot 2 \\
T(3) &= T(2) + 2 = 6 = 2 \cdot 3 \\
T(4) &= T(3) + 2 = 8 = 2 \cdot 4 \\
&\vdots \\
T(n) &= = 2 \cdot n
\end{align*}
\]

Next prove by induction that \( T(n) = 2n \) for \( n \geq 1 \).

- **Induction base.** \( T(1) = 2 = 2 \cdot 1 \).
- **Induction hypothesis.** \( T(n - 1) = 2(n - 1) \) for \( n \geq 2 \).
- **Inductive step.** For \( n \geq 2 \),

\[
\begin{align*}
T(n) &= T(n - 1) + 2 \quad (\text{* definition of } T(n) \text{*}) \\
&= 2(n - 1) + 2 \quad (\text{* induction hypothesis *}) \\
&= 2n \quad (\text{* algebra *})
\end{align*}
\]

Finally, express \( S(n) \) as a function of \( n \) given the closed form for \( T(n) \).

\[
S(n) = T(n) + 1 = 2n + 1
\]

**Remark:** In a similar proof, it can be shown that \( S(n) = S(n) + 2 \) and since \( S(1) = 3 \) it follows that \( S(n) = 2n + 1 \) which in turn implies that \( T(n) = 2n \).
7. Solve the following recurrence and prove that your solution is correct.

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

Small values of \( n \):

- \( T_1 = 1 \)
- \( T_2 = 2 \)
- \( T_3 = T_1 + 3 = 1 + 3 = 4 \)
- \( T_4 = T_2 + 4 = 2 + 4 = 6 \)
- \( T_5 = T_3 + 5 = 4 + 5 = 9 \)
- \( T_6 = T_4 + 6 = 6 + 6 = 12 \)
- \( T_7 = T_5 + 7 = 9 + 7 = 16 \)
- \( T_8 = T_6 + 8 = 12 + 8 = 20 \)

Solution: \( T_n = \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1) \) for \( n \geq 1 \).

Guesses: For an odd \( n \) the implied sub-sequence 1, 4, 9, 16, \ldots is a sequence of squares. Hence, for \( k \geq 1 \), if \( n = 2k - 1 \) then \( T_n = k^2 \). For an even \( n \) the implied sub-sequence 2, 6, 12, 20, \ldots is a sequence of products of consecutive integers. Hence, for \( k \geq 1 \), if \( n = 2k \) then \( T_n = k(k + 1) \).

Claim: If \( T_n = \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1) \) for \( n \geq 1 \) then

\[ T_n = \begin{cases} 
k^2 & \text{for } n = 2k - 1 \text{ and } k \geq 1 \\
k(k + 1) & \text{for } n = 2k \text{ and } k \geq 1 
\end{cases} \]

Proof: Assume first that \( n = 2k - 1 \). Then \( \lceil n/2 \rceil = k \) and \( \lfloor n/2 \rfloor + 1 = (k - 1) + 1 = k \) and therefore \( \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1) = k \cdot k = k^2 \). Next assume that \( n = 2k \). Then \( \lceil n/2 \rceil = k \) and \( \lfloor n/2 \rfloor + 1 = k + 1 \) and therefore \( \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1) = k(k + 1) \).

Proof by induction on \( k \) that \( T_{2k-1} = k^2 \) for \( k \geq 1 \):

- Induction base \( k = 1 \). \( T_{2-1} = T_1 = 1 = 1^2 \).
- Induction hypothesis. \( T_{2(k-1)-1} = (k - 1)^2 \).
- Inductive step. For \( k \geq 2 \),

\[
T_{2k-1} = T_{2(k-1)-1} + (2k - 1) \quad (* \text{ definition of } T_n \ast) \\
= (k - 1)^2 + (2k - 1) \quad (* \text{ induction hypothesis } \ast) \\
= (k^2 - 2k + 1) + (2k - 1) \quad (* \text{ algebra } \ast) \\
= k^2 \quad (* \text{ algebra } \ast)
\]

Proof by induction on \( k \) that \( T_{2k} = k(k + 1) \) for \( k \geq 1 \):

- Induction base \( k = 1 \). \( T_{2-1} = T_2 = 2 = 1 \cdot 2 \).
- Induction hypothesis. \( T_{2(k-1)} = (k - 1)k \).
- Inductive step. For \( k \geq 2 \),

\[
T_{2k} = T_{2k-2} + 2k \quad (* \text{ definition of } T_n \ast) \\
= (k - 1)k + 2k \quad (* \text{ induction hypothesis } \ast) \\
= k((k - 1) + 2) \quad (* \text{ algebra } \ast) \\
= k(k + 1) \quad (* \text{ algebra } \ast)
\]
8. Solve the following recurrence and prove that your solution is correct.

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

Small values of \( n \):
\[
\begin{align*}
P_0 &= 1 \\
P_1 &= 2 \\
P_2 &= 5 \cdot 2 - 6 \cdot 1 = 10 - 6 = 4 \\
P_3 &= 5 \cdot 4 - 6 \cdot 2 = 20 - 12 = 8 \\
P_4 &= 5 \cdot 8 - 6 \cdot 4 = 40 - 24 = 16 \\
P_5 &= 5 \cdot 16 - 6 \cdot 8 = 80 - 48 = 32
\end{align*}
\]

Solution: \( P_n = 2^n \) for \( n \geq 0 \).

Proof by induction:

- Induction base. \( P_0 = 1 = 2^0 \) and \( P_1 = 2 = 2^1 \).
- Induction hypothesis. \( P_{n-1} = 2^{n-1} \) and \( P_{n-2} = 2^{n-2} \) for \( n \geq 2 \).
- Inductive step. For \( n \geq 2 \),

\[
\begin{align*}
P_n &= 5P_{n-1} - 6P_{n-2} \quad (\text{* definition of } P_n \text{ *)} \\
&= 5 \cdot 2^{n-1} - 6 \cdot 2^{n-2} \quad (\text{* induction hypothesis *)} \\
&= 5 \cdot 2^{n-1} - 3 \cdot 2^{n-1} \quad (\text{* algebra *)} \\
&= 2 \cdot 2^{n-1} \quad (\text{* algebra *)} \\
&= 2^n \quad (\text{* algebra *)}
\end{align*}
\]
3 Fibonacci numbers

1. The Fibonacci sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, … 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 \( F_n > 100 \)?

**Answer:** \( F_{11} = 89 \) and \( F_{12} = 144 \), therefore 12 is the smallest \( n \) for which \( F_n > 100 \).

(b) What is the smallest \( n \) for which \( F_n > 1000 \)?

**Answer:** \( F_{16} = 987 \) and \( F_{17} = 1597 \), therefore 17 is the smallest \( n \) for which \( F_n > 1000 \).

(c) Let \( A_n = \frac{(F_1 + F_2 + \cdots + F_n)}{n} \) be the average of the first \( n \) Fibonacci numbers. What is the smallest \( n \) for which \( A_n > 10 \)?

**Answer:** The sequence \( A_1, A_2, \ldots \) is

<table>
<thead>
<tr>
<th>( n )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A(n) )</td>
<td>1</td>
<td>( \frac{4}{3} )</td>
<td>( \frac{7}{4} )</td>
<td>( \frac{12}{5} )</td>
<td>( \frac{20}{6} )</td>
<td>( \frac{33}{7} )</td>
<td>( \frac{54}{8} )</td>
<td>88/9</td>
<td>143/10</td>
<td></td>
</tr>
</tbody>
</table>

Since \( A_9 = 88/9 < 10 \) and \( A_{10} = 143/10 > 10 \), it follows that 10 is the smallest \( n \) for which \( A_n > 10 \).

(d) Find all \( n \) for which \( F_n = n \).

**Answer:** By inspection, \( F_0 = 0, F_1 = 1 \), and \( F_5 = 5 \) while \( F_2 \neq 2, F_3 \neq 3, \) and \( F_4 \neq 4 \). Since \( F_n \) as a function of \( n \) grows faster than the function \( n \) for \( n > 5 \) (see the remark below), it follows that \( F_n > n \) for \( n > 5 \).

(e) Find all \( n \) for which \( F_n = n^2 \).

**Answer:** By inspection, \( F_0 = 0 = 0^2, F_1 = 1 = 1^2 \), and \( F_{12} = 144 = 12^2 \) while \( F_n \neq n^2 \) for \( n \in \{2, 3, \ldots, 11\} \). Since \( F_n \) as a function of \( n \) grows faster than the function \( n^2 \) for \( n > 12 \) (see the remark below), it follows that \( F_n > n^2 \) for \( n > 12 \).

**Remark:** \( F_{n+1}/F_n \approx \phi = 1.618 \ldots \). Therefore \( F_n \) as a function of \( n \) grows faster than the function \( n^2 \) for which \( (n+1)^2/n^2 \) approaches 1 as \( n \) tends to infinity. In particular after \( F_{12} = 144 \) for which \( F_n = n^2 \), it is always the case that \( F_n > n^2 \) for \( n > 12 \). This can be proven by induction. Similarly, after \( F_5 = 5 \) for which \( F_n = n \), it is always the case that \( F_n > n \) for \( n > 5 \).
2. Prove the following identity for \( n \geq 2 \):

\[
F_{n+1} + F_{n-1} = F_{n+2} - F_{n-2}
\]

**The cases** \( n = 2, 3, 4, 5, 6 \):

\[
\begin{align*}
F_3 + F_1 &= 2 + 1 = 3 = 3 - 0 = F_4 - F_0 \\
F_4 + F_2 &= 3 + 1 = 4 = 5 - 1 = F_5 - F_1 \\
F_5 + F_3 &= 5 + 2 = 7 = 8 - 1 = F_6 - F_2 \\
F_6 + F_4 &= 8 + 3 = 11 = 13 - 2 = F_7 - F_3 \\
F_7 + F_5 &= 13 + 5 = 18 = 21 - 3 = F_8 - F_4
\end{align*}
\]

**Proof:** For \( n \geq 2 \),

\[
\begin{align*}
F_{n+1} + F_{n-1} &= F_{n+1} + (F_n - F_{n-2}) \quad (* F_{n-1} = F_n - F_{n-2} *) \\
&= (F_{n+1} + F_n) - F_{n-2} \quad (* \text{rearranging parenthesis} *) \\
&= F_{n+2} - F_{n-2} \quad (* F_{n+2} = F_{n+1} + F_n *)
\end{align*}
\]
3. Prove the following identity for $n \geq 1$:

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

**Proof by induction:**

- **Induction base.** Verify correctness for $n = 1, 2, 3, 4$:

  $F_3 = 2 = 1^2 + 1^2 = F_2^2 + F_1^2$
  $F_5 = 5 = 2^2 + 1^2 = F_3^2 + F_2^2$
  $F_7 = 13 = 3^2 + 2^2 = F_4^2 + F_3^2$
  $F_9 = 34 = 5^2 + 3^2 = F_6^2 + F_4^2$

- **Induction hypothesis.** Assume that for $n \geq 3$:

  $F_{2n-1} = F_n^2 + F_{n-1}^2$
  $F_{2n-3} = F_{n-1}^2 + F_{n-2}^2$

- **Inductive step.** By replacing $F_{2n+1}$ with $F_{2n} + F_{2n-1}$, then replacing $F_{2n}$ with $F_{2n-1} + F_{2n-2}$ and combining terms, and then replacing $F_{2n-2}$ with $F_{2n-1} - F_{2n-3}$ and combining terms, $F_{2n+1}$ becomes a function of $F_{2n-1}$ and $F_{2n-3}$.

  $$F_{2n+1} = F_{2n} + F_{2n-1}$$
  $$= (F_{2n-1} + F_{2n-2}) + F_{2n-1}$$
  $$= 2F_{2n-1} + F_{2n-2}$$
  $$= 2F_{2n-1} + (F_{2n-1} - F_{2n-3})$$
  $$= 3F_{2n-1} - F_{2n-3}$$

After applying the induction hypothesis for $F_{2n-1}$ and $F_{2n-3}$ and simplifying, $F_{2n+1}$ becomes a function of the squares of $F_n$, $F_{n-1}$, and $F_{n-2}$.

  $$F_{2n+1} = 3F_{2n-1} - F_{2n-3}$$
  $$= 3(F_n^2 + F_{n-1}^2) - (F_{n-1}^2 + F_{n-2}^2)$$
  $$= 3F_n^2 + 2F_{n-1}^2 - F_{n-2}^2$$

By replacing $F_{n-2}$ with $F_n - F_{n-1}$ and simplifying, $F_{2n+1}$ becomes a function of the squares of $F_n$ and $F_{n-1}$.

  $$F_{2n+1} = 3F_n^2 + 2F_{n-1}^2 - F_{n-2}^2$$
  $$= 3F_n^2 + 2F_{n-1}^2 - (F_n - F_{n-1})^2$$
  $$= 3F_n^2 + 2F_{n-1}^2 - (F_n^2 - 2F_{n-1}F_n + F_{n-1}^2)$$
  $$= 2F_n^2 + F_{n-1}^2 + 2F_{n-1}F_n$$
  $$= (F_n^2 + F_{n-1}F_n + F_{n-1}^2) + F_n^2$$
  $$= (F_n + F_{n-1})^2 + F_n^2$$

The proof is completed by replacing $F_n + F_{n-1}$ with $F_{n+1}$.

  $$F_{2n+1} = (F_n + F_{n-1})^2 + F_n^2$$
  $$= F_{n+1}^2 + F_n^2$$
4. Define the following \((\text{almost Fibonacci})\) 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 \(G_0, G_1, \ldots, G_{10}\).

**Answer:** The first 11 values in the sequence \(G_0, G_1, \ldots, G_{10}\) are:

<table>
<thead>
<tr>
<th>(n)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G_n)</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>20</td>
<td>33</td>
<td>54</td>
<td>88</td>
<td>143</td>
</tr>
</tbody>
</table>

(b) Express \(G_n\) as a function of Fibonacci numbers.

**Answer:** There are two “natural” guesses for \(G_n\) as a function of Fibonacci numbers based on the first 13 numbers in the Fibonacci sequence:

<table>
<thead>
<tr>
<th>(n)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_n)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>55</td>
<td>89</td>
<td>144</td>
</tr>
</tbody>
</table>

\[ G_n = F_{n+2} - 1 \]
\[ G_n = F_0 + F_1 + \cdots + F_n \]

**Remark:** Observe that \(F_0 + F_1 + \cdots + F_n = F_{n+2} - 1\) is an identity that can be proven by induction.

(c) Prove that your expression for \(G_n\) is correct for all \(n \geq 0\).

**Proposition 1:** \(G_n = F_{n+2} - 1\) for \(n \geq 0\).

**Proof by induction:**

**Induction base.** Show that \(G_0 = F_2 - 1\) and that \(G_1 = F_3 - 1\):

\[
\begin{align*}
G_0 &= 0 = 1 - 0 = F_2 - 1 \\
G_1 &= 1 = 2 - 1 = F_3 - 1
\end{align*}
\]

**Induction hypothesis.** Assume that for \(n \geq 2\):

\[
\begin{align*}
G_{n-1} &= F_{n+1} - 1 \\
G_{n-2} &= F_n - 1
\end{align*}
\]

**Inductive step.** Prove that \(G_n = F_{n+2} - 1\) for \(n \geq 2\):

\[
\begin{align*}
G_n &= G_{n-1} + G_{n-2} + 1 \quad \text{(*) definition of } G_n \text{ *)} \\
&= (F_{n+1} - 1) + (F_n - 1) + 1 \quad \text{(*) induction hypothesis *)} \\
&= (F_{n+1} + F_n) - 1 \quad \text{(*) simplifying *)} \\
&= F_{n+2} - 1 \quad \text{(*) } F_{n+2} = F_{n+1} + F_n \text{ *)}
\end{align*}
\]
Proposition 2: \( G_n = F_0 + F_1 + \cdots + F_n \) for \( n \geq 0 \).

Proof by induction:

*Induction base.* Show that \( G_0 = F_0 \) and that \( G_1 = F_0 + F_1 \):

\[
G_0 = 0 = F_0 \\
G_1 = 1 = 0 + 1 = F_0 + F_1
\]

*Induction hypothesis.* Assume that for \( n \geq 2 \):

\[
G_{n-1} = F_0 + F_1 + F_2 + \cdots + F_{n-1} \\
G_{n-2} = F_0 + F_1 + \cdots + F_{n-2}
\]

*Inductive step.* Prove that \( G_n = F_0 + F_1 + \cdots + F_n \) for \( n \geq 2 \):

\[
G_n = G_{n-1} + G_{n-2} + 1 \\
= (F_0 + F_1 + F_2 + \cdots + F_{n-1}) + (F_0 + F_1 + \cdots + F_{n-2}) + 1 \\
= (F_0 + 1) + (F_1 + F_2 + \cdots + F_{n-1}) + (F_0 + F_1 + \cdots + F_{n-2}) \\
= F_1 + (F_1 + F_2 + \cdots + F_{n-1}) + (F_0 + F_1 + \cdots + F_{n-2}) \\
= F_1 + (F_1 + F_0) + (F_2 + F_1) + \cdots + (F_{n-1} + F_{n-2}) \\
= F_1 + F_2 + F_3 + \cdots + F_n \\
= F_0 + F_1 + F_2 + F_3 + \cdots + F_n
\]

\(*\) definition of \( G_n \) \(*\)

\(*\) induction hypothesis \(*\)

\(*\) rearranging terms \(*\)

\(*\) \( F_0 + 1 = F_1 \) \(*\)

\(*\) rearranging terms \(*\)

\(*\) Fibonacci recurrence \(*\)

\(*\) \( F_0 = 0 \) \(*\)
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 (0011110) 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.

**Answer:**

i. There are two 1-even binary strings of length 2

(00) (11)

There are two non-1-even binary string of length 2

(01) (10)

ii. There are three 1-even binary strings of length 3

(000) (011) (110)

There are five non-1-even binary string of length 3

(001) (010) (100) (101) (111)

iii. There are five 1-even binary strings of length 4

(0000) (0011) (0110) (1100) (1111)

There are eleven non-1-even binary string of length 4

(0001) (0010) (0100) (0101) (0111) (1000) (1001) (1010) (1011) (1101) (1110)

(b) For \( k \geq 1 \), as a function of \( k \), how many 1-even binary strings of length \( k \) exist?

**Answer:** For \( k \geq 1 \), denote by \( G_k \) the number of 1-even binary strings of length \( k \). Trivially, (0) is a 1-even binary string and (1) is not. Combined this with the results from parts (a), (b), and (c), it follows that \( G_1 = 1, G_2 = 2, G_3 = 3, \) and \( G_4 = 5 \). Recall that the first six Fibonacci numbers \( F_k \) for \( 0 \leq k \leq 5 \) are \( F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, \) and \( F_5 = 5 \).

**Proposition:** For \( k \geq 1 \), there are \( G_k = F_{k+1} \) 1-even binary strings of length \( k \).

**Proof Sketch:** The claim is true for \( k = 1 \) and \( k = 2 \). It remains to show that \( G_k = G_{k-1} + G_{k-2} \) for \( k \geq 3 \). Consider the following two cases depending on the value of the first bit of a given 1-even binary string.

* The first bit is 0: In this case, the remaining \( k - 1 \) bits must be a 1-even binary string of length \( k - 1 \).

* The first bit is 1: In this case, the second bit must be 1 and the remaining \( k - 2 \) bits must be a 1-even binary string of length \( k - 2 \).

In the other direction, every 1-even binary string of length \( k - 1 \) becomes a 1-even binary string of length \( k \) by appending 0 at its beginning and every 1-even binary string of length \( k - 2 \) becomes a 1-even binary string of length \( k \) by appending 11 at its beginning. The above arguments can be used to rigorously prove the correctness of the recursive formula \( G_k = G_{k-1} + G_{k-2} \) and complete the proof.

**Example:** In part (c) the first three 1-even binary strings of length 4 are the three 1-even binary strings of length 3 with 0 appended at their beginning and the last two 1-even binary strings of length 4 are the two 1-even binary strings of length 2 with 10 appended at their beginning.
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.

i. There are three 1-lonely binary strings of length 2
   - (00)
   - (01)
   - (10)

   There is only one non-1-lonely binary string of length 2
   - (11)

ii. There are five 1-lonely binary strings of length 3
    - (000)
    - (001)
    - (010)
    - (100)
    - (101)

   There are three non-1-lonely binary strings of length 3
    - (011)
    - (110)
    - (111)

iii. There are eight 1-lonely binary strings of length 4
     - (0000)
     - (0001)
     - (0010)
     - (0100)
     - (0101)
     - (1000)
     - (1001)
     - (1010)

   There are eight non-1-lonely binary strings of length 4
     - (0011)
     - (0110)
     - (0111)
     - (1011)
     - (1100)
     - (1101)
     - (1110)
     - (1111)

(b) For \( k \geq 1 \), as a function of \( k \), how many 1-lonely binary strings of length \( k \) exist? Justify your answer.

For \( k \geq 1 \), denote by \( G_k \) the number of 1-lonely binary strings of length \( k \). Trivially, both strings (0) and (1) of length 1 are 1-lonely strings. Combined this with the results from parts (a), (b), and (c), it follows that \( G_1 = 2 \), \( G_2 = 3 \), \( G_3 = 5 \), and \( G_4 = 8 \). Recall that the first seven Fibonacci numbers \( F_k \) for \( 0 \leq k \leq 6 \) are \( F_0 = 0 \), \( F_1 = 1 \), \( F_2 = 1 \), \( F_3 = 2 \), \( F_4 = 3 \), \( F_5 = 5 \), and \( F_6 = 8 \).

**Proposition:** For \( k \geq 1 \), there are \( G_k = F_{k+2} \) 1-lonely binary strings of length \( k \).

**Proof Sketch:** The claim is true for \( k = 1 \) and \( k = 2 \). It remains to show that \( G_k = G_{k-1} + G_{k-2} \) for \( k \geq 3 \). Consider the following two cases depending on the value of the first bit of a given 1-lonely binary string.

- The first bit is 0: In this case, the remaining \( k - 1 \) bits must be a 1-lonely binary string of length \( k - 1 \).
- The first bit is 1: In this case, the second bit must be 0 and the remaining \( k - 2 \) bits must be a 1-lonely binary string of length \( k - 2 \).

In the other direction, every 1-lonely binary string of length \( k - 1 \) becomes a 1-lonely binary string of length \( k \) by appending 0 at its beginning and every 1-lonely binary string of length \( k - 2 \) becomes a 1-lonely binary string of length \( k \) by appending 10 at its beginning. The above arguments can be used to rigorously prove the correctness of the recursive formula \( G_k = G_{k-1} + G_{k-2} \) and complete the proof.

**Example:** In part (c) the first five 1-lonely binary strings of length 4 are the five 1-lonely binary strings of length 3 with 0 appended at their beginning and the last three 1-lonely binary strings of length 4 are the three 1-lonely binary strings of length 2 with 10 appended at their beginning.
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 \( \pi(3) = 5 \).

**Answer:** For \( n \geq 1 \), there are \( F_{n+1} \) permutations \( \pi = (\pi(1), \pi(2), \ldots, \pi(n)) \) of the numbers \{1, 2, \ldots, n\} in which the value of \( \pi(i) \) is either \( i-1 \), or \( i \), or \( i+1 \) for all \( 1 \leq i \leq n \) where \( F_n \) is the \( n^{\text{th}} \) Fibonacci number.

**Proof:** For \( n \geq 1 \), a permutation \( \pi \) is **good** if \( \pi(i) \) is either \( i-1 \), or \( i \), or \( i+1 \) for all \( 1 \leq i \leq n \). Otherwise, \( \pi \) is a **bad** permutation. Equivalently, if \( \pi \) is a **bad** permutation then there exists at least one index \( i \) for which \( |\pi(i) - i| \geq 2 \) while if \( \pi \) is a **good** permutation then \( |\pi(i) - i| \leq 1 \) for all \( 1 \leq i \leq n \).

\[ n = 1: \text{The only permutation (1) is a **good** permutation.} \]
\[ n = 2: \text{The two permutations (1, 2) and (2, 1) are **good** permutations.} \]
\[ n = 3: \text{Out of the six permutations, the three permutations (1, 2, 3), (2, 1, 3), and (1, 3, 2) are **good** permutations while the three permutations (3, 1, 2), (2, 3, 1), and (3, 2, 1) are **bad** permutations since either } \pi(1) = 3 \text{ or } \pi(3) = 1. \]
\[ n = 4: \text{Out of the 24 permutations, only the five permutations (1, 2, 3, 4), (2, 1, 3, 4), (1, 3, 2, 4), (1, 2, 4, 3), and (2, 1, 4, 3) are **good** permutations.} \]

For \( n \geq 1 \), let \( G_n \) be the number of **good** permutations. The above shows that \( G_1 = 1 = F_2 \), \( G_2 = 2 = F_3 \), \( G_3 = 3 = F_4 \), and \( G_4 = 5 = F_5 \).

For \( n \geq 3 \), let \( \pi \) be a **good** permutation. Therefore, \( \pi(n) = n \) or \( \pi(n) = n - 1 \) because otherwise \( |\pi(n) - n| \geq 2 \).

- Assume \( \pi(n) = n \). Then \( \pi' = (\pi(1), \pi(2), \ldots, \pi(n-1)) \) must be a **good** permutation for the numbers \{1, 2, \ldots, n - 1\}. It follows that there are \( G_{n-1} \) **good** permutations \( \pi \) in which \( \pi(n) = n \).
- Assume \( \pi(n) = n - 1 \). Then \( \pi(n-1) = n \) because otherwise \( \pi(i) = n \) for some \( i \leq n - 2 \) which contradicts the **goodness** of \( \pi \). Moreover, \( \pi'' = (\pi(1), \pi(2), \ldots, \pi(n-2)) \) must be a **good** permutation for the numbers \{1, 2, \ldots, n - 2\}. It follows that there are \( G_{n-2} \) **good** permutations \( \pi \) in which \( \pi(n) = n - 1 \) and \( \pi(n-1) = n \).

The above two cases imply that \( G_n = G_{n-1} + G_{n-2} \) for \( n \geq 3 \). Since \( G_1 = 1 = F_2 \) and \( G_2 = 2 = F_3 \), it follows that the number of **good** permutations for \( n \geq 1 \) is the Fibonacci number \( F_{n+1} \).

**Example:** For \( n = 5 \), there are \( F_6 = 8 \) **good** permutations. In \( F_5 = 5 \) of them \( \pi(5) = 5 \):

\[
(1, 2, 3, 4, 5) \quad (2, 1, 3, 4, 5) \quad (1, 3, 2, 4, 5) \quad (1, 2, 4, 3, 5) \quad (2, 1, 4, 3, 5)
\]

and in \( F_4 = 3 \) of them \( \pi(4) = 5 \) and \( \pi(5) = 4 \):

\[
(1, 2, 3, 5, 4) \quad (2, 1, 3, 5, 4) \quad (1, 3, 2, 5, 4)
\]
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 \( S[1] = [239, 253, 127, 191, 254, 247, 223, 251] \) and \( M = 255 \).

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?

A by-definition \( \Theta(n^2) \)-Algorithm: Compute \( S[i] = A[1] + \cdots + A[i-1] + A[i+1] + \cdots + A[n] \) for all \( 1 \leq i \leq n \).

**Complexity:** For \( 1 \leq i \leq n \), computing \( S_i \) is done by exactly \( n - 2 \) plus operations. Therefore, the total number of plus operations in this algorithm is \( n(n - 2) = n^2 - 2n = \Theta(n^2) \).

A \( \Theta(n) \)-Algorithm:

- Compute the prefix-sum of the first \( n - 1 \) integers in \( A \). For \( 1 \leq i \leq n - 1 \), let \( P[i] = \sum_{j=1}^{i} A[j] \).
- Compute the suffix-sum of the last \( n - 1 \) integers in \( A \). For \( n \geq i \geq 2 \) let \( Q[i] = \sum_{j=i}^{n} A[j] \).
- Compute the array \( S \)

\[
S[i] = \begin{cases} 
Q[2] & \text{for } i = 1 \\
P[n-1] & \text{for } i = n \\
P[i-1] + Q[i+1] & \text{for } 2 \leq i \leq n - 1
\end{cases}
\]

**Correctness:** By definition, \( S_1 = Q[2] \) and \( S_n = P[n-1] \). Fix \( 2 \leq i \leq n - 1 \). Then \( P[i-1] = A[1] + \cdots + A[i-1] \) and \( Q[i+1] = A[i+1] + \cdots + A[n] \). Therefore,

\[
\]

**Remark:** Note that there is no need to compute the last values of the prefix-sum \( (P[n]) \) and the last value of the suffix-sum \( (Q[1]) \) because they are not required for the computations of \( S[1], S[2], \ldots, S[n] \).

**Complexity:** The \( n - 1 \) prefix-sum values can be computed with \( n - 2 \) plus operations and so are the \( n - 1 \) suffix-sum values. Then, for \( 2 \leq i \leq n - 1 \), all the \( S_i \) values are computed with \( n - 2 \) plus operations. The total number of plus operations in this algorithm is

\[
(n - 2) + (n - 2) + (n - 2) = 3n - 6 = \Theta(n)
\]
Example:

\[
A = [16, 2, 128, 64, 1, 8, 32, 4]
\]

\[
P = [16, 18, 146, 210, 211, 219, 251, *]
\]

\[
Q = [* , 239, 237, 109, 45, 44, 36, 4]
\]

\[
S = [239, 253, 127, 191, 254, 247, 223, 251]
\]

\[
\]

\[
\]

\[
\]

\[
\]

\[
\]

\[
\]

\[
\]

\[
\]
2. For \( n \geq 3 \), let \( A = A[1] < A[2] < \cdots < A[n] \) be a sorted array with \( n \) distinct positive integers. Describe an efficient algorithm that finds two integers from the array whose sum is even. What is the complexity of your algorithm?

**A by-definition algorithm:** Examine the sums of all possible \( \binom{n}{2} \) pairs of integers from the array until finding a pair whose sum is even.

Algorithm \( X(A) \):

```plaintext
for i := 1 to n − 1 do
    for j := i + 1 to n do
        if \( A[i] + A[j] \) is even
            then return \((A[i], A[j])\)

return \("there are no two integers in A whose sum is even")
```

Algorithm \( X \) is correct because it inspects the sums of all possible pairs in \( A \).

The complexity of algorithm \( X \) is \( O(n^2) \) because in the worst-case the two loops terminate when there are no two integers in \( A \) whose sum is even. However, if \( A[1] \) is even then \( i \) will be 2 only if the rest of the integers in the array including \( A[2] \) and \( A[3] \) are odd. But then \( A[2] + A[3] \) is even and \( i \) is never greater than 2. Similarly, if \( A[1] \) is odd then \( i \) will be 2 only if the rest of the integers in the array including \( A[2] \) and \( A[3] \) are even. But then \( A[2] + A[3] \) is even and \( i \) is never greater than 2. As a result, the worst-case complexity of algorithm \( X \) is \( \Theta(n) \).

**A constant time algorithm:** The \( \Theta(n) \) analysis of algorithm \( X \) works even if only the first three integers in \( A \) are examined. This is because two of the integers among \( A[1], A[2], A[3] \) must have the same parity and as a result their sum is even.

Algorithm \( Y(A) \):

```plaintext
```

There are constant number of operations in algorithm \( Y \) and therefore its complexity is \( \Theta(1) \).

**Remark:** Algorithm \( Y \) works with any three integers from the array \( A \) and it works even if \( A \) is not sorted.