COMP 2804 — Solutions Assignment 2

Question 1:

- Write your name and student number.

Solution:

- Name: Frenkie de Jong
- Student number: 21

Question 2: The function $f : \{1,2,3,\ldots\} \rightarrow \mathbb{R}$ is defined by

$$\begin{array}{rcl} f(1) & = & 2, \\ f(n) & = & \frac{1}{2}\left(f(n-1) + \frac{1}{f(n-1)}\right) \quad \text{if } n \geq 2. \end{array}$$

- Prove that for every integer $n \geq 1$,

$$f(n) = \frac{3^{2^{n-1}}+1}{3^{2^{n-1}}-1}.$$

Note that $3^{2^{n-1}}$ is 3 to the power of 2^{n-1}.

Solution: We prove the claim by induction on n.

The base case is when $n = 1$. Since $f(1) = 2$ and

$$\frac{3^{2^{1-1}}+1}{3^{2^{1-1}}-1} = \frac{3^{2^0}+1}{3^{2^0}-1} = \frac{3^1+1}{3^1-1} = \frac{3+1}{3-1} = 2,$$

the base case holds.

For the induction step, let $n \geq 2$ be an integer, and assume that the claim holds for $n-1$. Thus, we assume that

$$f(n-1) = \frac{3^{2^{n-2}}+1}{3^{2^{n-2}}-1}.$$

We will use the recurrence to prove that the claim also holds for n:

$$\begin{array}{rcl} f(n) & = & \frac{1}{2}\left(f(n-1) + \frac{1}{f(n-1)}\right) \\ & = & \frac{1}{2}\left(\frac{3^{2^{n-2}}+1}{3^{2^{n-2}}-1} + \frac{3^{2^{n-2}}-1}{3^{2^{n-2}}+1}\right) \\ & = & \frac{1}{2} \cdot \frac{\left(3^{2^{n-2}}+1\right)^2 + \left(3^{2^{n-2}}-1\right)^2}{\left(3^{2^{n-2}}-1\right)\left(3^{2^{n-2}}+1\right)}. \end{array}$$

Both terms on the top of this fraction are of the form

$$(x \pm 1)^2 = x^2 \pm 2x + 1,$$

whereas the term at the bottom is of the form

$$(x-1)(x+1) = x^2 - 1,$$

where $x = 3^{2^{n-2}}$. Note that

$$x^2 = \left(3^{2^{n-2}}\right)^2 = 3^{2 \cdot 2^{n-2}} = 3^{2^{n-1}}.$$

This gives

$$\begin{aligned} f(n) &= \frac{1}{2} \cdot \frac{(x^2+2x+1)+(x^2-2x+1)}{x^2-1} \\ &= \frac{1}{2} \cdot \frac{2x^2+2}{x^2-1} \\ &= \frac{x^2+1}{x^2-1} \\ &= \frac{3^{2^{n-1}}+1}{3^{2^{n-1}}-1}. \end{aligned}$$

This proves the induction step.

Question 3: The function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$\begin{aligned} f(m,0) &= 0, && \text{if } m \geq 0, \\ f(m,n) &= m + f(m,n-1) && \text{if } m \geq 0 \text{ and } n \geq 1. \end{aligned}$$

- Solve this recurrence, i.e., express $f(m,n)$ in terms of m and n only. As always, prove that your answer is correct.

Solution: We consider an example, from which we (hopefully) see the solution:

$$\begin{aligned} f(7,3) &= 7 + f(7,2) \\ &= 7 + (7 + f(7,1)) \\ &= 7 + 7 + (7 + f(7,0)) \\ &= 7 + 7 + 7 + 0 \\ &= 21. \end{aligned}$$

From this, it looks like the function f does multiplication by repeated addition. Thus, we guess that for all $m \geq 0$ and $n \geq 0$,

$$f(m,n) = mn.$$

We will prove by induction that this is correct. We choose an integer $m \geq 0$. Now we do induction on n.

The base case is when $n = 0$. Since $f(m, 0) = 0$ and $m \cdot 0 = 0$, the base case holds.

For the induction step, let $n \geq 1$ be an integer, and assume that the claim holds for $n-1$. Thus, we assume that

$$f(m, n-1) = m(n-1).$$

We will use the recurrence to prove that the claim also holds for n:

$$\begin{aligned} f(m,n) &= m + f(m, n-1) \\ &= m + m(n-1) \\ &= mn. \end{aligned}$$

This proves the induction step.

Question 4: In class, we have seen that for any integer $m \geq 1$, the number of 00-free bitstrings of length m is equal to f_{m+2}, which is the $(m+2)$-th Fibonacci number.

Let $n \geq 2$ be an integer. For each of the following, justify your answer.

- How many 00-free bitstrings of length n do not contain any 0?

  Solution: There is only one such bitstring: the bitstring consisting of n many 1's.

- How many 00-free bitstrings of length n have the following property: The rightmost 0 is at position 1.

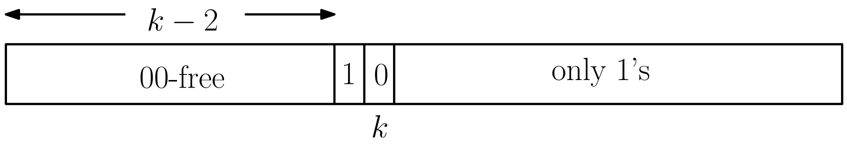
  Solution: If the rightmost 0 is at position 1, then there is a 1 at each of the positions $2, 3, \ldots, n$. This bitstring is 00-free. Thus, the answer is 1.

- How many 00-free bitstrings of length n have the following property: The rightmost 0 is at position 2.

  Solution: If the rightmost 0 is at position 2, then there is a 1 at position 1, and there is a 1 at each of the positions $3, 4, \ldots, n$. This bitstring is 00-free. Thus, the answer is 1.

- Let k be an integer with $3 \leq k \leq n$. How many 00-free bitstrings of length n have the following property: The rightmost 0 is at position k.

  Solution: Assume the rightmost 0 is at position k. Then there is a 1 at position $k-1$, and there is a 1 at each of the positions $k+1, \ldots, n$. The positions $1, 2, \ldots, k-2$ contain an arbitrary 00-free bitstring of length $k-2$. Thus, the answer is f_k.



- Use the previous results to prove that

$$f_{n+2} = 1 + \sum_{k=1}^{n} f_k .$$

  Solution: We know that the total number of 00-free bitstrings of length n is equal to f_{n+2}. We are going to divide these bitstrings into groups, depending on the position of the rightmost 0; there is one extra group for the bitstrings that do not contain any 0. All these groups have been covered in the previous parts. Thus, if we add the answers to the previous parts, then we have counted each 00-free bitstring of length n exactly once. This gives

$$\begin{aligned} f_{n+2} &= (\text{part } 1) + (\text{part } 2) + (\text{part } 3) + \sum_{k=3}^{n} (\text{part } 4) \\ &= 1 + 1 + 1 + \sum_{k=3}^{n} f_k \\ &= 1 + f_1 + f_2 + \sum_{k=3}^{n} f_k \\ &= 1 + \sum_{k=1}^{n} f_k . \end{aligned}$$

Question 5: Let $n \geq 1$ be an integer and consider a set S consisting of n numbers. A function $f : S \rightarrow S$ is called *cool*, if for all elements x of S,

$$f(f(f(x))) = x.$$

Let A_n be the number of cool functions $f : S \rightarrow S$.

- Let $f : S \rightarrow S$ be a cool function, and let x be an element of S. Prove that the set

$$\{x, f(x), f(f(x))\}$$

  has size 1 or 3.

  Solution: If x, $f(x)$, and $f(f(x))$ are pairwise distinct, then the set

$$X = \{x, f(x), f(f(x))\}$$

  has size 3.

  We will show the following: If two of x, $f(x)$, and $f(f(x))$ are equal, then all three are equal. This will imply that the set X cannot have size 2.

1. If $x = f(x)$, then $f(x) = f(f(x))$.
2. If $x = f(f(x))$, then $f(x) = f(f(f(x))) = x$.
3. If $f(x) = f(f(x))$, then $f(f(x)) = f(f(f(x))) = x$.

- Let $f : S \to S$ be a cool function, and let x and y be two distinct elements of S. Assume that $f(y) = y$. Prove that $f(x) \neq y$.

  Solution: We will prove this by contradiction. Thus, we assume that $f(x) = y$. Then

  $$f(f(x)) = f(y) = y,$$

  which implies that

  $$f(f(f(x))) = f(y) = y.$$

  Since f is cool, this implies that $x = y$, which is a contradiction.

- Prove that for any integer $n \geq 4$,

  $$A_n = A_{n-1} + (n-1)(n-2) \cdot A_{n-3}.$$

  Hint: Let y be the largest element in S. Some cool functions f have the property that $f(y) = y$, whereas some other cool functions f have the property that $f(y) \neq y$.

  Solution: Let S be a set of size n. The number of cool functions $f : S \to S$ is equal to A_n.

  We fix an arbitrary element y in S. (Note that the argument will work for any y, even if it is not the largest element in S.)

  We are going to divide the cool functions $f : S \to S$ into two groups.

  Group 1: All cool functions f for which $f(y) = y$.

  Thus, for each function f in this group, the value $f(y)$ is fixed.

  From the second part of this question, we know the following: If $x \in S \setminus \{y\}$, then $f(x) \in S \setminus \{y\}$. This implies the following: If we restrict the domain of the function f to $S \setminus \{y\}$, then we obtain a cool function

  $$f : S \setminus \{y\} \to S \setminus \{y\}.$$

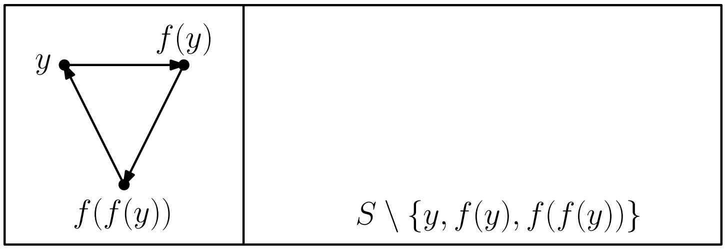
  Since the set $S \setminus \{y\}$ has size $n-1$, the number of functions in this group is equal to A_{n-1}.

  Group 2: All cool functions f for which $f(y) \neq y$.

  Task 1: Choose $f(y)$. There are $n-1$ ways to do this.

  Task 2: Choose $f(f(y))$.

  Since $f(y) \neq y$, we know from the first part that $f(f(y)) \notin \{y, f(y)\}$. Thus, there are $n-2$ ways to do this second task.



Task 3: Specify $f(x)$ for each $x \notin \{y, f(y), f(f(y))\}$.

We observe: if $x \notin \{y, f(y), f(f(y))\}$, then $f(x) \notin \{y, f(y), f(f(y))\}$. (Why: because otherwise, $f(f(f(x))) \neq x$.)

This implies the following: If we restrict the domain of the function f to $S \setminus \{y, f(y), f(f(y))\}$, then we obtain a cool function

$$f : S \setminus \{y, f(y), f(f(y))\} \rightarrow S \setminus \{y, f(y), f(f(y))\}.$$

Since the set $S \setminus \{y, f(y), f(f(y))\}$ has size $n-3$, the number of ways to do this third task is equal to A_{n-3}.

By the Product Rule, the number of functions in Group 2 is equal to

$$(n-1)(n-2) \cdot A_{n-3}.$$

Conclusion:

$$\begin{aligned} A_n &= \text{number of cool functions } f : S \rightarrow S \\ &= \text{size of Group } 1 + \text{size of Group } 2 \\ &= A_{n-1} + (n-1)(n-2) \cdot A_{n-3}. \end{aligned}$$

Question 6: In this exercise, we will denote Boolean variables by lowercase letters, such as p and q. A *proposition* is any Boolean formula that can be obtained by applying the following recursive rules:

1. For every Boolean variable p, p is a proposition.
2. If f is a proposition, then $\neg f$ is also a proposition.
3. If f and g are propositions, then $(f \vee g)$ is also a proposition.
4. If f and g are propositions, then $(f \wedge g)$ is also a proposition.

- Let p and q be Boolean variables. Prove that

$$\neg ((p \wedge \neg q) \vee (\neg p \vee q))$$

is a proposition.

Solution: We have to show that the given logical formula can be "built" using the rules 1.—4. Here we go:

A: From 1.: p is a proposition.

B: From 1.: q is a proposition.

C: From B and 2.: $\neg q$ is a proposition.

D: From A, C, and 4.: $(p \wedge \neg q)$ is a proposition.

E: From A and 2.: $\neg p$ is a proposition.

F: From E, B, and 3.: $(\neg p \vee q)$ is a proposition.

G: From D, F, and 3.:

$$((p \wedge \neg q) \vee (\neg p \vee q))$$

is a proposition.

H: From G and 2.:

$$\neg((p \wedge \neg q) \vee (\neg p \vee q))$$

is a proposition.

- Let $\uparrow$ denote the *not-and* operator. In other words, if f and g are Boolean formulas, then $(f \uparrow g)$ is the Boolean formula that has the following truth table (0 stands for *false*, and 1 stands for *true*):

f	g	$(f \uparrow g)$
0	0	1
0	1	1
1	0	1
1	1	0

  – Let p be a Boolean variable. Use a truth table to prove that the Boolean formulas $(p \uparrow p)$ and $\neg p$ are equivalent.

  Solution: Here is the truth table:

p	$(p \uparrow p)$	$\neg p$
0	1	1
1	0	0

  Since the two rightmost columns are equal, $(p \uparrow p)$ and $\neg p$ are equivalent.

  – Let p and q be Boolean variables. Use a truth table to prove that the Boolean formulas $((p \uparrow p) \uparrow (q \uparrow q))$ and $p \vee q$ are equivalent.

  Solution: Here is the truth table:

p	q	$(p \uparrow p)$	$(q \uparrow q)$	$((p \uparrow p) \uparrow (q \uparrow q))$	$p \vee q$
0	0	1	1	0	0
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	1	1

Since the two rightmost columns are equal, $((p \uparrow p) \uparrow (q \uparrow q))$ and $p \vee q$ are equivalent.

- Let p and q be Boolean variables. Express the Boolean formula $(p \wedge q)$ as an equivalent Boolean formula that only uses the $\uparrow$-operator. Use a truth table to justify your answer.

  Solution: After some trying, we think that $((p \uparrow q) \uparrow (p \uparrow q))$ and $p \wedge q$ are equivalent. To verify this, we construct the truth table:

p	q	$(p \uparrow q)$	$((p \uparrow q) \uparrow (p \uparrow q))$	$p \wedge q$
0	0	1	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

  Since the two rightmost columns are equal, $((p \uparrow q) \uparrow (p \uparrow q))$ and $p \wedge q$ are equivalent.

  Note: We could have used De Morgan:

  $$p \wedge q = \neg\neg(p \wedge q) = \neg(\neg p \vee \neg q).$$

  On the right-hand side, we only use the logical operators $\neg$ and $\vee$, which we have done before.

- Prove that any proposition can be written as an equivalent Boolean formula that only uses the $\uparrow$-operator.

  Solution: We are going to show the following: Let f be an arbitrary proposition (thus, f only uses Boolean variables, $\neg$, $\vee$, and $\wedge$). Then f can be converted to an equivalent Boolean formula f' that only uses the $\uparrow$-operator.

  Since propositions are defined using recursion, the proof is by induction.

  Base case: The base case is when $f = p$, where p is a Boolean variable. In this case, we convert f to $f' = p$.

  Induction step 1: Assume the claim is true for the proposition f. Thus, f can be converted to an equivalent Boolean formula f' that only uses the $\uparrow$-operator.

  Then, we convert the proposition $\neg f$ to $(f' \uparrow f')$. Note that the latter only uses the $\uparrow$-operator.

  Induction step 2: Assume the claim is true for the propositions f and g. Thus, f can be converted to an equivalent Boolean formula f' that only uses the $\uparrow$-operator, and g can be converted to an equivalent Boolean formula g' that only uses the $\uparrow$-operator.

  Then, we convert the proposition $(f \vee g)$ to $((f' \uparrow f') \uparrow (g' \uparrow g'))$. Note that the latter only uses the $\uparrow$-operator.

Induction step 3: Assume the claim is true for the propositions f and g. Thus, f can be converted to an equivalent Boolean formula f' that only uses the $\uparrow$-operator, and g can be converted to an equivalent Boolean formula g' that only uses the $\uparrow$-operator.

Then, we convert the proposition $(f \wedge g)$ to $((f' \uparrow g') \uparrow (f' \uparrow g'))$. Note that the latter only uses the $\uparrow$-operator.

Question 7: In this exercise, we consider strings of characters, where each character is an element of $\{a, b, c\}$. Such a string is called *awesome*, if it does not contain the substring ab and does not contain the substring ba. For any integer $n \geq 1$, let

1. S_n denote the number of awesome strings of length n,
2. A_n denote the number of awesome strings of length n that start with a,
3. B_n denote the number of awesome strings of length n that start with b,
4. C_n denote the number of awesome strings of length n that start with c.

- Determine S_1 and S_2.

  Solution: If $n = 1$: There are 3 possible strings of length 1: a, b, and c. Since all of these are awesome, we have $S_1 = 3$.

  If $n = 2$: There are $3^2 = 9$ possible strings. Out of these, only ab and ba are not awesome. Therefore, $S_2 = 9 - 2 = 7$.

- Let $n \geq 1$ be an integer. Express S_n in terms of A_n, B_n, and C_n.

  Solution: Since S_n counts all awesome strings of length n, and each of them starts with one of a, b, and c, it is obvious that

$$S_n = A_n + B_n + C_n. \tag{1}$$

- Let $n \geq 2$ be an integer. Express C_n in terms of S_{n-1}.

  Solution: Each string of length n that is counted in C_n is awesome and starts with c. If we delete the first letter, then we are left with an arbitrary awesome string of length $n - 1$ (without any restriction on the first letter). Therefore,

$$C_n = S_{n-1}. \tag{2}$$

- Let $n \geq 2$ be an integer. Prove that

$$S_n = (S_{n-1} - B_{n-1}) + (S_{n-1} - A_{n-1}) + S_{n-1}. \tag{3}$$

  Solution: The number of awesome strings of length n is equal to S_n. We divide them into three groups (based on the first letter):

1. Awesome strings of length n that start with a.

   If we delete the first letter, then we obtain an awesome string of length $n-1$ that does *not* start with b. The number of such strings is equal to $S_{n-1} - B_{n-1}$.

2. Awesome strings of length n that start with b.

   If we delete the first letter, then we obtain an awesome string of length $n-1$ that does *not* start with a. The number of such strings is equal to $S_{n-1} - A_{n-1}$.

3. Awesome strings of length n that start with c.

   If we delete the first letter, then we obtain an awesome string of length $n-1$ (without any restriction on the first letter). The number of such strings is equal to S_{n-1}.

From this, (3) follows.

- Let $n \geq 3$ be an integer. Prove that

$$S_n = 2 \cdot S_{n-1} + S_{n-2}.$$

Solution: We have

$$\begin{aligned} S_n &= (S_{n-1} - B_{n-1}) + (S_{n-1} - A_{n-1}) + S_{n-1} \qquad \text{(from (3))} \\ &= 2 \cdot S_{n-1} + (S_{n-1} - A_{n-1} - B_{n-1}) \qquad \text{(basic algebra)} \\ &= 2 \cdot S_{n-1} + C_{n-1} \qquad \text{(from (1))} \\ &= 2 \cdot S_{n-1} + S_{n-2}. \qquad \text{(from (2))} \end{aligned}$$

- Prove that for every integer $n \geq 1$,

$$S_n = \frac{1}{2}\left(1+\sqrt{2}\right)^{n+1} + \frac{1}{2}\left(1-\sqrt{2}\right)^{n+1}.$$

Hint: What are the solutions of the equation $x^2 = 2x + 1$? Using these solutions will simplify the proof.

Solution:

The equation $x^2 = 2x + 1$ has two solutions: $\alpha = 1 + \sqrt{2}$ and $\beta = 1 - \sqrt{2}$. Note that

$$\alpha^2 = 2\alpha + 1,$$

$$\beta^2 = 2\beta + 1,$$

and

$$\alpha + \beta = 2.$$

We have to show that for all $n \geq 1$,

$$S_n = \frac{1}{2}\left(\alpha^{n+1} + \beta^{n+1}\right).$$

The first base case is when $n = 1$. Since $S_1 = 3$ and

$$\begin{aligned} \frac{1}{2}\left(\alpha^2 + \beta^2\right) &= \frac{1}{2}(2\alpha + 1 + 2\beta + 1) \\ &= \frac{1}{2} \cdot 6 \\ &= 3, \end{aligned}$$

the first base case holds.

The second base case is when $n = 2$. We know that $S_2 = 7$. Thus, we have to show that

$$\frac{1}{2}\left(\alpha^3 + \beta^3\right) = 7.$$

Note that

$$\begin{aligned} \alpha^3 &= \alpha \cdot \alpha^2 \\ &= \alpha(2\alpha + 1) \\ &= 2\alpha^2 + \alpha \\ &= 2(2\alpha + 1) + \alpha \\ &= 5\alpha + 2. \end{aligned}$$

By the same algebra, we get

$$\beta^3 = 5\beta + 2.$$

It follows that

$$\begin{aligned} \frac{1}{2}\left(\alpha^3 + \beta^3\right) &= \frac{1}{2}(5\alpha + 2 + 5\beta + 2) \\ &= \frac{1}{2} \cdot 14 \\ &= 7. \end{aligned}$$

This proves the second base case.

For the induction step, let $n \geq 3$ and assume that the claim is true for $n - 1$ and $n - 2$. Thus, we assume that

$$S_{n-1} = \frac{1}{2}(\alpha^n + \beta^n)$$

and

$$S_{n-2} = \frac{1}{2}\left(\alpha^{n-1} + \beta^{n-1}\right).$$

By applying the recurrence and the induction hypothesis, we get

$$
\begin{aligned}
S_n &= 2 \cdot S_{n-1} + S_{n-2} \\
&= (\alpha^n + \beta^n) + \frac{1}{2}\left(\alpha^{n-1} + \beta^{n-1}\right) \\
&= \frac{1}{2}\left(2\alpha^n + 2\beta^n + \alpha^{n-1} + \beta^{n-1}\right) \\
&= \frac{1}{2}\left(\alpha^{n-1}(2\alpha+1) + \beta^{n-1}(2\beta+1)\right) \\
&= \frac{1}{2}\left(\alpha^{n-1}\left(\alpha^2\right) + \beta^{n-1}\left(\beta^2\right)\right) \\
&= \frac{1}{2}\left(\alpha^{n+1} + \beta^{n+1}\right).
\end{aligned}
$$

This proves the induction step.

Question 8: Consider the following recursive algorithm, which takes as input a sequence $(a_1, a_2, \ldots, a_n)$ of n numbers, where n is a power of two, i.e., $n = 2^k$ for some integer $k \geq 0$:

```
Algorithm Mystery($a_1, a_2, \ldots, a_n$):

    if $n = 1$
    then return $a_1$
    else for $i = 1$ to $n/2$
        do $b_i = \min(a_{2i-1}, a_{2i})$        $(*)$
        endfor;
        Mystery($b_1, b_2, \ldots, b_{n/2}$)
    endif
```

- Determine the output of algorithm Mystery$(a_1, a_2, \ldots, a_n)$. As always, justify your answer.
- For any integer $n \geq 1$ that is a power of two, let $T(n)$ be the number of times that line $(*)$ is executed when running algorithm Mystery$(a_1, a_2, \ldots, a_n)$. Derive a recurrence for $T(n)$ and use it to prove that for any integer $n \geq 1$ that is a power of two,

$$T(n) = n - 1.$$

Solution: The output of algorithm Mystery$(a_1, a_2, \ldots, a_n)$ is the minimum of the input numbers $a_1, a_2, \ldots, a_n$. The proof is by induction on n (which are powers of two).

The base case is when $n = 1$. In this case, the output of algorithm Mystery(a_1) is a_1, which is the minimum of the length-one sequence a_1.

For the induction step, let $n \geq 2$ be a power of two, and assume that the claim is true for the previous power of two, which is $n/2$. Thus, we assume that for any sequence $c_1, c_2, \ldots, c_{n/2}$, algorithm $\textsc{Mystery}(c_1, c_2, \ldots, c_{n/2})$ returns the minimum of the sequence $c_1, c_2, \ldots, c_{n/2}$.

Consider an input sequence $a_1, a_2, \ldots, a_n$ of length n.

1. We see from the pseudocode that the output of algorithm $\textsc{Mystery}(a_1, a_2, \ldots, a_n)$ is the same as the output of algorithm $\textsc{Mystery}(b_1, b_2, \ldots, b_{n/2})$, where $b_i = \min(a_{2i-1}, a_{2i})$.
2. By the induction hypothesis, the output of algorithm $\textsc{Mystery}(b_1, b_2, \ldots, b_{n/2})$ is equal to
$$\min(b_1, b_2, \ldots, b_{n/2}).$$
3. From the way the values b_i are chosen, it is clear that
$$\min(b_1, b_2, \ldots, b_{n/2}) = \min(a_1, a_2, \ldots, a_n).$$
4. It follows that the output of algorithm $\textsc{Mystery}(a_1, a_2, \ldots, a_n)$ is equal to
$$\min(a_1, a_2, \ldots, a_n).$$

This proves the induction step.

Next we consider the function $T(n)$. It follows from the pseudocode that $T(1) = 0$. Let $n \geq 2$ be a power of two. Then $T(n)$ is equal to the sum of

1. $n/2$, which is the number of times that line $(*)$ is executed during the for-loop, before the recursive call, and
2. $T(n/2)$, which is the number of times that line $(*)$ is executed during the recursive call.

Thus, we get the recurrence

$$T(n) = \begin{cases} 0 & \text{if } n = 1, \\ n/2 + T(n/2) & \text{if } n \geq 2 \text{ and } n \text{ is a power of two.} \end{cases}$$

We prove by induction that $T(n) = n - 1$ for any integer $n \geq 1$ that is a power of two: For the base case, i.e., when $n = 1$, we have

$$T(1) = 0 = 1 - 1.$$

For the induction step, if $T(n/2) = n/2 - 1$, then

$$T(n) = n/2 + T(n/2) = n/2 + (n/2 - 1) = n - 1.$$