COMP 2804 — Assignment 2

Due: Sunday October 15, 2023, 11:59 pm.

Assignment Policy:

- Your assignment must be submitted as a single .pdf file. Typesetting (using Latex, Word, Google docs, etc) is recommended but not required. Marks will be deducted for illegible or messy solutions. This includes but is not limited to excessive scribbling, shadows, blurry photos, messy handwriting, etc.
- **No late assignments will be accepted.**
- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.
- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams (which is where most of the marks are).
- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Question 1:

- Write your name and student number.

Question 2: A degree in computer science has 8 elective topics, and students must complete 2 or 3 of them in order to graduate. If 1000 students graduate, show that there is a group of at least 12 students that all completed the same electives.

Solution: The number of different ways for a student to choose electives is (using the sum rule):

$$\binom{8}{2} + \binom{8}{3} = 84. \tag{1}$$

To show there are at least 12 students with the same set of electives, we can use the generalized pigeonhole principle, which states (paraphrased) that the maximum is at least as big as the average. Let $E_{\max}$ be the set of electives that is shared by the most students. For each distinct choice of electives, the average number of students is $1000/84 > 11.9$. That means $E_{\max} > 11.9$. Since we cannot have a fraction of a student, the maximum number of students who took all the same electives $E_{\max} \geq 12$.

Question 3: Consider the following recursive function:

- $f(0) = 8$,
- $f(n) = 2 \cdot f(n-1) - 5n + 3$.

Prove that the closed form is $f(n) = 2^n + 5n + 7$.
Solution: We will prove by induction.

Base Case: $f(0) = 2^0 + 0n + 7 = 8$.
So the base case holds.

Inductive hypothesis: $f(k) = 2^k + 5k + 7, \quad \forall k < n$.
Inductive step:

$$\begin{aligned} f(n) &= 2 \cdot f(n-1) - 5n + 3 \\ &= 2 \cdot (2^{n-1} + 5(n-1) + 7) - 5n + 3 && \text{inductive hypothesis} \\ &= 2^n + 10n - 10 + 14 - 5n + 3 \\ &= 2^n + 5n + 7. \end{aligned}$$

Question 4: The functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N}^2 \to \mathbb{N}$ are recursively defined as follows, where n and m are each a power of 2:

$$\begin{array}{lcll} f(0) & = & 1, & \\ f(n) & = & g(n, f(n-1)) & \text{if } n \geq 1 \\ g(0,n) & = & 0 & \text{if } n \geq 1 \\ g(m,n) & = & g(m-1,n) + 2n & \text{if } m \geq 1 \text{ and } n \geq 1 \end{array}$$

(a) Express $g(m,n)$ as a function of m and n.

Solution: $g(m,n) = 2mn$ for $m \geq 1$.

(b) Prove by induction that your closed form expression for $g(m,n)$ is correct.

Solution:

We prove this by induction on m that $g(m,n) = 2mn$.

Base Case: $m = 0$. We note that by definition $g(0,n) = 0n = 0$. So the base case holds.

Inductive Hypothesis: Assume that for all values $m < k, k > 1$, that $g(m,n) = 2mn$.

Inductive Step: Using the recursive definition, we must show that for $m = k$, we have $g(k, n) = 2kn$.

$$\begin{aligned} g(k,n) &= g(k-1,n) + 2n && \text{by the recursive definition} \\ &= 2(k-1)n + 2n && \text{by the inductive hypothesis} \\ &= 2kn - 2n + 2n \\ &= 2kn. \end{aligned}$$

as required.

(c) Express the recurrence for $f(n)$ in terms of $f(n-1)$ and without any $g(\ldots)$ terms.

Solution: $f(n) = 2nf(n-1)$ for $n \geq 1$ and $f(0) = 1$.

(d) Express $f(n)$ in terms of n.

Solution: After expanding it out a bit we guess that $f(n) = 2^n n!$, $\forall n \geq 0$

(e) Prove by induction that your closed form expression for $f(n)$ is correct.

Solution: We prove this by induction on n.

Base Case: $n = 0$. $f(0) = 2^0 0! = 1$. So the base case holds.

Inductive Hypothesis: Assume that for all values $k < n, n \geq 1$, that $f(k) = 2^k k!$.

Inductive Step: Using the recursive definition and the inductive hypothesis we will show that $f(n) = 2^n n!$.

$$\begin{aligned} f(n) &= 2nf(n-1) && \text{by the recursive definition} \\ &= 2n2^{n-1}(n-1)! && \text{by the inductive hypothesis} \\ &= 2^n n! \end{aligned}$$

as required.

Question 5: Your assignment is due tomorrow so you decide to stream some shows on your laptop. To make sure you finish finish your assignment, you run the following algorithm.

```
Algorithm AssignmentAlgo($n$):

    if $n = 1$:
        Complete one question;
        return;
    else
        Watch $n$ episodes of a show;
        AssignmentAlgo($n/4$);
        AssignmentAlgo($n/4$);
```

You should assume that n is a power of 4.

a) Determine the number of questions that you complete as a function of n. Hint: You can change the base of a logarithm with the following formula (here we change base a to base c):

$$\log_a b = \frac{\log_c b}{\log_c a}$$

Solution: Let $F(n)$ be the number of questions that you complete in the above algorithm on an input of n. We know that $F(1) = 1$. A call to $F(n)$ calls itself recursively twice. We have

$$\begin{aligned} F(n) &= 2 \cdot F(n/4) \\ &= 2 \cdot 2 \cdot F(n/4^2) \\ &= 2^2 \cdot F(n/4^2) \\ &= 2^3 \cdot F(n/4^3) \\ &\dots \\ &= 2^k \cdot F(n/4^k). \end{aligned}$$

The recursion ends at $F(n/4^k) = F(1)$, or when $n/4^k = 1$ or $n = 4^k$. If we take $\log_4$ of both sides we have $k = \log_4 n$. Thus

$$\begin{aligned} F(n) &= 2^{\log_4 n} \cdot F(1) \\ &= 2^{\log_4 n} \\ &= 2^{\frac{\log_2 n}{\log_2 4}} \\ &= 2^{\log_2 n \cdot \frac{1}{2}} \\ &= n^{\frac{1}{2}} \\ &= \sqrt{n}. \end{aligned}$$

So we end up completing $\sqrt{n}$ questions.

b) Determine the number of episodes that you watch as a function of n. You may wish to use

$$\sum_{j=0}^{k-1} ar^j = a\left(\frac{1-r^k}{1-r}\right),$$

where $0 < r < 1$ and a is a real number (this formula gives the closed form of a geometric series).

Solution: Let $T(n)$ be the number of episodes that you watch in the above algorithm for an input of n. Each call to AssignmentAlgo(n) results in you watching n episodes, then it recursively calls itself twice with $\frac{n}{4}$ as a parameter. Let k be the number of times we recursively call AssignmentAlgo. Thus our recursion is:

$$\begin{aligned}
T(n) &= 2\cdot T(n/4) + n \\
T(n) &= 2\cdot\left(2\cdot T(n/4^2) + \frac{n}{4}\right) + n \\
&= 2^2\cdot T(n/4^2) + n\left(\frac{1}{2}\right) + n \\
T(n) &= 2^2\cdot\left(2\cdot T(n/4^3) + \frac{n}{4^2}\right) + n\left(\frac{1}{2}\right) + n \\
T(n) &= 2^3\cdot T(n/4^3) + n\left(\frac{1}{2}\right)^2 + n\left(\frac{1}{2}\right) + n \\
&\dots \\
T(n) &= 2^k\cdot T(n/4^k) + n\left(\frac{1}{2}\right)^{k-1} + n\left(\frac{1}{2}\right)^{k-2} + \cdots + n\left(\frac{1}{2}\right) + n
\end{aligned}$$

This recursion ends when $n/4^k = 1$, or when $n = 4^k$, or (taking the $\log_4$ of both sides) when $k = \log_4 n$. In $T(1)$ we watch no further episodes, thus $T(n/4^k) = T(1) = 0$. Thus the total number of episodes we watch is

$$n\left(\frac{1}{2}\right)^{k-1} + n\left(\frac{1}{2}\right)^{k-2} + \cdots + n\left(\frac{1}{2}\right) + n$$

which we can rewrite as

$$n\cdot\left(1 + \frac{1}{2} + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 + \dots \left(\frac{1}{2}\right)^{k-1}\right).$$

We can use the formula

$$\sum_{j=0}^{k-1} ar^j = a\left(\frac{1-r^k}{1-r}\right)$$

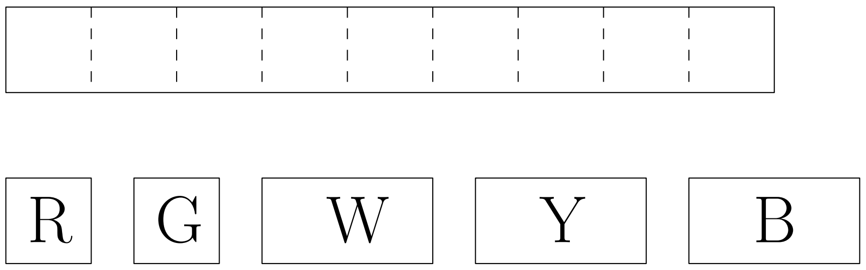
where in our case $a = n$ and $r = \frac{1}{2}$. Thus

$$\begin{aligned}\sum_{j=0}^{k-1} n \cdot \left(\frac{1}{2}\right)^j &= n \cdot \frac{1 - \left(\frac{1}{2}\right)^k}{1 - \frac{1}{2}} \\ &= n \cdot \frac{1 - \left(\frac{1}{2}\right)^{\log_3 n}}{1 - \frac{1}{2}} \\ &= 2n \left(1 - \left(\frac{1}{2}\right)^{\log_4 n}\right) \\ &= 2n \left(1 - \frac{1}{n^{\log_4 n}}\right) \\ &= 2n \left(1 - \frac{1}{\sqrt{n}}\right) \\ &= 2n - 2 \cdot \sqrt{n}.\end{aligned}$$

Therefore you watch $2n - 2 \cdot \sqrt{n}$ episodes over the course of the night.

Question 6:

Let $n \geq 1$ be an integer and consider a $1 \times n$ board B_n consisting of n cells, each one having side length one. The top part of the figure below shows B_9.



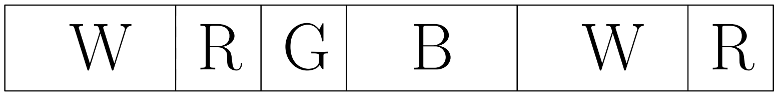
We have an unlimited supply of *bricks* which are of the following types (see the bottom part of the figure above):

- There are red (R) and green (G) bricks which are 1×1 cells.
- There are white (W), yellow (Y), and blue (B) bricks which are 1×2 cells.

A *tiling* of a board B_n is a placement of bricks on the board such that:

- the bricks exactly cover B_n and
- no two bricks overlap.

In a tiling and colour can be used more than once and some colours might not be used at all. The figure below shows one possible tiling of B_9.



Let T_n be the number of different tilings of the board B_n.

(a) Determine T_1 and T_2.

Solution: We can tile any board B_1 using a red or green block. Thus $T_1 = 2$. We can tile any board B_2 one of 7 different ways:

1) 1 white block,
2) 1 blue block,
3) 1 yellow block,
4) 2 red blocks,
5) 2 green blocks,
6) 1 red and 1 green block,
7) 1 green and 1 red block.

Thus $T_2 = 7$.

(b) Express T_n recursively for $n \geq 3$.

Solution: Consider B_n. If we put a red or green block in the first position, then there are T_{n-1} ways to tile the rest of B_n. If we put a white block in the first two positions, then there are T_{n-2} ways to tile the rest of B_n, and the same goes if we put a yellow block or a blue block in the first two positions. Therefore:

$$T_n = 2 \cdot T_{n-1} + 3 \cdot T_{n-2}$$

(c) Prove that for any integer $n \geq 1$,

$$T_n = \frac{1}{4}\left[3^{n+1} + (-1)^n\right].$$

Solution: We will prove by induction. We have $T_1 = 2, T_2 = 7$, and $T_n = 2 \cdot T_{n-1} + 3 \cdot T_{n-2}$. We require two base cases.

$$\begin{aligned} T_1 &= \frac{1}{4}\left[3^{1+1} + (-1)^1\right] \\ &= \frac{1}{4}\left[9 + (-1)\right] \\ &= \frac{1}{4} \cdot 8 \\ &= 2 \end{aligned}$$

as required.

$$\begin{aligned} T_2 &= \frac{1}{4}\left[3^{2+1} + (-1)^2\right] \\ &= \frac{1}{4}\left[3^3 + 1\right] \\ &= \frac{1}{4} \cdot 28 \\ &= 7 \end{aligned}$$

as required. Thus the two base cases hold. For the inductive case we have

$$\begin{aligned} T_n &= 2 \cdot T_{n-1} + 3 \cdot T_{n-2} \\ &= 2 \cdot \frac{1}{4}\left[3^n + (-1)^{n-1}\right] + 3 \cdot \frac{1}{4}\left[3^{n-1} + (-1)^{n-2}\right] \\ &= \frac{1}{4}\left[2 \cdot 3^n + 2 \cdot (-1)^{n-1} + 3 \cdot 3^{n-1} + 3 \cdot (-1)^{n-2}\right] \\ &= \frac{1}{4}\left[2 \cdot 3^n + 3^n + 2 \cdot (-1)^{n-1} + 3 \cdot (-1)^{n-2}\right] \\ &= \frac{1}{4}\left[3 \cdot 3^n + 2 \cdot (-1)^{n-1} + 3 \cdot (-1)^{n-2}\right] \\ &= \frac{1}{4}\left[3^{n+1} + (-1)^n\right] && \text{proof below} \end{aligned}$$

We will break the last equality into 2 cases.

Case 1: n is even. Then $2 \cdot (-1)^{n-1} + 3 \cdot (-1)^{n-2} = 2 \cdot (-1) + 3 = 1 = (-1)^n$, as required.

Case 2: n is odd. Then $2 \cdot (-1)^{n-1} + 3 \cdot (-1)^{n-2} = 2 - 3 = -1 = (-1)^n$, as required.

(d) Show that $T_{2n+2} = T_{n+1}^2 + 3 \cdot T_n^2$. Hint: you can use the closed form from above and prove this, or prove this by induction, but those are both long and painful. It would be easier to prove it combinatorially by describing corresponding tilings.

Solution: Number the locations from left to right as $1..2n+2$. Consider locations $n+1$ and $n+2$. If there is a white, blue, or yellow tile on $n+1$ and $n+2$, then there are T_n ways to tile the first n tiles and T_n ways to tile the last n tiles, and 3 ways to choose the colour of the tile on $n+1$ and $n+2$. Using the product rule that gives us $3 \cdot T_n^2$ possible tilings in this case. If there is not a white, blue or yellow tile spanning locations $n+1$ and $n+2$, then we can break the tiling down into T_{n+1} ways to tile the first $n+1$ locations, and T_{n+1} to tile the last $n+1$ locations, for T_{n+1}^2 ways total. Since those two cases cover all possibilities, $T_{2n+2} = T_{n+1}^2 + 3 \cdot T_n^2$.

(e) Show that $T_{2n+1} = 2 \cdot T_n^2 + 6 \cdot T_n T_{n-1}$.

Solution: If location $n+1$ has a red or green tile, then there are 2 ways to select this tile, T_n ways to tile the first n locations and T_n ways to tile the last n locations for a total of $2 \cdot T_n^2$ ways. If it is a white, blue, or yellow tile on location $n+1$, then there are 2 choices of where to put this tile:

(a) if that tile covers n and $n+1$, then there are T_{n-1} ways to tile the first $n-1$ locations and T_n ways to tile the last n locations.

(b) if that tile covers $n+1$ and $n+2$, then there are T_n ways to tile the first n locations and T_{n-1} ways to tile the last $n-1$ locations.

That gives us 2 choices for the tile location, 3 choices for the colour, and $T_n T_{n-1}$ ways to tile the remaining locations. Thus $T_{2n+1} = 2 \cdot T_n^2 + 6 \cdot T_n T_{n-1}$ as required.

Question 7:

Consider the following recursive algorithm:

```
Algorithm Eu-seful($time, x, y$):

    if $x < time$:
        study math
        Eu-seful($time, 2x + 3y, x$)
    else:
        print("My brain is fried.")
```

Let n be the number of times you *study math* on a call to Eu-seful($time, 2, 1$). We will prove that $n \leq 2 + \log_3 time$, $\forall n \geq 2$. For the inequalities below assume that $n \geq 2$. Use the definition of T_n from the previous question.

(a) ~~Prove that $T_{n-1} \leq time \leq T_n$.~~

Prove that $T_n \leq time \leq T_{n+1}$.

Solution: The recursion ends when $x > time$. If you *study math* n times that means there were $n+1$ recursive calls in total, including the initial call, on a call to Eu-seful($time, 2, 1$) (since on the last call, call $n+1$ your brain is fried and you do not *study math*). We will prove by induction on the number of calls to Eu-seful that the final call is made with parameters $x = T_{n+1}$ and $y = T_n$. We will then use this fact to prove the above.

There are two possible ways to define the base case. If we take $n = 0$ then there is only the initial call Eu-seful($time, 2, 1$), where we have $x = 2 = T_1 = T_{n+1}$ and $y = 1$. If the student defines $T_0 = 1$, then the base case holds. If we use $n = 1$ as the base case, then on the second call to Eu-seful, $x = 2(2) + 3(1) = 7 = T_2 = T_{n+1}$, and $y = 2 = T_1 = T_n$, so the base case holds (I really should have specified to prove this for $n \geq 2$, but c'est la vie).

For our inductive hypothesis, assume that on recursive call n, $n \geq 2$, the arguments were $x' = T_n$ and $y' = T_{n-1}$. We established in Question 6 that $T_{n+1} = 2 \cdot T_n + 3 \cdot T_{n-1}$. Then

$$y = x' = T_n$$

and

$$\begin{aligned} x &= 2 \cdot x' + 3 \cdot y' \\ &= 2 \cdot T_n + 3 \cdot T_{n-1} \\ &= T_{n+1} \end{aligned}$$

as required.

Since we make $n+1$ recursive calls and the recursion ends when $x > time$, and we have shown that on the final call that $x = T_{n+1}$, it must be that $time < T_{n+1}$ (which implies $time \leq T_{n+1}$), which shows the second inequality. We will prove the first inequality ($T_n \leq time$) by contradiction. Thus assume $time < T_n$. Since $x = T_n$ on recursive call n, then $x > time$ and the recursion would have ended after n calls, which is a contradiction to our assumption that there were $n+1$ recursive calls.
As long as the student arguments follows this approach loosely, you should give them the marks.

(b) ~~Prove that $3^{n-2} \leq T_{n-1}$.~~
Prove that $3^{n-2} \leq T_n$.

Solution: The student can prove this using the closed form defined in the previous question, or induction. The closed form should be straightforward, so we will proceed by induction.

- Base Cases:
  - $n = 1$: $3^{-1} = \frac{1}{3} \leq T_1 = 2$, which is correct.

  - $n = 2$: $3^0 = 1 \le T_2 = 7$, which is correct.
  - $n = 3$: $3^1 = 3 \le T_3 = 20$, which is correct.

  Thus the base cases hold. The student should have two of these - if they have only one, deduct 1/2 mark.

- Inductive Hypothesis, $n \ge 4$:
  - $3^{n-3} \le T_{n-1}$ and
  - $3^{n-4} \le T_{n-2}$.

  Then we have:

$$\begin{aligned} 3^{n-2} &= 3 \cdot 3^{n-3} \\ &= 2 \cdot 3^{n-3} + 3^{n-3} \\ &= 2 \cdot 3^{n-3} + 3 \cdot 3^{n-4} \\ &\le 2 \cdot T_{n-1} + 3 \cdot T_{n-2} && \text{by the Inductive Hypothesis} \\ &= T_n \end{aligned}$$

  as required.

(c) Using the inequalities from above, prove that $n \le 2 + \log_3 time$.

Solution: We have $3^{n-2} \le T_n \le time$, thus

$$\begin{aligned} 3^{n-2} &\le time \\ \log_3 3^{n-2} &\le \log_3 time && \text{take } log_3 \text{ of both sides} \\ n - 2 &\le \log_3 time \\ n &\le 2 + \log_3 time \end{aligned}$$

as required.