50.050/50.550 – Advanced Algorithms

January–April Term, 2026

Homework Set 2

Due by: Week 3 Friday (13 February 2026) 1pm.
Please submit your homework online via eDimension.

Note: This homework set focuses on proofs. Remember, you would want your proof to be absolutely correct without any doubt, and not just "sound correct", so think about how to make your proof precise. We have discussed several proofs in Week 2; try to emulate the level of details given in these proofs.

Question 1. Let $a_0, a_1, a_2, \ldots$ be an infinite sequence of integers indexed by $\mathbb{N}$, such that $a_0 = 5$, $a_1 = 4$, and $a_n = 3a_{n-1} + 10a_{n-2}$ for all $n \geq 2$. Prove using induction or a variant of induction that a_n can be computed via the formula

$$a_n = 2(5^n) + 3(-2)^n$$

for all $n \in \mathbb{N}$. [5 marks]

Note: This question is meant to help you practise writing induction arguments. No credit will be given if you do not use induction or a variant of induction. Please state the base case(s) and the induction step very clearly. Please show as much details as possible, and please do not skip steps, otherwise you may not get full credit.

Question 2. Prove that for all non-negative integers $m \leq n$, we have the following identity:

$$\sum_{r=m}^{n} \binom{n}{r}\binom{r}{m} = 2^{n-m}\binom{n}{m}.$$

[5 marks]

Hint: It is up to you to choose whichever proof method you want, but there is an intuitive proof using double counting.

Question 3. There are 25 letter A's and 25 letter B's written on a whiteboard. You may erase any two letters, then write A if they are unequal and write B if they are equal. Do this until there is only one letter remaining on the whiteboard. Is it A or B? Could it be either A or B, depending on how you erased the letter? Or is this last remaining letter always the same? For each possible remaining letter, give an explicit example of how you could get that letter. Justify your answers with a proof, and show as much details as possible. [5 marks]

Question 4. In a magical kingdom, there are 10 castles, and 500 immortal warriors. On Day 1, each of the 500 warriors randomly chooses to defend one of the castles. Every warrior has the option of "relocation", which is to move from defending one castle to defending another castle, subject to the rule that relocation from castle A to castle B is allowed only if after relocating, the number of warriors at castle A would still be at least the number of warriors at castle B. From Day 2 onwards, as long as an allowable relocation is possible for at least one of the 500 warriors, there will always be some warrior who would relocate that day (while following the rule for relocation). Prove that no matter what the initial distribution of warriors was on Day 1, and no matter which castles were selected for relocation (without breaking the rule for relocation), eventually every castle would have exactly 50 warriors. [5 marks]

Question 5. (**For post-graduate students only**[1]) [Loop invariant proofs] To ensure that an algorithm works as intended, you would need to prove the correctness of the algorithm. One common approach for proving correctness is to use a loop invariant proof.

The underlying idea is simple: Suppose your algorithm has a main for-loop or while-loop. If there is a condition P that is true before and after the execution of every iteration of the for-loop/while-loop, then this condition P is called a **loop invariant**. We can use the idea of loop invariants to prove an algorithm is correct, by identifying a suitable condition P, and showing the following:

- **Initialization:** Condition P is true prior to the first iteration of the loop.
- **Maintenance:** If P is true immediately before an iteration of the loop, then it remains true immediately after that iteration.
- **Termination:** When the loop terminates, the conditions for termination, together with P, would imply the desired outcome of the algorithm.

Formally, P is a predicate that depends on the loop variable(s). A loop variable is a variable whose value is updated within the loop. Since every for-loop can be re-expressed as a while-loop, we shall focus on loop invariants for while-loops. For example, consider the following simple algorithm to compute the maximum value of the entries of an integer array $A[1..n]$. (Following notation from the CLRS textbook, we shall write "$A[1..n]$" to mean that A is an array whose first entry is $A[1]$ and whose last entry is $A[n]$; in other words, array A has n entries and the indexing for this array starts from 1.)

```
function maxValue($A$)
    Require: $A[1..n]$ is an integer array
1: $n \leftarrow A$.length
2: $val \leftarrow -\infty$
3: $i \leftarrow 0$
4: while $i < n$ do
5:     $i \leftarrow i + 1$
6:     if $A[i] > val$ then
7:         $val \leftarrow A[i]$
8: return $val$
```

Given an integer array $A[1..n]$, we can compute maxValue(A) to get the maximum value of the entries of A. To see why this is true, let $P = P(i, val)$ be the predicate given by

$$val \text{ equals the maximum value of the entries of the subarray } A[1..i],$$

where the domain of discourse for P is all pairs (i, val) satisfying $i \in \{0, 1, \ldots, n\}$ and $val \in \mathbb{Z}$. (Note that i and val are variables, whose values change with each iteration of the while-loop.) For each $0 \leq j \leq n$, let m_j denote the maximum value of the entries of the subarray $A[1..j]$. Then the predicate $P(i, val)$ is identically the statement "$val = m_i$", whose truth value (i.e. true or false) will depend on the values of val and i.

- **Initialization:** Before the first iteration of the while-loop, the value of i is initialized as $i = 0$, and the value of val is initialized as $val = -\infty$. In particular, $i = 0$ means the subarray $A[1..i]$ is the empty array, so the maximum value of the entries of this empty array is $-\infty$, which is exactly the value of val (prior to the first iteration). Since the proposition $P(0, -\infty)$ is true, we infer that $P(i, val)$ is true prior to the first iteration.

[1]Undergraduate students may try this question, but no marks will be awarded, i.e. this will not count towards your final grade. Only feedback will be given (if any).

- **Maintenance:** Assume that $P(i, val)$ is true immediately before the execution of an iteration of the while-loop, based on the values of i and val at the start of the iteration. Our goal is to show that $P(i, val)$ is true immediately after the execution of this iteration, based on the values of i and val at the end of the iteration.
  - $P(i, val)$ says "$val = m_i$".

  Let $i = k$ and $val = v$ (for some (k, v) in the domain of discourse $\{0, 1, \ldots, n\} \times \mathbb{Z}$) be the values of i and val at the start of the iteration. By assumption, $P(i, val)$ is true at the start of the iteration, which means that proposition $P(k, v)$ is true. Equivalently, we have $v = m_k$, and the proposition $P(k, v)$ is identically the proposition $P(k, m_k)$. By line 5 of the pseudocode, we know that the value of i would become $i = k + 1$ at the end of the iteration. Hence, our goal is to show that the value of val changes from $val = m_k$ at the start of the iteration, to $val = m_{k+1}$ at the end of the iteration.
  Note that for any $1 \leq i \leq n$, the maximum value of the subarray $A[1..i]$ is either the maximum value m_{i-1} of the entries of the subarray $A[1..(i-1)]$, or the value of the entry $A[i]$, whichever is larger. In other words, $m_i = \max\{m_{i-1}, A[i]\}$. Within the iteration, running lines 6 and 7 of the pseudocode would update val as follows:

  $$val \leftarrow \max\{val, A[i]\}.$$

  Since $val = m_k$ at the start of the iteration, we then infer that after updating val, we get $val = m_{k+1}$ at the end of the iteration. Thus, we conclude that $P(i, val) = P(k+1, m_{k+1})$ is true immediately after the execution of this iteration.
- **Termination:** We exit the while-loop after the value of i reaches $i = n$ at the end of some iteration. Let v_{final} be the value of val at the end of the last iteration. Since $P(i, val)$ is true after exiting the while-loop, it means that proposition $P(n, v_{\text{final}})$ is true. Now, proposition $P(n, v_{\text{final}})$ says that $v_{\text{final}} = m_n$, i.e. v_{final} is the maximum value of the entries of the subarray $A[1..n]$. But the subarray $A[1..n]$ is exactly the entire array A. Therefore, the value $val = v_{\text{final}}$ returned in line 6 of the pseudocode is indeed the maximum value of the entries of the input array A.

Additional resources for understanding loop invariants:

- CLRS textbook, Chapter 2.1.
- Youtube video on loop invariant proofs: `https://www.youtube.com/watch?v=_maJ4Qy7Q0E`
- Survey article on examples of loop invariants: `https://arxiv.org/pdf/1211.4470`
- Blog post on proving correctness of algorithms: `https://www.dinalherath.com/2017/Proving-Correctness-of-Algorithms/`

In this question, we shall explore loop invariants. Consider the following algorithm:

```
function FUSE($n, m$)
    Require: $n$ and $m$ are positive integers.
1: Initialise a new empty array $A$
2: $a \leftarrow 2n$
3: $b \leftarrow m$
4: while $a > 1$ do
5:     $a \leftarrow \lfloor \frac{a}{2} \rfloor$
6:     if $a$ is odd then
7:         Append array $A$ with new entry with value $b$
8:     $b \leftarrow 2b$
9: return sum of entries of $A$
```

(i) Given positive integers n and m, let val be the value returned from running FUSE(n, m). Express val in terms of n and m. (To get full credit, your expression must be as simple as possible.) [1 mark]

(ii) Mimicking the style of the loop invariant proof for MAXVALUE, prove that your proposed expression from part (i) is indeed the value returned from the execution of FUSE(n, m). Your proof must explicitly state a loop invariant. [4 marks]

Hint 1: Your loop invariant should be a predicate $P = P(a, b, A)$ that depends on the loop variables a, b and A of the while-loop.
Hint 2: n is a positive integer. What is the binary representation of n?
Hint 3: The predicate $P(a, b, A)$ would be a statement that is true, when variables a, b, A are substituted with the values they have at either the start or the end of each iteration of the while-loop. Be very careful with indices!
Hint 4: Can you find a predicate that involves $\lfloor \log_2(2n) \rfloor - \lfloor \log_2(a) \rfloor$? It is possible to express $P(a, b, A)$ as a "compound" predicate with three parts:

- the first part expresses the sum of entries of A in terms of $\lfloor \log_2(2n) \rfloor - \lfloor \log_2(a) \rfloor$;
- the second part expresses b in terms of $\lfloor \log_2(2n) \rfloor - \lfloor \log_2(a) \rfloor$;
- the third part expresses a in terms of $\lfloor \log_2(2n) \rfloor - \lfloor \log_2(a) \rfloor$.