

IDMA 2026

Problem set 1

Abdulkareem Al-Rifai, kbt185

16-02-26

Contents

1	Provide formal proofs for the following claims	2
1.a	For the sequence $a_1 = 1, a_n = 2 \cdot a_{n-1} + 1, \forall n \geq 2$ prove that $a_n = 2^n - 1, \forall n \in \mathbb{Z}^+$	2
1.b	Prove that it holds for $\forall n \in \mathbb{N}^+ 3 4^n + 5$	2
2	For each of the propositional logic formulas below, determine whether it is a tautology or not. If the formula is not a tautology, show how to add a single connective to make it into a tautology. Please make sure to justify your answers (e.g., by presenting truth tables, or by using rules for rewriting logic formulas that we have learned in class).	3
2.a	$\neg((p \rightarrow q) \vee r) \rightarrow ((\neg q \wedge \neg r) \wedge p)$	3
2.b	$((p \wedge q) \rightarrow r) \leftrightarrow ((q \vee r) \vee \neg p)$	3
3	Your task in this problem is to analyse a modified algorithm by Jakob intended to fix this. The merge part of the algorithm would be essentially the same as before:	4
3.a	Is the pseudocode algorithm above correct in that for any input array A it will be the case that <code>mergerunnsort (A)</code> returns the same array but sorted in increasing order?	4
3.b	Regardless of whether the sorting algorithm is correct or not, does it always terminate (assuming that the input is an array of elements that can be compared with \neq) and, if so, what is the worst-case time complexity?	4
3.c	Can you give any example of a family of input arrays of growing size for which Jakob's merge runs sort algorithm will output a correctly sorted array and be asymptotically faster than the merge sort algorithm that we have covered in class? Can you give any example of a family of input arrays of growing size for which merge runs sort will be asymptotically slower than standard merge sort?	5
4	When constructing this problem set, Jakob ran into some very unfortunate problems. 1. that all swans have the same colour (presumably white, so that there are no black swans after all), and 2. that all positive integers are in fact equal. Both of these claims are fairly disturbing from a mathematical point of view. Please help Jakob by pointing out clearly what goes wrong in his induction proofs below.	5
4.a	All swans have the same colour.	5
4.b	All positive intergers are equal to each other	5

1 Provide formal proofs for the following claims

1.a For the sequence $a_1 = 1, a_n = 2 \cdot a_{n-1} + 1, \forall n \geq 2$ **prove that** $a_n = 2^n - 1, \forall n \in \mathbb{Z}^+$

Using principles of mathematical induction, we start by showing that the first proposition holds :
Let $P(n)$ be the predicate $a_n = 2^n - 1$ Base case $P(1)$:

$$1 = a_1 = 2^1 - 1 = 1(HOLDS) \quad (1)$$

Now we try to prove that $P(k) \implies P(k+1)$ is a tautology

$$P(k) : a_k = 2^k - 1 \quad (2)$$

$$P(k+1) : a_{k+1} = 2^{k+1} - 1 \quad (3)$$

The next term in the sequence (left hand side) is

$$a_{k+1} = 2 \cdot a_k + 1 = 2 \cdot (2^k - 1) + 1 = 2^{k+1} - 1$$

substituting (2) in , we get

$$a_{k+1} = 2 \cdot (2^k - 1) + 1 = 2^{k+1} - 1 \quad (4)$$

We have shown that $a_n = 2^n - 1$ holds for all values $n \in \mathbb{N}^+$

1.b Prove that it holds for $\forall n \in \mathbb{N}^+ \ 3|4^n + 5$

Base case : $P(n_0)$ We start with proving the base case $n_0 = 1$ is true

$$P(n_0) : 4^1 + 5 = 9 \quad (5)$$

$$3|9 : True \quad (6)$$

induction step :

$$P(k) : 3|4^k + 5 \quad (7)$$

let $f(k) = 4^k + 5$

$$P(k+1) : 3|4^{k+1} + 5 \quad (8)$$

The main point here is to manipulate $P(k+1)$ into an expression in terms of $P(k)$

$$P(k+1) : 4^{k+1} + 5 = 4 \cdot 4^k + 5 = 4^k + 4^k + 4^k + 4^k + 5 = 3 \cdot 4^k + (4^k + 5) \quad (9)$$

We can write now that $P(k+1) = f(k) + s, s \in \mathbb{N}$. We have already proved that 3 divides $f(k)$, if we can prove that 3 also divides s , we can use the properties of integers and division and say that if $a|b \wedge a|c$ then $a|b+c$: T $s = 3 \cdot 4^k$, since s is a multiple of 3, we can easily conclude that s is divisible by 3, hence $3|f(k) + s : T$

2 For each of the propositional logic formulas below, determine whether it is a tautology or not. If the formula is not a tautology, show how to add a single connective to make it into a tautology. Please make sure to justify your answers (e.g., by presenting truth tables, or by using rules for rewriting logic formulas that we have learned in class).

2.a $\neg((p \rightarrow q) \vee r) \rightarrow ((\neg q \wedge \neg r) \wedge p)$

To solve this problem I made a truth table, then i realized that it would take alot of effort and time, therefore I decided to use logical thinking; Tautology requires that an expression is true for all the possible scenarios, so we technically only have to find one situation where the expression translates to false to prove that it is not a tautology. An implication of the form $A \rightarrow B$ is false in one scenario only, A:T and B: F. take that $B : ((\neg q \wedge \neg r) \wedge p)$. B is false in the following 4 scenarios.

p	q	r	$\neg q \wedge \neg r$	$((\neg q \wedge \neg r) \wedge p)$
T	F	T	F	F
T	T	F	F	F
F	T	T	F	F
F	F	F	T	F
T	T	T	F	F
T	F	F	T	T

We can now work backward, by taking the values that return a false B, and plugging them in A. The disjunction in A, evaluates to true, if one of the statements is true, hence, we are looking for the one statement in our subset of combinations where A : True, because it will evaluate to true when we apply the negation.

p	q	r	$(p \Rightarrow q)$	$\neg((p \Rightarrow q) \vee r)$
T	T	T	T	F
T	F	T	F	F
T	T	F	T	F
F	T	T	T	F
F	F	F	T	F
T	F	F	F	F

Since all the combination we used for B: false failed to break the argument, I know find the the values where A: True, since A is disjunction, there is only one scenario where that happens (maybe I should have started there), a disjunction is false only when both elements are false. This happens when p:T , q: f and r: false. Plugging those values in the first truth table we get that B: T, and with that we completely fail to break the logic of the expression.

2.b $((p \wedge q) \rightarrow r) \leftrightarrow ((q \vee r) \vee \neg p)$

Using the truth table, a biconditional statement $A \leftrightarrow B$ is only false in 2 scenarios, A:F \wedge B:T \vee A:T \wedge B:F. Take that

$$A : ((p \wedge q) \rightarrow r), B : ((q \vee r) \vee \neg p) \quad (10)$$

A: T, B:F : I am going to list down the possible scenarios where A is true

p	q	$p \wedge q$	r	A
T	T	T	T	T
T	F	F	F	T
F	F	F	F	T
F	T	F	T	T
F	F	F	T	T
		T	F	F (excluded)

Before going any further and using more time on the table, we can now go back to B and check for which scenario does it return false when A is true, the disjunction in B returns false in one scenario, that is, p: T, q: F, r: F. We can see in our unfinished table (row 2), that these values return A: T, and since they return B: F, we can stop here and conclude that this statement/expression is not a tautology. (I still find this method to be time consuming, and somehow luck-based, I hope I can get more feedback on this, and perhaps more feedback on the wording as I still feel unsure about using terms like statements expressions etc..)

3 Your task in this problem is to analyse a modified algorithm by Jakob intended to fix this. The merge part of the algorithm would be essentially the same as before:

3.a Is the pseudocode algorithm above correct in that for any input array A it will be the case that `mergesort(A)` returns the same array but sorted in increasing order?

Yes, The algorithm initiates a while loop that ensures the splitting process is not started as long as $A[i] \leq A[i+1]$, once the while-loop fails, the splitting process on the unsorted right-array begins and it keeps running until $i = n$ (in the right array), when `mergesort(R)` terminates, the merge function initiates and begins to sort the elements by comparing the left and right arrays.

3.b Regardless of whether the sorting algorithm is correct or not, does it always terminate (assuming that the input is an array of elements that can be compared with \leq) and, if so, what is the worst-case time complexity?

Assuming that the array is not of infinite length, then yes, it should terminate after merging the splitted sub-arrays. To find the worst case running time complexity, we have to analyse the cost of each function. The running time of an algorithm can be expressed using the function $T(n)$

-The while-loop checks the elements i -times $O(i)$

-splitting an array that has n elements takes n -times units $O(n)$

-The merge function sorts n elements in array of length n , that takes n -time units $O(n)$

$$T(n) = O(i) + O(n) + O(n - i) \quad (11)$$

In the best case scenario where the array is already sorted and no split-merge is required, the algorithm running time is proportional to the size of the array. So time is a linear function. In the worst case, the while loop terminates at $i = 1$, and the array is split into $L[1:1]$, and $R[n-1]$ and R is then recursively splitted until it is sorted. The time it takes to split the array is $n + (n - 1) + (n - 2) \dots (n - n + 1)$. This is an arithmetic function that can be written in the form $\frac{n(n+1)}{2} = \frac{n^2+n}{2}$

We can now write the time function in the following form

$$T(n) = O(n^2) + O(n) + O(1) = O(n^2) \quad (12)$$

We dropped the lower order term, as they don't effect the running time for large n .

- 3.c Can you give any example of a family of input arrays of growing size for which Jakob's merge runs sort algorithm will output a correctly sorted array and be asymptotically faster than the merge sort algorithm that we have covered in class? Can you give any example of a family of input arrays of growing size for which merge runs sort will be asymptotically slower than standard merge sort?**

An input array that is already sorted (in a growing order) should run faster with Jakobs algorithm. An input array that is sorted in a decreasing order should run faster with the typical mergesort that splits any array into half.

(I wish to get extra feedback on the $T(n)$ function, I still feel weak when it comes to constructing those, so any feedback and explanation is appreciated)

- 4 When constructing this problem set, Jakob ran into some very unfortunate problems. 1. that all swans have the same colour (presumably white, so that there are no black swans after all), and 2. that all positive integers are in fact equal. Both of these claims are fairly disturbing from a mathematical point of view. Please help Jakob by pointing out clearly what goes wrong in his induction proofs below.**

4.a All swans have the same colour.

4.b All positive intergers are equal to each other

For the base case Jakob uses the fact that 1 is equal to itself, hence he uses the equation $n=n$. For the induction step, he uses a different induction hypothesis $n= n-1$. This is counter intuitive. If we work backward and test the base case for the induction hypothesis, we see that the induction hypothesis fails in the base case scenario, which proves that the theorem is wrong. For $n_0 = 1$

$$P(n_0) : n_0 = n_0 - 1 = 1 \neq 1 - 1 \quad (13)$$