CSCI 3104: Algorithms
Spring 2022
Recitation #1 - Induction

SOLUTIONS

Learning Objectives

- know the parts of an inductive proof
- use induction to prove facts about trees
- identify errors in inductive proofs

Problem 1

Your friend excitedly shows you her proof that, for all $n \geq 1$, $\sum_{i=1}^{n}(4i-3) = n(2n-1)$. She tells you that she's pretty sure she's gotten it right. Find any errors in your friend's proof and explain why each is incorrect. You don't need to write a corrected version, only explain the errors.

> **Base Case.** Let $k = 1$. Then, $\sum_{i=1}^{k}(4i-3) = 4 \cdot 1 - 3 = 1$, and $k(2k-1) = 1(2 \cdot 1 - 1) = 1$, and the statement holds.
>
> **Inductive Hypothesis.** Assume the statement holds for all $k \geq 1$.
>
> **Inductive Step.** We show it also holds for $k+1$. By the IH,
>
> $$\sum_{i=1}^{k+1}(4i-3) = (k+1)(2(k+1)-1), \tag{1}$$
>
> so we can conclude that
>
> $$\sum_{i=1}^{k}(4i-3) = k(2k-1) - (4(k+1)-3) \tag{2}$$
>
> $$\sum_{i=1}^{k}(4i-3) = k(2k-1) \tag{3}$$
>
> as claimed.

SOLUTION. There are several errors here:

- Your friend's IH assumes the statement holds **for all** $k \geq 1$, which simply assumes the claim is true and uses the claim to prove itself.
- In the IS, your friend has worked both sides of the equation, which can lead to incorrect results

- In the IS, your friend is working in the wrong direction from the base case even given a correctly stated IH. Say the IH were that the statement holds for $k+1 \geq 1$. We cannot take the first step to expand which values we know this works for: at $k+1=1$, our IS shows that this holds for $k=0$, but not for $k=1$, the direction we are interested in showing.
- You might consider encouraging you friend to explain her work in the IS more thoroughly

A correct proof might be the following:

Base Case. Let $k=1$. Then, $\sum_{i=1}^{k}(4i-3) = 4\cdot 1-3 = 1$, and $k(2k-1) = 1(2\cdot 1-1) = 1$, and the statement holds.

Inductive Hypothesis. Assume the statement holds for some $k \geq 1$.

Inductive Step. We show it also then holds for $k+1$. We write the LHS of the $k+1$st case:

$$\sum_{i=1}^{k+1}(4i-3) = \sum_{i=1}^{k}(4i-3) + 4(k+1) - 3 \tag{4}$$
$$= k(2k-1) + 4k + 1 \tag{5}$$
$$= 2k^2 + 3k + 1 \tag{6}$$
$$= (k+1)(2k+1) = (k+1)(2(k+1)-1), \tag{7}$$

where the IH is used to obtain equation (5) from (4).

Problem 2

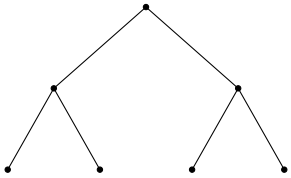
Definition 1. The *complete, balanced binary tree* of depth d for $d \in \mathbb{N}$, denoted $\mathcal{T}(d)$, is defined as follows:

- $\mathcal{T}(0)$ is a single vertex
- for $d > 0$, $\mathcal{T}(d)$ is a root vertex with two children, each the root of $\mathcal{T}(d-1)$.

For example:

$\mathcal{T}(0)$.

$\mathcal{T}(1)$.



$\mathcal{T}(2)$.

Using induction, show that the tree $\mathcal{T}(d)$ has exactly $2^d - 1$ non-leaf nodes.

a. State your induction variable and base case. Show that your base case is correct.

b. State your inductive hypothesis.

c. State your goal for the inductive step. Prove your inductive step. Where did you use your inductive hypothesis?

SOLUTION. Let v_d denote the number of non-leaf vertices in the tree $\mathcal{T}(d)$.

a. We induct on the depth of the tree d, with base case $d = 0$. In this case, $v_0 = 0$ since there are no non-leaf nodes, and $2^0 - 1 = 0$, so the claim holds.

b. Assume the claim holds for some fixed $d \geq 0$, and $v_d = 2^d - 1$.

c. We want to show that, if the claim holds for $\mathcal{T}(d)$, then the claim also holds for $\mathcal{T}(d+1)$, and $v_{d+1} = 2^{d+1} - 1$.

Firstly, note that $v_{d+1} = 2v_d + 1$, since $\mathcal{T}(d+1)$ has all the non-leaf vertices of two trees $\mathcal{T}(d)$, plus a new root (non-leaf) vertex, and all leaf vertices in the subtrees $\mathcal{T}(d)$ are still leaves. Then,

$$\begin{aligned} v_{d+1} &= 2v_d + 1 && (8)\\ &= 2\left(2^d - 1\right) + 1 && (9)\\ &= 2^{d+1} - 2 + 1 = 2^{d+1} - 1, && (10) \end{aligned}$$

where the IH gives us that line (8) can be written as (9).

Problem 3

Recall MERGESORT, which sorts an array $A[1, \ldots, n]$ (here, the notation $A[1, \ldots, n]$ indicates an array of length n with indices $1, \ldots, n$). MERGESORT recursively sorts $A[1, \ldots, n/2]$ and $A[n/2 + 1, \ldots, n]$, and then merges the two sorted sub-arrays into a single sorted array. Pseudocode for MERGESORT is below. You may assume the MERGE function correctly merges the two sorted sub-arrays into a single sorted array. The goal of this problem is to prove the correctness of MERGESORT by induction.

a. Clearly identify the base case(s). Why does MERGESORT correctly sort arrays in these cases?

b. Formulate an inductive hypothesis. [**Note:** Will you want to use a weak inductive hypothesis or strong inductive hypothesis? Try to answer this question and explain your reasoning. Don't hesitate to talk with your classmates or TA if you get stuck!]

c. We now consider the inductive step.

   i. What are the smaller cases that we need to consider? Why?

   ii. How do we know that MERGESORT correctly sorts these smaller cases?

   iii. Once MERGESORT sorts these smaller cases, how do we know that MERGESORT correctly sorts $A[1, \ldots, n]$, the original input array?

d. Using your work in parts (a)-(c), write a proof of correctness for MERGESORT.

Algorithm 1 Mergesort

```
1:  procedure MERGESORT(Array A[1, ..., n])
2:      if len(A) ≤ 1 then return A
3:      Left ← []
4:      Right ← []
5:
6:      for i ← 1; i ≤ len(A)/2; i ← i + 1 do
7:          Left[i] ← A[i]
8:
9:      for i ← len(A)/2 + 1, j ← 1; i ≤ len(A); i ← i + 1, j ← j + 1 do
10:         Right[j] ← A[i]
11:
12:     Mergesort(Left)
13:     Mergesort(Right)
14:     Merge(Left, Right, A)
```

SOLUTION.

a. The base cases are when $n = 0, 1$. Here, MERGESORT returns the array. Note that an array of length $n = 0$ or $n = 1$ is already sorted–there is no additional work that needs to be done.

b. Fix $k \geq 1$. Suppose that MERGESORT correctly sorts any array of length at most k. This is *strong induction*, since we assume MERGESORT correctly sorts anything of length $\leq k$ rather than of lenfth exactly k.

Strong induction is necessary here because MERGESORT on an array of length n requires that MERGESORT be able to successfully sort lists of length approximately $n/2$, not $n-1$.

c. We now consider the inductive step.

i. MERGESORT recursively sorts the `Left` and `Right` sub-arrays. Therefore, these are the two smaller cases we need to consider, each of size approximately $n/2$ (rounded up and down, since if n is not even one list will take the extra element).

ii. MERGESORT correctly sorts these smaller cases because `Left` and `Right` have fewer than k elements. By the inductive hypothesis, MERGESORT correctly sorts both `Left` and `Right`.

iii. Once MERGESORT sorts these smaller cases, how do we know that MERGESORT correctly sorts $A[1, \ldots, n]$, the original input array?

By assumption in the problem statement, the MERGE function correctly merges the two sorted sub-arrays into a single sorted array.

d. Using your work in parts (a)-(c), write a proof of correctness for MERGESORT. We prove by induction on $n \in \mathbb{N}$ the length of the input array A, that MERGESORT correctly sorts the input array.

Base Cases: We consider the cases when $n = 0, 1$. Note that an array of length $n \leq 1$ is already sorted, and so there is no more work to be done. In these cases, MERGESORT (at line 2) returns A and terminates.

Inductive Hypothesis: Fix $k \geq 1$. Suppose that MERGESORT correctly sorts any array of length at most k.

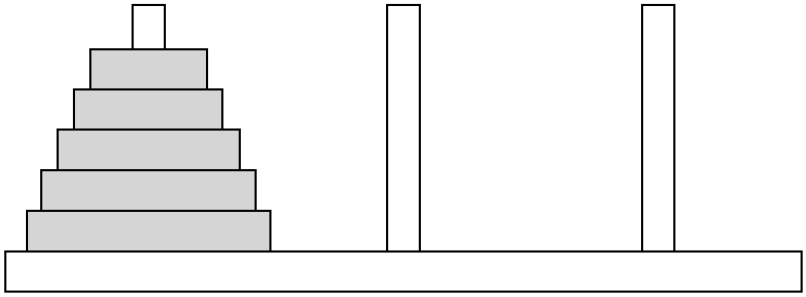
Inductive Step: Suppose $A[1, \ldots, k+1]$ is an array of length $k+1$. At lines 3-10, MERGESORT constructs arrays `Left` and `Right`, where the elements of $A[1, \ldots, (k+1)/2]$ are copied into `Left` in the order they appear in A, and the elements of $A[(k+1)/2, \ldots, k+1]$ are copied into `Right` in the order they appear.

Now at lines 12 and 13, MERGESORT recursively invokes itself, passing `Left` and `Right`, respectively. As `Left` and `Right` have length less than k, we have by the inductive hypothesis that MERGESORT correctly sorts both `Left` and `Right`. So at Line 14, both `Left` and `Right` are sorted. Therefore, by assumption, MERGE correctly merges `Left` and `Right` back into A, so that A is sorted.

The result follows by induction.

Problem 4 (Bonus)

The puzzle game the Tower of Hanoi consists of three upright pegs, and n disks of different diameters which fit over the pegs. The puzzle starts with the disks all stacked on the first peg, with the largest diameter on the bottom decreasing in size to the smallest disk on top, as follows:



The objective of the puzzle is to transfer this "pyramid" to the third peg in the same configuration: largest disk on the bottom up to the smallest on top. Each move consists of moving a single disk from its current peg to a different peg, and disks can only be placed onto either an empty peg or a disk of a larger diameter.

Use induction to show that a Tower of Hanoi with n disks can be solved in $2^n - 1$ moves.

(hint: without loss of generality, if you can move the tower to the third peg in k moves, you can also move the tower to the second peg in k moves)

SOLUTION. We can generalize the problem to be the number of moves to move a tower of n disks to any peg, starting from any peg, assuming there are no smaller disks on either other peg.

We induct on n, the number of disks. Let x_n denote the number of moves it takes to move an n-disk tower to a different peg.

Base Case. When $n = 1$, we can simply pick up the entire tower and move it to the desired peg in one move. Since $2^1 - 1 = 1$, the claim holds.

Inductive Hypothesis. Assume that the claim holds for some fixed height of $n \geq 1$ disks.

Inductive Step. We show we can also move a tower of height $n + 1$ disks to peg 3 in $2^{n+1} - 1$ moves.

Since we can move a tower of n disks to another peg in $v_n = 2^n - 1$ moves, we can move the top n disks to the second peg, move the $n + 1$st disk to the third peg, and then move the n−-disk stack on top of our $n + 1$st peg to move the whole tower. Thus, the complete

tower can be moved to a different peg in

$$\begin{aligned} v_{n+1} &= v_n + 1 + v_n \\ &= 2^n - 1 + 1 + 2^n - 1 \\ &= 2(2^n) - 1 \\ &= 2^{n+1} - 1 \end{aligned}$$

moves, as claimed.