1 Structure of Certain Binary Trees

Problem
In a binary tree all nodes are either internal or they are leaves. In our definition,
internal nodes always have two children and leaves have zero children. Prove
that for such trees, the number of leaves is always one more than the number
of internal nodes.

Solution 1

Proof. Weak induction on \( n \), the number of internal nodes in the tree. In
this proof, let the proposition \( P(n) \) represent the statement that “for all such
binary trees with \( n \) internal nodes, the number of leaves is always
one more than the number of internal nodes”.

Base Case
• Goal: \( P(0) \). There is only one tree with no internal nodes: the lone leaf.
  By inspection, the number of leaves, 1, is one more than the number of
  internal nodes, 0.

Inductive Case
• Goal: for \( n \geq 1 \), \( P(n) \Rightarrow P(n + 1) \).
  Let \( T \) be an arbitrary binary tree with \( n+1 \) internal nodes. Let \( \text{Int}(T) \)
  be number of internal nodes in tree \( T \) and leaf nodes be \( \text{Leaf}(T) \). Thus
  \( \text{Int}(T) = n + 1 \). Select an internal node ‘x’ which is parent of two leaf
  nodes. Since \( n > 1 \), there is at least one internal node. Replace the
  subtree rooted at ‘x’ by a leaf node. The resulting tree \( T' \) has one less
  internal node and one less leaf node than \( T \), i.e.,
  \[
  \text{Int}(T') = \text{Int}(T) - 1 = n + 1 - 1 = n
  \]
  \[
  \text{Leaf}(T') = \text{Leaf}(T) - 1
  \] (1)
Solution 2

Proof. Strong induction on \( n \), the number of internal nodes in the binary tree. In this proof, let the proposition \( P(n) \) represent the statement that “for all such binary trees with \( n \) internal nodes, the number of leaves is always one more than the number of internal nodes”.

Base Case

- Goal: \( P(0) \). There is only one binary tree with no internal nodes: the lone leaf. By inspection, the number of leaves, 1, is one more than the number of internal nodes, 0.

Inductive Case

- Goal: for \( n \geq 1 \), \( \forall_{k<n} P(k) \Rightarrow P(n) \). Given a binary tree with \( n \geq 1 \) internal nodes, we know it has two subtrees with numbers of internal nodes given by \( l \) and \( r \) such that \( l + r + 1 = n \) (all of the internal nodes of the given tree are distributed between the left subtree, the right subtree, and the root). By our inductive hypothesis, \( P(l) \) and \( P(r) \) tell us that the number of leaves in the left and right subtrees (which are smaller than the
given tree) are \( l + 1 \) and \( r + 1 \) respectively. The total number of leaves in the given tree is then their sum \( l + r + 2 \). By our definition, the number of internal nodes in the given tree is \( l + r + 1 \). The total number of leaves is \( (l + r + 1) + 1 \), exactly one more than the number of internal nodes. Thus \( P(n) \) holds.

By induction, the original claim is proven for any tree of the given structure.

**Discussion**

The binding of \( n \) to *internal* nodes was simply our choice. Similar proofs may be found in which \( n \) represents either the number of leaves or the total number of nodes. These proofs will have different sets of base cases, but the intuition behind the inductive case is identical, namely that the number of internal nodes and leaves in the parent is related to those of the children, for which our proposition is already known to hold via the inductive hypothesis.

**2 Set Cardinality**

**Problem**

Prove that for \( n \geq 1 \), the number of subsets of \{1, 2, ..., n\} having an even number of elements is \( 2^{n-1} \). Here 0 counts as an even number.

**Solution**

**Induction Proof**

This can be proved using an interesting inductive argument. Here we induct on \( n \), the number of elements in set. We use weak induction for this purpose. The proposition \( P(n) \) is that “for the set of \( n \) elements, the number of even subsets is \( 2^{n-1} \)”. Note that while this statement is true for the set \{1, 2, ..., n\}, it is also true for any set of \( n \) elements.

**Base Case**

When \( n=1 \), there is \( 2^{1-1} = 2^0 = 1 \), one subset (the null set) with even number of elements. Hence \( P(1) \) is true.

**Inductive Case**

We assume that for an arbitrary value of \( n \geq 1 \), \( P(n) \) is true and our goal is to prove that \( P(n) \Rightarrow P(n + 1) \). Now consider the set \{1, 2, ..., n, n + 1\}. We can divide the subsets of this set into two classes.

Class 1: subsets which do not contain \( n + 1 \).

Class 2: subsets which contain the element \( n + 1 \).
In fact that there is one-to-one correspondence between these two classes. For every subset $S_1$ in class 1 which does not have $n + 1$, we can obtain the corresponding subset $S_2$ in class 2 by adding $n + 1$ to it, i.e., $S_2 = S_1 \cup \{n + 1\}$. So clearly, there are as many subsets in class 2 as there are in class 1. Also note that class 1 is nothing but all the subsets of $\{1, ..., n\}$. By inductive hypothesis, there are $2^{n-1}$ even subsets and $2^{n-1}$ odd subsets in class 1. Also note that for every odd subset in class 1, there is a corresponding even subset in class 2, obtain by adding adding $n + 1$ to the odd set. Thus we have $2^{n-1}$ even subsets in class 2. Adding up, we have $2^{n}$ even subsets in the set $\{1, ..., n + 1\}$.

Using Binomial Theorem  
Recall the binomial theorem.

$$ (x + 1)^n = \sum_{k=0}^{n} \binom{n}{k} x^k 1^{n-1} $$

Note that $\binom{n}{k}$ is the number of ways of choosing $k$ elements from set $\{1, ..., n\}$ or in other words, number of subsets of $k$ length. For example, $\binom{n}{0}$ is number of subsets of size 0 and $\binom{n}{n}$ is the number of subsets of size $n$. To count all the subsets of set $\{1, ..., n\}$, we add the numbers of all subsets of size $k$, where $0 \leq k \leq n$.

$$ \text{number of subsets} = \sum_k \binom{n}{k} $$
This can be evaluated very simply by setting \( x = 1 \) in (2). Thus the total number of subsets = \((2)^n = \sum_k \binom{n}{k}\).

By setting \( x=-1 \), we can see that there are equal number of even and odd subsets.

\[
((-1) + 1)^n = \sum_k \binom{n}{k} (-1)^{n-k} \n\]

\[
0 = \sum_{\{k:0 \leq k \leq n, k \text{ is even}\}} \binom{n}{k} - \sum_{\{k:0 \leq k \leq n, k \text{ is odd}\}} \binom{n}{k} \n\]

\[
\sum_{\{k:0 \leq k \leq n, k \text{ is even}\}} \binom{n}{k} = \sum_{\{k:0 \leq k \leq n, k \text{ is odd}\}} \binom{n}{k} \n\]

In the last expression above, the LHS is count of all even subsets and and RHS is the count all odd subsets. Since there are equal number of even and odd subsets, the number of even subsets is exactly half the total number of subsets, which is \(2^{n-1}\).

### 3 Summation

#### Problem

Prove that for every \( n \geq 0 \), \( \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1} \).

#### Solution using telescoping series

We start by noting that the quantity \( \frac{1}{i(i+1)} = \frac{1}{i} - \frac{1}{i+1} \). Using this in the above summation and expanding, we get

\[
\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{1}{1} - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \ldots + \frac{1}{n-1} - \frac{1}{n} + \frac{1}{n} - \frac{1}{n+1} \n\]

\[
= \frac{1}{1} + \{-\frac{1}{2} + \frac{1}{2}\} + \{-\frac{1}{3} + \frac{1}{3}\} \ldots + \{-\frac{1}{n} + \frac{1}{n}\} - \frac{1}{n+1} \n\]

\[
= \frac{1}{1} - \frac{1}{n+1} \n
= \frac{n}{n+1} \n
\]

Note that other the first and last terms, all terms cancel out with the succeeding term. This kind of series is called a telescoping series. As can be seen here, if a quantity can be represented as a telescoping series, it tremendously simplifies the evaluation task.
Solution using Induction

We induct on \( n \), the number of terms in the summation and use a weak induction for this proof. The assertion \( P(n) \) is that \( \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1} \) for a particular \( n \).

Base Case

With \( n=1 \), we can easily see that \( \frac{1}{(1(2))} = \frac{1}{2} \).

Induction step

We assume that \( \forall n \geq 1, P(n) \Rightarrow P(n+1) \). Now consider the sum for \( n+1 \) terms.

\[
\sum_{i=1}^{n+1} \frac{1}{i(i+1)} = \sum_{i=1}^{n} \frac{1}{i(i+1)} + \frac{1}{(n+1)(n+2)}
\]

\[
= \frac{n}{n+1} + \frac{1}{(n+1)(n+2)} \text{ by induction hypothesis}
\]

\[
= \frac{1}{n+1} \left( n + \frac{1}{(n+2)} \right)
\]

\[
= \frac{1}{n+1} \left( \frac{n^2 + 2n + 1}{(n+2)} \right)
\]

\[
= \frac{1}{n+1} \left\{ \frac{(n+1)^2}{(n+2)} \right\}
\]

\[
= \frac{n+1}{n+2}
\]

4 Algorithm Analysis

Problem

(Problem 2.2-2 on p. 29 of the text): Show the Initialization, Maintenance and Termination part of the loop invariant as was done in the text for Insertion Sort.

Solution

\texttt{SELECTION-SORT(A)}

\begin{verbatim}
  n <- length[A]
  for j <- 1 to n - 1
    do
      smallest <- j
      for i <- j + 1 to n
        do
            then
              smallest <- i
\end{verbatim}
exchange $A[j] \leftrightarrow A[\text{smallest}]$

The loop invariant that is maintained at the start of each iteration of the outer for loop is that **sub array $A[1 \ldots j-1]$ is sorted and contains the set of j-1 smallest elements of the array $A$**. Note that algorithm maintains a sorted section $A[1 \ldots j-1]$ and an unsorted section $A[j \ldots n]$.

**INITIALIZATION**: $j$ is set to 1, so sorted section is empty and loop invariant is trivially true.

**MAINTENANCE**: The sorted section $A[1 \ldots j-1]$ contains the smallest j-1 elements in sorted order i.e, $A[1] \leq A[2] \ldots \leq A[j-1]$. The unsorted section starts at index $j$ and $\forall k, j \leq k \leq n, A[j - 1] \leq A[k]$. The inner loop finds the smallest element in the unsorted section $A[j \ldots n]$ and swaps this value into $A[j]$, thus increasing the size of sorted section by 1 and maintaining the order in the sorted section.

**TERMINATION**: At the end of outer for loop, the sorted section $A[1 \ldots n-1]$ contains n-1 smallest numbers in sorted order. The element $A[n]$ must be the largest element.

Note: The run time for this algorithm is $\Theta(n^2)$ in all the cases, even when the input list is sorted. This follows from the observation that the inner loop is of order $\Theta(n - j)$. Adding up for all the iteration, $\sum_j \Theta(n - j) = \Theta(n^2)$.

### 5 Average Case Analysis for Linear search

**Problem**

(Problem 2.2-3 on p. 29 of the text): Assume that the element being searched is in the list exactly one time and assume it is equally likely in each position

**Solution**

On average, a linear search will search through roughly half of the array elements before it finds the right element. (note that we assume the element is present in the array). This can be seen from the following. Depending on where the search succeeds, the algorithm may search through 1 or 2 ... upto n elements. If the correct element is at position $i$, the algorithm will have done $i$ searches. Each of these is a mutually exclusive case with probability $\frac{1}{n}$. The expected number of searches is $1 \cdot \frac{1}{n} + 2 \cdot \frac{1}{n} + \ldots + n \cdot \frac{1}{n} = \frac{n+1}{2} \cdot \frac{1}{n} = \frac{n+1}{2}$. In the worst case the algorithm will have to search all of the array elements. Thus both average case and worst case are $\Theta(n)$.