CS 473: Fundamental Algorithms, Spring 2013

Discussion 1

January 15, 2013 Version: **1.1**

1.1. Inductive proofs – All horses are of the same color.

Professor Kipod Metorlal[1] had declared that if you have n horses in a room, then they must all be of the same color. This declaration had been met with disbelief in the horse raising community (which know that all their horses have different colors), and Professor Kipod Metorlal had been forced to provide a proof of the claim. Here is the proof:

> *Proof*: The proof is by induction. For the case $n = 1$ we have a single horse in the room, and it has a single color, so the claim is true.
> Next, assume that we have $n + 1$ horses in the room for $n > 0$: $h_1, \ldots, h_{n+1}$. We remove one horse h_1 from the room. We are left with n horses in the room (i.e. $h_2, \ldots, h_{n+1}$), and by induction hypothesis we know that they all have the same color. We now put h_1 back in the room, and remove the horse h_{n+1}. Again, we remain with n horses (i.e., $h_1, \ldots, h_n$) and they all have the same color. It follows that the color of h_1 is identical to the color of $h_2, \ldots, h_n$, and the color of h_{n+1} is identical to the color of $h_2, \ldots, h_n$, and it thus follows that all the horses have the same color. QED. ■

Why is the proof incorrect?

1.2. Inductive proofs –Number of edges in a tree.

The following is an inductive proof of the following claim:

Claim 1.1. *In every tree T, it holds $|E(T)| = |V(T)| - 1$ (i.e a tree with n vertices has $n - 1$ edges).*

Proof: The proof is by induction on $|V(T)|$.
Base case: Base case is when $|V(T)| = 1$. A tree with a single vertex has no edge, so $|E(T)| = 0$. Therefore in this case the formula is true since $0 = 1 - 1$.
Inductive step: Assume that the formula is true for all trees T where $|V(T)| = k$. We will prove that the formula is true for trees with $k + 1$ nodes. A tree T with $k + 1$ nodes can be obtained from a tree T' with k nodes by attaching a new vertex to a leaf of T'. This way we add exactly one vertex and one edge to T', so $|V(T)| = |V(T')| + 1$ and $|E(T)| = |E(T')| + 1$. Since $|V(T')| = k$ by induction hypothesis we have $|E(T')| = |V(T')| - 1$.
Combining the last three relations we have $|E(T)| = |E(T')| + 1 = |V(T')| - 1 + 1 = |V(T)| - 1 - 1 + 1 = |V(T)| - 1$, which means that the formula is true for tree T. ■
Show that the above is *not* a correct inductive proof! You must argue why it is not correct, and in particular produce a tree that the above argument does not cover.

[1] "KIPOD METORLAL" is a phrase in Hebrew which means "deranged Porcupine".

1.3. Coloring.

A k-coloring of a graph G is a labeling $f : V(G) \to S$ from vertices to colors where $|S| = k$. A k-coloring is proper if all adjacent vertices are assigned different colors. A graph is k-colorable if it has a proper k-coloring. Prove that any graph G has a proper $(\Delta+1)$-coloring where Δ is the maximum degree of a vertex of G (no vertex has more than Δ neighbors). For example, any cycle is 3-colorable as $\Delta = 2$ for cycles.

1.4. GCD Algorithms.

Euclid's algorithm for finding the greatest common divisor (gcd) of two non-negative numbers a, b is the following.

```
Euclid($a, b$):
  if ($b = 0$)
    return $a$
  else
    return Euclid($b, a \mod b$)
```

Prove via induction that the algorithm correctly computes the gcd of a, b. Also prove that the running time of the algorithm is polynomial in the input size. Note that the input size is $\Theta(\log a + \log b)$. Assume that the $\mod$ operation along with other basic arithmetic operations take constant time. *Hint:* For both parts think about how $a + b$ is changing in each recursive call.

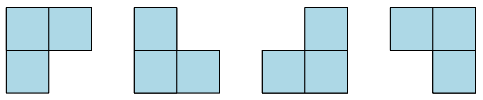
A slow version of the Euclid algorithm is the following.

```
SlowEuclid($a, b$):
  if ($a = 0$) return $b$
  if ($b = 0$) return $a$
  if ($b \geq a$)
    return SlowEuclid($a, b - a$)
  else
    return SlowEuclid($b, a$)
```

Verify for yourself that the above algorithm correctly computes the gcd of a and b. Show that the above algorithm can take exponential time in the input size. You can do this by giving a class of instances $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n), \ldots$ where $\log a_n + \log b_n \to \infty$ and the running time of the algorithm on (a_n, b_n) is exponential in $\log a_n + \log b_n$ (the input size) for each n.

1.5. Tiling.

You are given a $2^n \times 2^n$ chessboard with a single square removed. Prove that you can tile the entire chessboard (minus the missing square) using copies of the 2×2 L's shown on the right

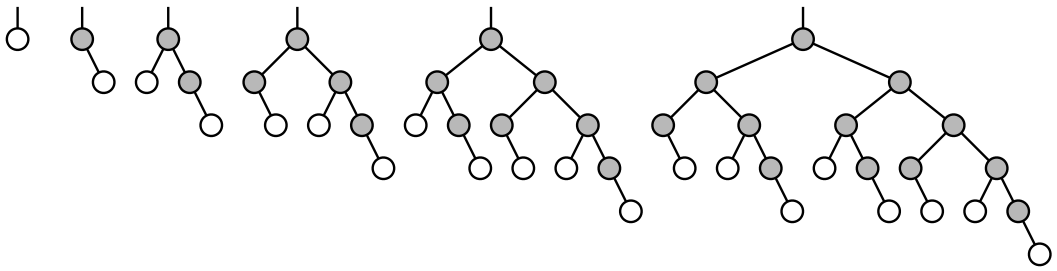


1.6. Fibonacci binary tree.

The nth *Fibonacci binary tree* $\mathcal{F}_n$ is defined recursively as follows:

- $\mathcal{F}_1$ is a single root node with no children.
- For all $n \geq 2$, $\mathcal{F}_n$ is obtained from $\mathcal{F}_{n-1}$ by adding a right child to every leaf and adding a left child to every node that has only one child.

The following figure shows the first six Fibonacci binary trees (i.e., $F_1, F_2, F_3, F_4, F_5, F_6$). In each tree $\mathcal{F}_n$, the subtree of gray nodes is $\mathcal{F}_{n-1}$.



(A) Prove that the number of leaves in $\mathcal{F}_n$ is precisely the nth Fibonacci number: $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$.

(B) How many nodes does $\mathcal{F}_n$ have?

(C) (*) What is the depth of $\mathcal{F}_n$'s most shallow leaf?