List induction

1. State what you’re trying to prove!
2. State and prove the base case (often empty list)
3. Assume it’s true for sublists — inductive hypothesis
4. Show that it holds for the full list
List fact

\[ \text{len} \left( \text{map} \ f \ \text{lst} \right) = \text{len} \ \text{lst} \]

What does this say?
Does it make sense?

List induction

1. State what you’re trying to prove!
2. State and prove the base case (often empty list)
3. Assume it’s true for sublists – inductive hypothesis
4. Show that it holds for the full list

Base case: \( \text{lst} = [] \)
Want to prove: \( \text{len} \left( \text{map} \ f \ [\] \right) = \text{len} \ [\] \)
Proof?

Prove: \( \text{len} \left( \text{map} \ f \ \text{lst} \right) = \text{len} \ \text{lst} \)

\begin{verbatim}
fun len [] = 0
  | len (x::xs) = 1 + len xs
fun map f [] = []
  | map f (x::xs) = (f x) :: (map f xs);
\end{verbatim}

Definition of map!
Prove: \( \text{len}(\text{map} f \text{ lst}) = \text{len} \text{ lst} \)

```plaintext
fun \text{len} [] = 0
  | \text{len} (x::xs) = 1 + \text{len} xs

fun \text{map} f [] = []
  | \text{map} f (x::xs) = (f x) :: (\text{map} f xs);
```

Inductive hypothesis: \( \text{len}(\text{map} f \text{ xs}) = \text{len} \text{ xs} \)
Want to prove: \( \text{len}(\text{map} f (x::xs)) = \text{len} (x::xs) \)

Proof?

\[
\begin{align*}
\text{len}(\text{map} f (x::xs)) &= \text{len}((f x) :: (\text{map} f xs)) & \text{definition of map} \\
&= 1 + \text{len}(\text{map} f xs) & \text{definition of \text{len}} \\
&= 1 + \text{len} (x::xs) & \text{inductive hypothesis} \\
&= \text{len} (x::xs) & \text{definition of \text{len}}
\end{align*}
\]

Done!

Some list “facts”

1. \([\ ]@\text{v} = \text{v}\)
2. \(u@[] = u\)
3. \((u@\text{v})@\text{w} = u@((\text{v}@\text{w}))\)
4. \([u]@u@u = u@u@u\)
Another list fact

\[ \text{len} (\text{xlst} @ \text{ylst}) = \text{len} \text{xlst} + \text{len} \text{ylst} \]

What does this say?
Does it make sense?

Prove:
\[ \text{len} (\text{xlst} @ \text{ylst}) = \text{len} \text{xlst} + \text{len} \text{ylst} \]

1. State what you're trying to prove!
2. State and prove the base case (often empty list)
3. Assume it's true for smaller lists – inductive hypothesis
4. Show that it holds for the current list

Base case: \( \text{xlst} = [] \)
Want to prove: \( \text{len} ([] @ \text{ylst}) = \text{len} [] + \text{len} \text{ylst} \)

Proof?

Base case: \( \text{xlst} = [] \)
Want to prove: \( \text{len} ([] @ \text{ylst}) = \text{len} [] + \text{len} \text{ylst} \)

\[ \text{len} ([] @ \text{ylst}) = \ldots = \text{len} [] + \text{len} \text{ylst} \]

1. Start with left hand side
2. Show a set of justified steps that derive the right hand size

Prove: \( \text{len} (\text{xlst} @ \text{ylst}) = \text{len} \text{xlst} + \text{len} \text{ylst} \)

1. \[ [] @ v1 = v1 \]
2. \[ u1[] = u1 \]
3. \((u1v1)@w1 = u1(v1@w1)\)
4. \[ [u]@us = u::us \]

use induction on \( \text{xlst} \)
Prove: \( \text{len}(\text{xlst} @ \text{ylst}) = \text{len}(\text{xlst}) + \text{len}(\text{ylst}) \)

**Base case:** \( \text{xlst} = [] \)

Want to prove: \( \text{len}([] @ \text{ylst}) = \text{len}([]) + \text{len}(\text{ylst}) \)

\[
\begin{align*}
\text{len}([]) @ \text{ylst} &= \text{len}(\text{ylst}) \quad \text{fact 1} \\
&= 0 + \text{len}(\text{ylst}) \quad \text{math} \\
&= \text{len}([]) + \text{len}(\text{ylst}) \quad \text{definition of len}
\end{align*}
\]

Prove: \( \text{len}(\text{xlst} @ \text{ylst}) = \text{len}(\text{xlst}) + \text{len}(\text{ylst}) \)

**Inductive hypothesis:** \( \text{len}(\text{xs} @ \text{ylst}) = \text{len}(\text{xs}) + \text{len}(\text{ylst}) \)

Want to prove: \( \text{len}((\text{x}::\text{xs}) @ \text{ylst}) = \text{len}((\text{x}::\text{xs})) + \text{len}(\text{ylst}) \)

Prove: \( \text{len}(\text{xlst} @ \text{ylst}) = \text{len}(\text{xlst}) + \text{len}(\text{ylst}) \)

Want to prove: \( \text{len}((\text{x}::\text{xs}) @ \text{ylst}) = \text{len}((\text{x}::\text{xs})) + \text{len}(\text{ylst}) \)

\[
\begin{align*}
\text{len}((\text{x}::\text{xs}) @ \text{ylst}) &= \text{len}(\text{x}::\text{xs}) + \text{len}(\text{ylst}) \\
&= \text{len}(\text{x}::\text{xs}) + \text{len}(\text{ylst}) \\
&= \text{len}(\text{x}::\text{xs}) + \text{len}(\text{ylst}) \\
&= \text{len}(\text{x}::\text{xs}) + \text{len}(\text{ylst})
\end{align*}
\]
Want to prove: \( \text{len} ((\text{x} :: \text{xs}) @ \text{ylst}) = \text{len} (\text{x} :: \text{xs}) + \text{len} \text{ylst} \)

\[
\text{len} ((\text{x} :: \text{xs}) @ \text{ylst}) = \quad ? \quad = \text{len} (\text{x} :: \text{xs}) + \text{len} \text{ylst}
\]

Inductive hypothesis: \( \text{len} (\text{xs} @ \text{ylst}) = \text{len} \text{xs} + \text{len} \text{ylst} \)

Want to prove: \( \text{len} ((\text{x} :: \text{xs}) @ \text{ylst}) = \text{len} (\text{x} :: \text{xs}) + \text{len} \text{ylst} \)

1. [\text{v}1] = \text{v}1
2. \text{u}1[\text{v}1] = \text{u}1
3. (\text{u}1\text{v}1)@\text{v}1 = \text{u}1(\text{v}1\text{v}1)
4. (\text{u}1\text{v}1) = \text{u}1::\text{v}1

\[
\text{fun} \quad \text{len} [] = 0
\quad \text{len} (\text{x} :: \text{xs}) = 1 + \text{len} \text{xs}
\]

Blast from the past

\[
\text{fun cart} \quad [] = []
\text{cart} (u :: u) \text{v}1 = (\text{map} (\text{fn} \ \text{x} => (u,\text{x})) \text{v}1) \ @ \ (\text{cart} \ \text{us} \ \text{v}1);
\]

What does the anonymous function do?

Takes a value, \( x \), and creates a tuple with \( u \) as the first element and \( x \) as the second.
What does the map part of this function do?

What is the type signature?

What does this function do?

fun cart (u::us) :: vl = (map (fn x => (u,x)) vl) @ (cart us vl);

What is the type signature?

fun cart (u::us) :: vl = (map (fn x => (u,x)) vl) @ (cart us vl);

4. [2 points] Write a function cartesian that takes two lists and forms a list of all the ordered pairs, with one element from the first list and one from the second. For example, cartesian [(1,3)] [(2,4)] will return [(1,2),(1,3),(1,4),(3,2),(3,4),(3,3),5,2,5,4)].

cartesian : 'a list -> 'b list -> ('a * 'b) list
A property of cart

fun cart [] [] = []
  | cart (u::us) vl = (map (fn x => (u,x)) vl) @ (cart us vl);

len(cart ul vl) = (len ul) * (len vl)

What does this say?
Does it make sense?

Proof by induction. Which variable, ul or vl?
Base case: \( \text{ulist} = [] \)
Want to prove: \( \text{len} (\text{cart} [] \text{vl}) = (\text{len} []) \times (\text{len} \text{vl}) \)

Proof?

Prove: \( \text{len}(\text{cart} \text{ul} \text{vl}) = (\text{len} \text{ul}) \times (\text{len} \text{vl}) \)

Inductive hypothesis: \( \text{len} (\text{cart us vl}) = (\text{len} \text{us}) \times (\text{len} \text{vl}) \)
Want to prove: \( \text{len} (\text{cart (us::us) vl}) = (\text{len} (\text{us::us})) \times (\text{len} \text{vl}) \)

Prove: \( \text{len}(\text{cart ul vl}) = (\text{len} \text{ul}) \times (\text{len} \text{vl}) \)
Want to prove: \[ \text{len} \left( \text{cart} \left( \text{us} :: \text{us} \right) \text{vl} \right) = \text{len} \left( \text{us} \right) \ast \text{len} \left( \text{vl} \right) \]

\[
\begin{align*}
\text{len} \left( \text{cart} \left( \text{us} :: \text{us} \right) \text{vl} \right) &= \text{len} \left( \text{map} \left( \text{fn} \ x \rightarrow (u,x) \right) \text{vl} \right) @ \left( \text{cart us vl} \right) \\
&= \text{len} \left( \text{map} \left( \text{fn} \ x \rightarrow (u,x) \right) \text{vl} \right) + \text{len} \left( \text{cart us vl} \right) \text{''map'' fact} \\
&= \text{len} \left( \text{vl} \right) + \text{len} \left( \text{cart us vl} \right) \text{''map'' fact} \\
&= \text{len} \left( \text{vl} \right) + \left( \text{len us} \right) \ast \left( \text{len vl} \right) \text{ \text{IH}} \\
&= \left( 1 + \left( \text{len us} \right) \right) \ast \left( \text{len vl} \right) \text{ \text{IH}} \\
&= \left( \text{len us} \right) \ast \left( \text{len vl} \right) \text{ \text{IH}} \\
\end{align*}
\]

Quick refresher: datatypes

- **datatype** direction = North | South | East | West;
- **datatype** student = Freshmen of string | Sophomore of string | Junior of string | Senior of string;
- **datatype** pos = Pos of int list | Zero | Neg of int list;

Recursive datatype

- **datatype** 'a binTree = Empty | Node of 'a binTree * 'a * 'a binTree;

What is this?
Recursive datatype

```haskell
datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;
```

What does this look like?

Node(Empty, 1, Empty);

Recursive datatype

```haskell
datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;
```

Node(Empty, 1, Empty);

What does this look like?

Recursive datatype

```haskell
datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;
```

Node(Empty, 1, Empty);

What does this look like?
Recursive datatype

datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;
Node(Node(Empty, 3, Node(Empty, 4, Empty)), 5, Node(Empty, 9, Empty));

What does this look like?

Recursive datatype

datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;
Node(Node(Empty, "apple", Node(Empty, "banana", Empty)),
  "carrot",
  Node(Empty, "rubarb", Empty));

Facts about binary trees

datatype 'a binTree =
  Empty
| Node of 'a binTree * 'a * 'a binTree;

Counting elements in a tree N( ):
N(Empty) =

How many Nodes (i.e. values) are in an empty binary tree?
Facts about binary trees

```
datatype 'a binTree =
    Empty
  | Node of 'a binTree * 'a * 'a binTree;
```

Counting elements in a tree \( N() \):

\[
N(Empty) = 0
\]

\[
N(Node(u, elt, v)) = 1 + N(u) + N(v)
\]

One element stored in this node plus the nodes in the left tree and the nodes in the right tree

---

Leaves

A “leaf” is a Node at the bottom of the tree, i.e.
\( \text{Node}(\text{Empty}, \text{elt}, \text{Empty}) \)

```
Node(Node(Empty, 3), Node(Empty, 4), Node(Node(Empty, 9), Node(Empty, Empty), Empty));
```

Which are the leaves?
A "leaf" is a Node at the bottom of the tree, i.e.
Node(Empty, elt, Empty)

Node(Node(Empty, 3, Node(Empty, 4, Empty)), 5, Node(Empty, 9, Empty));

**Facts about binary trees**

**Counting leaves in a tree L()**:
- $L(\text{Empty}) = 0$
- $L(\text{Empty, elt, Empty}) = 1$
- $L(\text{Node}(u, elt, v)) = L(u) + L(v)$
Facts about binary trees

datatype 'a binTree =
  Empty
  | Node of 'a binTree * 'a * 'a binTree;

Counting Empty in a tree E( ):

\[ E(\text{Empty}) = 1 \]
\[ E(\text{Node}(u, elt, v)) = E(u) + E(v) \]

Notation summarized

- \( N(\cdot) \): number of elements/values in the tree
- \( L(\cdot) \): number of leaves in the tree
- \( E(\cdot) \): number of Empty nodes in the tree

Tree induction

1. State what you’re trying to prove!
2. State and prove the base case(s) (often Empty and/or Leaf)
3. Assume it’s true for smaller subtrees – inductive hypothesis
4. Show that it holds for the full tree

\[ N(\text{Empty}) = 0 \]
\[ N(\text{Node}(u, elt, v)) = 1 + N(u) + N(v) \]

\[ E(\text{Empty}) = 1 \]
\[ E(\text{Node}(u, elt, v)) = E(u) + E(v) \]

\[ L(\text{Empty}) = 0 \]
\[ L(\text{Node}(u, elt, Empty)) = 1 \]
\[ L(\text{Node}(u, elt, v)) = L(u) + L(v) \]

What is this saying in English?

\[ N(t) = E(t) - 1 \]
Base case: $t = \text{Empty}$
Want to prove: $N(\text{Empty}) = E(\text{Empty}) - 1$

Proof:

$N(\text{Empty}) = 0$
$N(\text{Node}(u, elt, v)) = 1 + N(u) + N(v)$
$E(\text{Empty}) = 1$
$E(\text{Node}(u, elt, v)) = E(u) + E(v)$
$L(\text{Empty}) = 0$
$L(\text{Node}(u, elt, v)) = L(u) + L(v)$

$N(t) = E(t) - 1$
Number of nodes/values is equal to number of Emptyys minus one

Sanity check: is it right here?

Base case: $t = \text{Empty}$
Want to prove: $N(\text{Empty}) = E(\text{Empty}) - 1$

Proof:

$N(\text{Empty}) = 0$
$N(\text{Node}(u, elt, v)) = 1 + N(u) + N(v)$
$E(\text{Empty}) = 1$
$E(\text{Node}(u, elt, v)) = E(u) + E(v)$
$L(\text{Empty}) = 0$
$L(\text{Node}(u, elt, v)) = L(u) + L(v)$

$N(t) = E(t) - 1$
Number of nodes/values is equal to number of Emptyys minus one

4 nodes = 5 Emptyys - 1
Inductive hypotheses: 
- $N(u) = E(u) - 1$
- $N(v) = E(v) - 1$ (Relation holds for any subtree)

Want to prove: $N(\text{Node}(u, elt, v)) = E(\text{Node}(u, elt, v)) - 1$

**Prove: $N(t) = E(t) - 1$**

<table>
<thead>
<tr>
<th>$N(\text{Empty})$</th>
<th>$0$</th>
<th>$N(\text{Node}(u, elt, v)) = 1 + N(u) + N(v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(\text{Empty})$</td>
<td>$1$</td>
<td>$E(\text{Node}(u, elt, v)) = E(u) + E(v)$</td>
</tr>
<tr>
<td>$L(\text{Empty})$</td>
<td>$0$</td>
<td>$L(\text{Node}(u, elt, v)) = L(u) + L(v)$</td>
</tr>
</tbody>
</table>

Want to prove: $N(\text{Node}(u, elt, v)) = E(\text{Node}(u, elt, v)) - 1$

<table>
<thead>
<tr>
<th>$N(u)$</th>
<th>$E(u) - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(v)$</td>
<td>$E(v) - 1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N(\text{Empty})$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(\text{Node}(u, elt, v)) = 1 + N(u) + N(v)$</td>
<td></td>
</tr>
<tr>
<td>$E(\text{Empty})$</td>
<td>$1$</td>
</tr>
<tr>
<td>$E(\text{Node}(u, elt, v)) = E(u) + E(v)$</td>
<td></td>
</tr>
<tr>
<td>$L(\text{Empty})$</td>
<td>$0$</td>
</tr>
<tr>
<td>$L(\text{Node}(u, elt, v)) = L(u) + L(v)$</td>
<td></td>
</tr>
</tbody>
</table>

Other interesting tree facts
Summary of induction proofs

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

Recurrence relations:
\[ \text{count},(k) = \frac{k(k+1)}{2} \]
\[ \text{count},(k) = 2^{\text{count}}, - k = 2 \]

Code equivalence:
\[ \text{fibrec}(n) = \text{fibiter}(n) \]
\[ \text{len(map f xlst)} = \text{len} xlst \]
\[ \text{len[xlst @ ylst]} = \text{len} xlst + \text{len} ylst \]
\[ \text{len(cart ul vl)} = \text{len} ul * \text{len} vl \]

Induction on lists:
\[ \text{N(t)} = E(t) - 1 \]

Outline for a “good” proof by induction

1. Prove: what_to_prove
2. Base case: the_base_case(s)
   step by step proof
   with each step clearly justified
3. Assuming: the_inductive_hypothesis
4. Show: what_you’re_trying_to_prove
   step by step proof
   with each step clearly justified

Be careful!

I believe there is one true soul mate for every person.

Me Must Be Very Busy

I Mean One Per Person

Your Line Would Be Stupid

Can Your Soul Mate Be A Monkey?