Proving Recursive Programs Correct.

Induction is the way to prove that a recursive program is correct. In this lecture we consider a couple of examples, and in the UGP there is another example.

1. Factorial.

```plaintext
1: procedure FACT(n) ▷ Assume n ∈ N.
2:   if n = 1 then:
3:     return 1
4:   else:
5:     return n·FACT(n − 1)
```

We now prove the following

**Theorem 1.** For all positive integers n, FACT(n) returns n!

*Proof.* We prove by induction the statement ∀n ∈ N : P(n), where P(n) is the predicate “FACT(n) returns n!”.

**Base Case:** Let us verify P(1). By definition of the factorial function, 1! = 1. Now, if n = 1, then Line (3) returns 1. Thus, the base case is verified; P(1) is indeed true.

**Inductive Case:** Let us now assume for a fixed k ∈ N that P(k) is true. That is FACT(k) indeed returns k!. We need to prove P(k + 1), that is, we need to prove FACT(k + 1) returns (k + 1)!. Inspecting Line (5), we see that FACT(k + 1) returns (k + 1) times the number returned by FACT(k). By the induction hypothesis, the latter number is k!. Therefore, FACT(k + 1) returns (k + 1) · k! = (k + 1)!. Therefore, the inductive case is true, and so by induction, the theorem is proved.

2. Binary Search
We say that BINSEARCH works properly on the input \((A, x)\) if it return true when \(x \in A\), and returns false otherwise. The correctness of BINSEARCH amounts to proving the following theorem.

**Theorem 2.** For any sorted array of numbers \(A\) and any number \(x\), BINSEARCH works properly on \((A, x)\).

**Proof.** Let \(P(n)\) be the predicate which is true if for any sorted array \(A\) of length \(n\) and any \(x\), BINSEARCH works properly on the input \((A, x)\). We wish to prove \(\forall n \in \mathbb{N} : P(n)\). We proceed by induction.

**Base Case.** Is \(P(1)\) true? That is, given any sorted array \(A\) of length 1 (that is, containing exactly one element), and any \(x\), does BINSEARCH work properly on \((A, x)\)? To answer this, let us fix a sorted array \(A\) and a number \(x\). If \(x\) was indeed in the array \(A\), then it must be that \(x = A[1]\). Line 6 then tells us that in this case BINSEARCH does return true. Similarly, if \(x\) was not in the array \(A\), then \(x \neq A[1]\). Line 8 then tells us that in this case BINSEARCH does return false. Thus, in both cases the algorithm behaves properly. \(P(1)\) is thus established to be true.

**Inductive Case.** Fix a natural \(k \in \mathbb{N}\). The (strong) induction hypothesis is that \(P(1), P(2), \ldots, P(k)\) are all true. We now need to prove \(P(k+1)\). For brevity’s sake, let us call \(N := k + 1\); we wish to prove \(P(N)\), and \(P(a)\) is true for all \(a < N\). And we have \(N > 1\).

That is, we need to show given any sorted array \(A\) of length \(N\), and any \(x\), BINSEARCH works properly on \((A, x)\). To this end, let us fix a sorted array \(A\) of length \(N\) and an \(x\).

**Case 1:** \(x \notin A\). In this case, the algorithm should return false. Since \(x \notin A\), \(x \neq A[m]\). Thus, Line 11 does not run. Furthermore, \(x \notin A\) implies \(x \notin A[1 : m]\) and \(x \notin A[m + 1 : N]\). Since both the arrays \(A[1 : m]\) and \(A[m + 1 : N]\) have lengths \([N/2]\) and \([N/2]\) which are \(< N\) for all

```
1: procedure BINSEARCH(A, x) ▷ Assume A is sorted strictly increasing.
2: ▷ Returns true if x ∈ A, otherwise returns false.
3: n ← A.length.
4: if n = 1 then:
5: if x = A[1] then:
6: return true.
7: else:
8: return false.
9: else:
10: m = \([n/2]\).
11: if x = A[m] then:
12: return true.
13: else if x < A[m] then:
14: return BINSEARCH(A[1 : m], x).
15: else: ▷ That is, x > A[m]
16: return BINSEARCH(A[m + 1 : n], x).
```
$N > 1$, by the (strong) induction hypothesis we have that both\ \textsc{BinSearch}(A[1:m],x)\ and\ \textsc{BinSearch}(A[m+1:N],x)\ return\ false.\ Thus,\ no\ matter\ which\ of\ Line\ 14\ or\ Line\ 16\ runs,\ the\ algorithm\ \textsc{BinSearch}(A[1:N],x)\ will\ return\ false.\ Thus,\ in\ this\ case,\ the\ algorithm\ works\ properly.

\textbf{Case 2:} $x \in A$. In this case, there is some $1 \leq j \leq N$ such that $x = A[j]$. If $j = m$, then Line 11 will return true. If $j < m$, then \textit{since the array is sorted} $x = A[j] < A[m]$. Thus, Line 14 will run. Since $x \in A[1:m]$ and $m = \lceil N/2 \rceil < N$, by the (strong) inductive hypothesis, we know that \textsc{BinSearch}(A[1:m],x) will return true. If $j > m$, then \textit{since the array is sorted} $x = A[j] > A[m]$. Thus, Line 16 will run. Since $x \in A[m+1:N]$ whose length is $\lceil N/2 \rceil < N$, by the (strong) inductive hypothesis, we know that \textsc{BinSearch}(A[m+1:N],x) will return true.

\hfill \Box

\begin{center}
\textbf{Minimal Counterexample: A Different look at Induction}
\end{center}

There is a different, and equivalent, at looking at mathematical induction proofs which, at times, may be more suitable. This is more of a “proof by contradiction” viewpoint. One assumes the assertion is false, picks the \textit{minimal counterexample} to the statement, and then tries to argue a contradiction. To make things concrete, let is give a “different” proof of something we saw in class.

\begin{center}
\textbf{Theorem 3.} Every natural number $\geq 2$ can be written as a product of primes and 1.
\end{center}

\textit{Proof.} Suppose not. Let $n$ be the minimal counter example to the statement, that is, it is \textit{smallest} number which \textit{cannot} be written as a product of primes and 1. Then $n$ cannot be a prime, for a prime is a product of primes and 1. So, $n = a \times b$ for two numbers $a$ and $b$ which are $< n$. Since $n$ is the \textit{minimal counter example}, both $a$ and $b$ can be expressed as a product of primes and 1. And thus, so can $n$ which is a contradiction to $n$ being a counterexample.

Indeed, the above is the same proof. But the mental image one has can differ. Let’s give another example. In the UGP, you are asked to prove this by induction.

\begin{center}
\textbf{Theorem 4.} Suppose a finite number of players play a round-robin tournament, with everyone playing everyone else exactly once. Each match has a winner and a loser (no ties). We say that the tournament has a cycle of length $m$ if there exist $m$ distinct players $(p_1, p_2, \ldots, p_m)$ such that $p_1$ beats $p_2$, $p_2$ beats $p_3$, …, $p_{m-1}$ beats $p_m$, and $p_m$ beats $p_1$. Clearly this is possible only for $m \geq 3$. If a tournament has at least one cycle, then it has a cycle of length exactly 3.
\end{center}

\textit{Proof.} Let us consider a tournament with a cycle, and consider among all cycles in the tournament, any one with the smallest length. Let this be $C = (p_1, p_2, \ldots, p_m)$ with length $m$. If $m = 3$, we are done. Therefore, suppose, for contradiction’s sake, $m > 3$. Now consider the players $p_1$ and $p_3$. Since there are no ties, either $p_1$ beats $p_3$ or $p_3$ beats $p_1$. If $p_3$ beats $p_1$, then $(p_1, p_2, p_3)$ is a shorter cycle (indeed its length is 3). If $p_1$ beats $p_3$, then $(p_1, p_3, p_4, \ldots, p_m)$ is a shorter cycle of length $m - 1$. This contradicts that $C$ was a \textit{smallest cycle}. Thus, $m = 3$. \hfill \Box
The Well-Ordering Principle and PMI

What we have used before, implicitly and rather matter-of-fact-ly, is the following *axiom* called the well-ordering principle (WOP).

Any non-empty subset $S \subseteq \mathbb{N}$ has a minimum element $x \in S$.  

(WOP)

An element $x \in S$ is minimum if for all $y \in S \setminus x$, we have $x < y$.

**Remark:** Note that $S$ needs to be non-empty. More importantly, note that if $S \subseteq \mathbb{Z}$, then the above statement is false; consider the set $S$ to be of all negative integers. Finally, note if $S \subseteq \mathbb{Q}^+$, that is, if it is a subset of positive rationals, then the statement would be false too. Indeed, let $S$ be the set of all rationals strictly greater than 0. Do you see why $S$ doesn’t have a minimum?

In both the above applications, we have used this principle on a subset generated by the counterexamples. In the prime factorization example, $S$ was the subset of numbers which cannot be written as a product of primes and 1. In the tournament example, $S$ was the lengths of the smallest cycles in tournaments which have cycles but none of length 3. The fact that $S$ was not empty was assumed for contradiction’s sake. And then the minimal element was used for obtaining a contradiction.

Let us end by showing that the WOP can be used to *prove* the principle of mathematical induction (PMI). Recall, the principle of mathematical (strong) induction (PMI) states that

**Theorem 5 (Induction).** Given predicates $P(1), P(2), P(3), \ldots$, if

- $P(1)$ is true (base case); and
- For all $k \in \mathbb{N}$, $(P(1) \land P(2) \land \cdots \land P(k)) \Rightarrow P(k + 1)$ (inductive case);

then, $\forall n \in \mathbb{N} : P(n)$ is true.

**Proof.** Suppose not. That is, the base case and the inductive case holds, but $P(n)$ is false for some non-negative integer $n$. Indeed, let $S \subseteq \mathbb{N}$ be the subset of non-negative integers $n$ for which $P(n)$ is false. By our supposition, $S$ is non-empty. Therefore, by WOP, $S$ has a minimal element $x$.

Now $x > 1$ because $P(1)$, as we know by the base-case, is true. Thus the set $\{1, 2, \ldots, x - 1\}$ is not empty. Furthermore, since $1, 2, \ldots, x - 1$ are all strictly $< x$, and $x$ is the minimum element of $S$, none of these elements can be in $S$. Therefore, $P(1), P(2), \ldots, P(x - 1)$ are all true. Thus, $P(1) \land \cdots \land P(x - 1)$ is true. The inductive case then implies $P(x)$ is true. But this contradicts the fact that $x \in S$. Thus our supposition is false, and hence PMI is true. \[\Box\]