Feb 2, 2011

A simple Inequality example:

Thm: $\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots\frac{1}{2^n}<1$

Pf: we want to prove the thm by Induction. We assume it is true for n, we prove it for $n+1$. However we know by induction hypothesis that the sum of the first n terms is less than 1 but when we add $\frac{1}{2^{n+1}}$ it can go more than 1. The trick is too use induction in a different order.

look at the last n terms $\frac{1}{4}+\frac{1}{8}+\cdots\frac{1}{2^n}+\frac{1}{2^{n+1}}=\frac{1}{2}\left[\frac{1}{2}+\frac{1}{4}+\cdots\frac{1}{2^n}\right]<1$ by induction hypothesis. now if we add $\frac{1}{2}$ to both sides, we get the desired result.

Arithmetic versus Geometric Mean thm:

Thm: if $x_1, x_2, \cdots x_n$ are all positive numbers, then

$$(x_1 x_2 \cdots x_n)^{\frac{1}{n}} \le \frac{x_1+x_2+\cdots+x_n}{n} \quad (\bigstar)$$

Pf: the proof in by induction on n. the induction hypo. is the same as the statement however, it is not as usual, we are using <u>reversed induction principle:</u>

If a statement P is true for an infinite subset of the natural numbers and if its truth for n implies its truth for $n-1$, then P is true for all natural numbers.

(It is correct since for an infinite set, for every natural number k, there is a greater number m in the set and we can use the reverse induction step to go backward from m to k.

First we show it is correct for an infinite subsets (i.e. for all powers of 2) using regular induction

For $n=1$ it is trivial and for $n=2$, we have $\sqrt{x_1x_2}\le\frac{x_1+x_2}{2}$, which is correct since $4x_1x_2\le(x_1+x_2)^2$ (since $x_1^2+x_2^2-2x_1x_2=(x_1-x_2)^2\ge 0$).

Now assume $(\bigstar)$ is correct for $n=2^k$, we prove it for $2n=2^{k+1}$

we rewrite the left hand side as

$$(x_1x_2\cdots x_{2n})^{\frac{1}{2n}}=\sqrt{\underbrace{(x_1x_2\cdots x_n)^{\frac{1}{n}}}_{y_1}\underbrace{(x_{n+1}x_{n+2}\cdots x_{2n})^{\frac{1}{n}}}_{y_2}}$$

Now we use the thm for $n=2$ for

$$(x_1x_2\cdots x_{2n})^{\frac{1}{2n}}=\sqrt{y_1y_2}\le\frac{y_1+y_2}{2}\overset{\text{by IH}}{\le}\frac{x_1+x_2+\cdots x_n}{n}+\frac{x_{n+1}+x_{n+2}+\cdots x_{2n}}{n}=\frac{x_1+x_2+\cdots x_{2n}}{2n}$$

Now we use reversed induction to prove the theorem for all n.
Assum (*) is true for an arbitrary n, and consider $n-1$.

define $z = \frac{x_1 + \cdots + x_{n-1}}{n-1}$.

we know the theorem is correct for n numbers in particular $x_1, x_2, \cdots x_{n-1}, z$ thus

$$(x_1 x_2 \cdots x_{n-1} z)^{\frac{1}{n}} \leq \frac{x_1 + x_2 + \cdots + x_{n-1} + z}{n} \overset{\text{due to choice of } z}{=} \frac{(n-1)z + z}{n} = z.$$

Thus $(x_1 \cdot x_2 \cdots x_{n-1} z) \leq z^n$ and $(x_1 x_2 \cdots x_{n-1})^{\frac{1}{n-1}} \leq z = \frac{x_1 + x_2 + \cdots + x_{n-1}}{n-1}$ that we wanted.

Loop Invariants:

Induction is very good for proving correctness of Algorithms. Assume a program has a loop that is supposed to compute a certain value. we can use induction on the number of times the loop is executed to prove that the result is correct. The induction hypo. should reflect the relationships between the variables during the loop execution. such an induction hypo. is called a loop invariant.

Algorithm Convert_to_Binary (n);
Input: n (a positive integer)
Output: b (an array of bits corresponding to the binary representation of n

E.g. $n = 13$, $(1101)_2$
$b =$ |1|0|1|1|0|

```
begin
  t := n {a new variable t to preserve n}
  k := 0;
  while t > 0 do
    k := k+1;
    b[k] := t mod 2;   (%)
    t := t div 2;      (/)
end.
```

Thm: the Algorithm Convert_to_Binary terminates, when the binary representation of n is stored in the array b.

pf: the proof is by induction on k, the number of times the loop is executed. (the induction can be applied only to a part of algorithm and not trivial parts.)

In this case the Induction hypothesis is the loop invariant.

IH: if m is the integer represented by the binary array $b[1..k]$, then $n = t \cdot 2^k + m$.

(Intuitively, it says at step k of the loop, the binary array represents the k least significant bits of n, and the value of t, when shifted by k, corresponds to the rest of the bits.

To prove the correctness of the Algorithm we need to prove
1) the hypo. is true at the beginning (basis)
2) the truth of the hypothesis at step K implies its truth at step K+1 (IH)
3) when the loop terminates, the IH implies the correctness of the algorithm.

1) and 3) are easy. For 1) $k=0$, $m=0$ (since the array is empty by def.) and $n=t$ thus $n = t\,2^0 + 0$.
For 3) $t=0$ and thus $n = 0\,2^k + m = m$.
Finally for 2) we consider two cases for the start of the Kth loop:
a) if t is even, then $t \bmod 2 = 0$ and thus no change to array, t is divided by 2 and k is increamented, i.e., $n = \frac{t}{2}\,2^{k+1} + m = t\,2^k + m$
b) if t is odd, then $b[k+1]$ is set to 1, which contributes 2^k to m, t is changed to $\frac{(t-1)}{2}$ and k is increamented. So the expression is $\frac{(t-1)}{2} \cdot 2^{k+1} + m + 2^k = (t-1)2^k + m + 2^k$
$= t\,2^k + m = n$, as desired. ∎

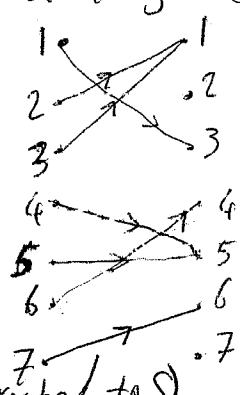
Read the Common Errors in the book to avoid them.

(2.13)

Design of Algorithms by induction: (so far we have seen the use in the proof of thms and correctness of Algorithms) In a sense induction idea give us Recursive Algorithms

Finding One-to-one Mappings:

Let f be a function that maps a finite set $A = \{1, 2, \ldots n\}$ to itself. Assume function f is represented by an array $f[1..n]$ such that $f[i]$ holds the value of $f(i)$ which is an integer between 1 and n. We call f a one-to-one function if for every element j, there is at most one element i that is mapped to j. We can show such a function via a diagram:



The problem: Given a finite set A and a mapping f from A to itself, find a subset $S \subseteq A$ with maximum number of elements, such that 1) the function f maps every element of S to another element of S (f maps S into itself) and 2) no two elements of S are mapped to the same element (i.e. f is one-to-one restricted to S)

If f is originally one-to-one, we are done. If on the other hand $f(i)=f(j)$ $i \neq j$ then S cannot contain both i and j. We can try all subsets $S \subseteq A$, but the running time is $2^n n$ which is exponential. We want a more efficient algorithm. For example here $f(2)=f(3)=1$ so we cannot have but 2 and 3. The choice between 2 and 3 is important if we eliminate 3, then 1 is eliminated and the 2 is eliminated. However if instead eliminate 2 only then we obtain $\{1,3\}$ as a good set.

So the reducing the problem to a smaller one is important and non-trivial. We can use induction hypo: we know how to solve the problem for sets of $n-1$ elements.

The base case is trivial (if there is only one element in the set, then it must be mapped to itself.) but IH is non-trivial when we have a choice (e.g. 2 and 3 in the previous example).

The idea: the element i that has no elements mapped to it cannot belong to S (and thus we must remove i), since if $i \in S$ and $|S|=k$, then those k elements are mapped to at most $k-1$ elements which cannot be one-to-one. So we remove $\{i\}$ from A and iterate. Note that the reduced problem is exactly the same (except the size) as the original problem (the only condition on the set A that f maps A to itself is correct now for $A-\{i\}$.

Here we need to consider n elements instead of 2^n choices for S. Thus the algorithm is much more efficient.

(see Fig 5-3. for the algorithm written in the book)