Mathematical Induction

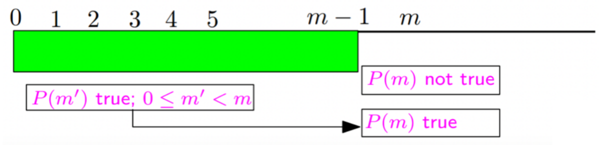
The statement $P(n)$ is true for all $n = 0, 1, 2, \ldots$

We prove this by

(i) Assume that a counterexample exists, i.e., There is some $n > 0$ for which $P(n)$ is false.

(ii) Let $m > 0$ be the smallest value for which $P(n)$ is false

(iii) Then, use the fact that $P(m')$ is true for all $0 \leq m' < m$ to show that $P(m)$ is true, contradicting the choice of m.



Contradiction!

Main idea: We used proof by smallest counterexample to derive that $P(n)$ is true for all $n \in N$.

This is an indirect proof. Is it possible to prove this fact directly?

Since $P(0)$ and $P(n-1) \to P(n)$, we see that

$$P(0) \text{ implies } P(1), \ P(1) \text{ implies } P(2), \ldots$$

The Principle of Mathematical Induction

Well-Ordering Property: Every nonempty set of nonnegative integers has a least element.

The well-ordering property permits us to assume that every set of nonnegative integers has a smallest element, allowing us to use the smallest counterexample.

This is actually equivalent to the principle of mathematical induction.

Principle. (Weak Principle of Mathematical Induction)

(a) Basic Step: the statement $P(b)$ is true

(b) Inductive Step: the statement $P(n-1) \to P(n)$ is true for all $n > b$

Thus, $P(n)$ is true for all integers $n \geq b$.

Another Form of Induction

We may have another form of direct proof as follows.

- First suppose that we have proof of $P(0)$
- Next suppose that we have a proof that, $\forall n > 0$,

$$P(0) \wedge P(1) \wedge P(2) \wedge \ldots \wedge P(n-1) \to P(n)$$

- Then, $P(0)$ implies $P(1)$
- $P(0) \wedge P(1)$ implies $P(2)$
- $P(0) \wedge P(1) \wedge P(2)$ implies $P(3)$ . . .

Iterating gives us a proof of $P(n)$ for all n.

Strong Induction

Principle (Strong Principle of Mathematical Induction):

(a) Basic Step: the statement $P(b)$ is true

(b) Inductive Step: for all $n > b$, the statement

$$P(b) \wedge P(b+1) \wedge \ldots \wedge P(n-1) \to P(n) \text{ is true.}$$

Then, $P(n)$ is true for all integers $n \geq b$.

The statement $P(n)$ is true for all $n = 0, 1, 2, \ldots$

- Proof by contradiction: find the smallest counterexample
- **Well-Ordering Property:** Every nonempty set of nonnegative integers has a least element.
- **Weak Principle of Mathematical Induction**
  - (a) Basic Step: the statement $P(b)$ is true
  - (b) Inductive Step: the statement $P(n-1) \to P(n)$ is true for all $n > b$
- **Strong Principle of Mathematical Induction**
  - (a) Basic Step: the statement $P(b)$ is true
  - (b) Inductive Step: for all $n > b$, the statement

$$P(b) \wedge P(b+1) \wedge \ldots \wedge P(n-1) \to P(n) \text{ is true.}$$

强弱 induction 是等价的

In practice, we do not usually explicitly distinguish between the weak and strong forms.

In reality, they are equivalent to each other in that the weak form is a special case of the strong form, and the strong form can be derived from the weak form.

Recursion

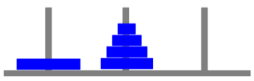
Towers of Hanoi

- 3 pegs, n disks of different sizes
- A legal move takes a disk from one peg and moves it onto another peg so that it is not on top of a smaller disk
- **Problem:** Find a (efficient) way to move all of the disks from one peg to another

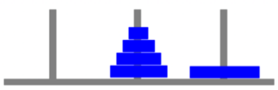
Recursion Base:
If $n = 1$, moving one disk from i to j is easy. Just move it.

Given $i, j \in \{1, 2, 3\}$. Let $\overline{\{i, j\}} = \{1, 2, 3\} - \{i\} - \{j\}$. For example, $\overline{\{1, 2\}} = \{3\}$.

To move $n > 1$ disks from i to j

1) 

move top $n - 1$ disks from i to $\overline{\{i, j\}}$

2) 

move largest disk from i to j

3) 

move top $n - 1$ disks from $\overline{\{i, j\}}$ to j

```
public class Hanoi
{

    public void move(int n, char a, char b, char c)
    {
        if (n == 1)
            System.out.println("plate " + n + " from " + a + " to " + c);
        else
        {
            move(n-1,a,c,b);
            System.out.println("plate " + n + " from " + a + " to " + c);
            move(n-1,b,a,c);
        }

    }
```

Proof of correctness

To prove correctness of solution, we are implicitly using induction:

$p(n)$ is statement that algorithm is correct for n

- $p(1)$ is statement that algorithm works for $n = 1$ disks, which is obviously true
- $p(n - 1) \to p(n)$ is recursion statement that

  if our algorithm works for $n - 1$ disks, then we can build a correct solution for n disks.

Running Time: $M(n)$ is number of disk moves needed for n disks.

- $M(1) = 1$
- if $n > 1$, then $M(n) = 2M(n-1) + 1$

Formally, given

$$M(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2M(n-1)+1, & \textit{otherwise} \end{cases}$$

We show that $M(n) = 2^n - 1$.

Proof. (by induction)

The base case $n = 1$ is true, since $2^1 - 1 = 1$.

For the inductive step, assume that $M(n-1) = 2^{n-1} - 1$ for $n > 1$.

Then $M(n) = 2M(n-1) + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1$

Note that we used induction twice.

- The first time was to derive correctness of algorithm and the recurrence

$$M(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2M(n-1)+1, & \textit{otherwise} \end{cases}$$

- The second time was to derive the closed form solution $M(n) = 2n - 1$ of the recurrence.

Recurrence

A recurrence equation or recurrence for a function:

- defined on the set of integers $\geq b$
- tell us how to compute the n-th value from some or all the first $n-1$ values

To completely specify a function on the basis of a recurrence:

- Basis step (initial condition): Specify the value of the function at zero.
- Recursive step: Give a rule for finding its value at an integer from its values at smaller integers.

Example 1: Let $S(n)$ be the number of subsets of a set of size n, i.e., $|\mathcal{P}(n)|$. What is the formula for $S(n)$?

Consider the subsets of $\{1,2\}$:

$$\emptyset, \{1\}, \{2\}, \{1,2\}$$

Consider the eight subsets of $\{1,2,3\}$:

$$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}$$

It can be seen that

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

This suggests that the recurrence for the number of subsets of an n-element set $\{1,2,\ldots,n\}$ is $S(0) = 1$ and $S(n) = 2S(n-1)$ for $n \geq 1$.

Towers of Hanoi:

- Move $n = 1$ disk; given how to move $n-1$ disks, move n disks
- Prove the correctness of algorithm and running time
- Prove the closed-form solution of the running time

To obtain **recurrence equation:**

- Basis step (initial condition): Specify the value of the function at zero.
- Recursive step: Give a rule for finding its value at an integer from its values at smaller integers.

Example: The number of subsets of a set of size n.

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

Iterating a Recurrence

Let $T(n) = rT(n-1) + a$, where r and a are constants.

Can we generalize this to find a closed-form solution?

$$\begin{aligned} T(n) &= rT(n-1) + a \\ &= r(rT(n-2) + a) + a \\ &= r^2T(n-2) + ra + a \\ &= r^2(rT(n-3) + a) + ra + a \\ &= r^3T(n-3) + r^2a + ra + a \\ &= r^3(rT(n-4) + a) + r^2a + ra + a \\ &= r^4T(n-4) + r^3a + r^2a + ra + a. \end{aligned}$$

Guess $T(n) = r^nT(0) + a\sum_{i=0}^{n-1} r^i$.

The method we used to guess the solution is called iterating the recurrence, because we repeatedly (iteratively) use the recurrence.

Another approach is to iterate from the "bottom-up" instead of "top-down".

$$\begin{aligned} &T(0) = b \\ &T(1) = rT(0) + a = rb + a \\ &T(2) = rT(1) + a = r(rb + a) + a = r^2b + ra + a \\ &T(3) = rT(2) + a = r^3b + r^2a + ra + a \end{aligned}$$

This would lead to the same guess:

$$T(n) = r^nb + a\sum_{i=0}^{n-1} r^i.$$

Formula of Recurrence

Theorem: If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^nb + a\frac{1-r^n}{1-r}$$

for all nonnegative integers n.

- Basis step: We verify that $T(0)$ holds:

$$T(0) = r^0b + a\frac{1-r^0}{1-r}$$

- Inductive step: We show that the conditional statement "if $T(n-1)$ holds, then $T(n)$ holds" for all $n \geq 1$:

  Now assume that $n > 0$ and

$$T(n-1) = r^{n-1}b + a\frac{1-r^{n-1}}{1-r}$$

- Basis step: We verify that $T(0)$ holds:
- Inductive step: We show that the conditional statement "if $T(n-1)$ holds, then $T(n)$ holds" for all $n \geq 1$:

  Now assume that $n > 0$ and

$$T(n-1) = r^{n-1}b + a\frac{1-r^{n-1}}{1-r}.$$

  Thus,

$$\begin{aligned} T(n) &= rT(n-1) + a \\ &= r\left(r^{n-1}b + a\frac{1-r^{n-1}}{1-r}\right) + a \\ &= r^nb + \frac{ar - ar^n}{1-r} + a \\ &= r^nb + \frac{ar - ar^n + a - ar}{1-r} \\ &= r^nb + a\frac{1-r^n}{1-r}. \end{aligned}$$

To obtain the closed-form solution:

- **Iterating a recurrence**:

$$\begin{aligned} T(n) &= rT(n-1) + a \\ &= r(rT(n-2) + a) + a \\ &= r^2T(n-2) + ra + a \\ &= r^2(rT(n-3) + a) + ra + a \\ &= r^3T(n-3) + r^2a + ra + a \\ &= r^3(rT(n-4) + a) + r^2a + ra + a \\ &= r^4T(n-4) + r^3a + r^2a + ra + a. \end{aligned}$$

$$\begin{aligned} &T(0) = b \\ &T(1) = rT(0) + a = rb + a \\ &T(2) = rT(1) + a = r(rb + a) + a = r^2b + ra + a \\ &T(3) = rT(2) + a = r^3b + r^2a + ra + a \end{aligned}$$

- **Theorem:** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^nb + a\frac{1-r^n}{1-r}$$

  for all nonnegative integers n.

- **Proof by mathematical induction.**

First-Order Linear Recurrence

A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a first-order linear recurrence.

- **First Order**: because it only depends upon going back one step, i.e., $T(n-1)$
- If it depends upon $T(n-2)$, then it would be a second-order recurrence, e.g., $T(n) = T(n-1) + 2T(n-2)$.
- **Linear**: because $T(n-1)$ only appears to the first power.
- Something like $T(n) = (T(n-1))^2 + 3$ would be a non-linear first-order recurrence relation.

$$T(n) = f(n)T(n-1) + g(n)$$

When $f(n)$ is a constant, say r, the general solution is almost as easy as we derived before. Iterating the recurrence gives

$$\begin{aligned}
T(n) &= rT(n-1) + g(n) \\
&= r(rT(n-2) + g(n-1)) + g(n) \\
&= r^2T(n-2) + rg(n-1) + g(n) \\
&= r^3T(n-3) + r^2g(n-2) + rg(n-1) + g(n) \\
&\vdots \\
&= r^nT(0) + \sum_{i=0}^{n-1} r^i g(n-i)
\end{aligned}$$

Theorem: For any positive constants a and r, and any function g defined on nonnegative integers, the solution to the first-order linear recurrence

$$T(n) = \begin{cases} rT(n-1) + g(n), & \text{if } n > 0 \\ a, & \text{if } n = 0 \end{cases}$$

is

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i).$$

Proof by induction

Theorem. For any real number $x \neq 1$,

$$\sum_{i=1}^{n} ix^i = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2}.$$

First-order linear recurrence: $T(n) = f(n)T(n-1) + g(n)$

- **First Order**: $T(n-1)$
- **Linear**: because $T(n-1)$ only appears to the first power

Theorem: For any positive constants a and r, and any function g defined on nonnegative integers, the solution to the first-order linear recurrence

$$T(n) = \begin{cases} rT(n-1) + g(n), & \text{if } n > 0 \\ a, & \text{if } n = 0 \end{cases}$$

is

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i).$$

Divide and Conquer Algorithm

We just analyzed recurrences of the form

$$T(n) = \begin{cases} b, & \text{if } n = 0 \\ rT(n-1) + a, & \text{if } n > 0 \end{cases}$$

These corresponded to the analysis of recursive algorithms in which a problem of size n is solved by recursively solving a problem of size $n-1$.

We will now look at recurrences of the form

$$T(n) = \begin{cases} \text{something given}, & \text{if } n \le n_0 \\ rT(n/m) + a, & \text{if } n > n_0 \end{cases}$$

Method: Each guess reduces the problem to one in which the range is only half as big.

This divides the original problem into one that is only half as big; we can now (recursively) conquer this smaller problem.

Note: When n is a power of 2, the number of questions in a binary search on $[1, n]$, satisfies

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ T(n/2) + 1, & \text{if } n \ge 2 \end{cases}$$

This can also be proven inductively.

Consider

$$T(n) = \begin{cases} T(1), & \text{if } n = 1, \\ 2T(n/2) + n, & \text{if } n \geq 2. \end{cases}$$

This corresponds to solving a problem of size n:

- using $T(1)$ work for "bottom" case of $n = 1$
- solving 2 subproblems of size $n/2$ and doing n units of additional work

Algebraically iterating the recurrence (assume that n is a power of 2):

$$T(n) = 2T\left(\tfrac{n}{2}\right) + n \quad = 2\left(2T\left(\tfrac{n}{4}\right) + \tfrac{n}{2}\right) + n$$

$$= 4T\left(\tfrac{n}{4}\right) + 2n \quad = 4\left(2T\left(\tfrac{n}{8}\right) + \tfrac{n}{4}\right) + 2n$$

$$= 8T\left(\tfrac{n}{8}\right) + 3n$$

$$\vdots \qquad \vdots$$

$$= 2^i T\left(\tfrac{n}{2^i}\right) + in$$

End when $i = \log_2 n$

$$\vdots \qquad \vdots$$

$$= 2^{\log_2 n} T\left(\tfrac{n}{2^{\log_2 n}}\right) + (\log_2 n)n$$

$nT(1) + n\log_2 n$

Three Different Behaviours

Compare the iteration for the recurrences

- $T(n) = 2T(n/2) + n$ $\quad nT(1) + n\log_2 n$
- $T(n) = T(n/2) + n$ $\quad \Theta(n)$
- $T(n) = 4T(n/2) + n$ $\quad 2n^2 - n$

Anything in common?

In each case, size of subproblem in next iteration is half the size in the preceding iteration level.

All three recurrences iterate $\log_2 n$ times.

Theorem: Suppose that we have a recurrence of the form

$$T(n) = aT(n/2) + n,$$

where a is a positive integer and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

- If $a < 2$, then $T(n) = \Theta(n)$.
- If $a = 2$, then $T(n) = \Theta(n \log n)$.
- If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$.

proof for a>2

$T(n) = aT(n/2) + n$, where $a > 2$. Assume that $n = 2^i$.

$$T(n) = a^i T\left(\tfrac{n}{2^i}\right) + \left(\frac{a^{i-1}}{2^{i-1}} + \frac{a^{i-2}}{2^{i-2}} + \cdots \frac{a}{2} + 1\right) n$$

$$T(n) = \underbrace{a^{\log_2 n} T(1)}_{\text{Work at "bottom"}} + \underbrace{n \sum_{i=0}^{\log_2 n - 1} \left(\tfrac{a}{2}\right)^i}_{\text{Iterated Work}}$$

$$a^{\log_2 n} T(1) + n \sum_{i=0}^{\log_2 n - 1} \left(\tfrac{a}{2}\right)^i$$

$$\Theta\left(n^{\log_2 a}\right) \qquad \Theta\left(n^{\log_2 a}\right)$$

Since $a > 2$, the geometric series is Θ of the largest term.

$$n \sum_{i=0}^{\log_2 n - 1} \left(\tfrac{a}{2}\right)^i = n\frac{1-(a/2)^{\log_2 n}}{1-a/2} = n\Theta((a/2)^{\log_2 n - 1})$$

To obtain $n(a/2)^{\log_2 n - 1}$:

$$n\left(\tfrac{a}{2}\right)^{\log_2 n - 1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

Note that

$$a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}$$

JST

The Master Theorem

Theorem: Suppose that we have a recurrence of the form

$$T(n) = aT(n/b) + cn^d,$$

where a is a positive integer, $b \geq 1$, c, d are real numbers with c positive and d nonnegative, and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

- If $a < b^d$, then $T(n) = \Theta(n^d)$.
- If $a = b^d$, then $T(n) = \Theta(n^d \log n)$.
- If $a > b^d$, then $T(n) = \Theta(n^{\log_b a})$.