

## Loop Invariants

- **The Division Algorithm:**

```
{Input: integers  $m \geq 0$  and  $n > 0$ }
{Output: integers  $q$  and  $r$  with  $q \cdot n + r = m$  and  $0 \leq r < n$ }
begin;
  {Initialize}
   $q := 0$ ;
   $r := m$ ;
  {Do the real work}
  while  $r \geq n$  do
     $q := q + 1$ ;
     $r := r - n$ ;
  end;
end.
```

- Example:  $m = 17$ ;  $n = 7$ :

$m = 17$ ; $n = 7$	$q$	$r$	$r \geq n$
Initially	0	17	True
After the first pass	1	10	True
After the second pass	2	3	False

- Our main concerns for the Division Algorithm are that:
  - After a while, it stops,
  - When it stops, it gives the right answer.
- How do we know the algorithm will stop?

Answer:

- In every iteration  $r$  is reduced by  $n$ .
  - This forms a decreasing sequence in  $\mathbb{N}$ :  $m, m-n, m-2n, \dots$
  - Sooner or later we will reach a  $k$  such that
$$r = m - k \cdot n < n$$
  - We know we will reach such a  $k$  by the **well ordering principle** which implies that decreasing sequences in  $\mathbb{N}$  must terminate.
- **Well Ordering Principle:**  
Every nonempty subset of  $\mathbb{N}$  has a smallest element.

- **Invariant of a Loop:**

We say that a proposition  $p$  is an **invariant** of the loop  
while  $g$  do  
     $S$

if it satisfies the following condition:

If  $p$  and  $g$  are true before we do  $S$ , then  $p$  is true after we finish  $S$ .

- **Loop Invariant Theorem:**

Suppose that  $p$  is an invariant of the loop "while  $g$  do  $S$ ", and that  $p$  is true on entry into the loop. Then  $p$  is true after each iteration of the loop. If the loop terminates, it does so with  $p$  true and  $g$  false.

- How do we know the Division Algorithm gives the right answer?

Answer:

- " $q \cdot n + r = m$  and  $r \geq 0$ " is a loop invariant.
- By the Loop Invariant Theorem, when the loop stops " $q \cdot n + r = m$  and  $r \geq 0$ " is true and " $r \geq n$ " is false.
- These are the exact conditions the output must satisfy.

- **Theorem:**

Given integers  $m \geq 0$  and  $n > 0$ , the Division Algorithm yields integers  $q$  and  $r$  with  $m = q \cdot n + r$  and  $0 \leq r < n$ .

- Example - Algorithm to calculate n!:

We use the fact that for  $n > 1$ ,  $n! = (n-1)! \cdot n$

```
{Input: integer  $n > 0$ }
{Output: integer FACT with  $FACT = n!$ }
begin
  m := 1;
  FACT := 1;
  {FACT = m!} {loop invariant}
  while m < n do
    m := m + 1;
    FACT := FACT · m;
  end;
end.
```

- Example - Alternate Algorithm for n!:

```
{Input: integer  $n > 0$ }
{Output: integer REV with  $REV = n!$ }
begin
  REV := 1;
  m := n;
  {REV·m! = n!} {loop invariant}
  while m > 0 do
    REV := REV · m;
    m := m - 1;
  end;
end.
```

## Mathematical Induction

- **Principle of Mathematical Induction:**

Let  $p(m), p(m+1), \dots$  be a sequence of propositions. If

(B)  $p(m)$  is true and

(I)  $p(k+1)$  is true whenever  $p(k)$  is true and  $m \leq k$ ,  
i.e.,  $p(k) \rightarrow p(k+1)$  for  $m \leq k$

then all the propositions are true.

- (B) is called the **basis**.

- (I) is called the **inductive step**.

$p(m)$	basis	
$p(m) \rightarrow p(m+1)$	inductive step	
$p(m+1)$	modus ponens	
$p(m+1) \rightarrow p(m+2)$	inductive step	
$p(m+2)$	modus ponens	
...		
$p(k)$	modus ponens	
$p(k) \rightarrow p(k+1)$	inductive step	
$p(k+1)$	modus ponens	
...		
$p(m) \wedge p(m+1) \wedge \dots p(k) \wedge p(k+1) \wedge \dots$		conjunction

- Example: Prove  $n! > 2^n$  for  $n \in \mathbb{N}$  and  $n \geq 4$ .

(B) For  $n = 4$  we have  $4! = 24 > 2^4 = 16$ .  
(The basis has been proven)

(I)  $(k! > 2^k) \rightarrow ((k+1)! > 2^{k+1})$   
(We must prove that this conditional proposition is true)

We assume that our given proposition is true for  $n = k$ ;  
i.e.,  $k! > 2^k$  is true.

We will show this implies that our given proposition  
must be true for  $n = k+1$ ;  
i.e.,  $(k! > 2^k) \rightarrow ((k+1)! > 2^{k+1})$ .

(I)  $(k+1)! = k! \cdot (k+1)$   
 $> 2^k \cdot (k+1)$  (by the inductive assumption  $k! > 2^k$ )  
 $\geq 2^k \cdot 2$  (since  $k+1 \geq 5 > 2$ )  
 $= 2^{k+1}$ .

- Example: Prove  $8^n - 2^n$  is divisible by 6 for every  $n \in \mathbb{P}$ .  
i.e.,  $p(n) = "8^n - 2^n \text{ is divisible by } 6"$  is true for  $n \in \mathbb{P}$ .

(B)  $p(1) = 8 - 2 = 6$  is clearly divisible by 6.  
(The basis has been proven)

(I) " $8^k - 2^k$  is divisible by 6"  $\rightarrow$  " $8^{k+1} - 2^{k+1}$  is divisible by 6"  
(We must prove that this conditional proposition is true)

Assume:  $8^k - 2^k$  is divisible by 6.

Then,

$$\begin{aligned} \text{(I) } p(k+1) &= 8^{k+1} - 2^{k+1} = 8(8^k - 2^k) + 8 \cdot 2^k - 2^{k+1} \\ &= 8(8^k - 2^k) + 8 \cdot 2^k - 2 \cdot 2^k = 8(8^k - 2^k) + 6 \cdot 2^k \\ &\text{which is divisible by 6.} \end{aligned}$$

## Recursive Definitions

- We say that a sequence is defined **recursively** provided that:
  - (B) Some finite set of values are specified (usually the first one or the first few).
  - (R) The remaining values of the sequence are defined in terms of the previous values of the sequence. A formula that gives such a definition is called a **recurrence formula** or **recurrence relation**.
- (B) is the **basis** for the definition.
- (R) is the **recurrence formula** or **recurrence relation**.
- Example - the sequence FACT:
  - (B)  $\text{FACT}(0) = 1$ ,
  - (R)  $\text{FACT}(n+1) = (n+1) \cdot \text{FACT}(n)$
- Example - the sequence  $\text{SUM}(n) = \sum_{i=0}^n \frac{1}{i!}$ 
  - (B)  $\text{SUM}(0) = 1$ ,
  - (R)  $\text{SUM}(n+1) = \text{SUM}(n) + \frac{1}{(n+1)!}$

## Calculating Terms of a Recursively Defined Sequence

- **Iterative Calculation:**

Find  $s_n$  by first computing all of the values  $s_1, s_2, \dots, s_{n-1}$  and then computing  $s_n$ .

- **Recursive Calculation:**

Calculate the value of  $s_n$  by calculating only the terms that  $s_n$  depends on, and calculating those terms only by calculating the terms that they depend on, etc..

- **Example:**

Define the sequence T by:

$$(B) T(1) = 1,$$

$$(R) T(n) = 2 \cdot T(\lfloor n/2 \rfloor) \quad \text{for } n \geq 2$$

$$\begin{aligned} T(73) &= 2 \cdot T(\lfloor 73/2 \rfloor) = 2 \cdot T(36) = 2 \cdot 2 \cdot T(18) = 2 \cdot 2 \cdot 2 \cdot T(9) \\ &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot T(4) = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot T(2) \\ &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot T(1) = 2^6 \end{aligned}$$

We only computed  $T(36)$ ,  $T(18)$ ,  $T(9)$ ,  $T(4)$ ,  $T(2)$ , and  $T(1)$ .

- **Example - The Fibonacci Numbers:**

$$(B) FIB(0) = FIB(1) = 1,$$

$$(R) FIB(n) = FIB(n-1) + FIB(n-2) \quad \text{for } n \geq 2.$$

The first few terms of the sequence are:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

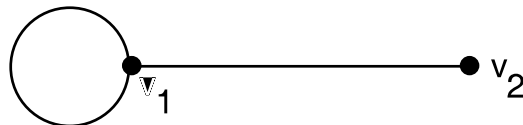
● Example:

Let  $\Sigma = \{a,b\}$ . We are interested in the number  $s_n$  of words of length  $n$  that do not contain the string 'aa'.

- Let  $A_n$  be the set of words in  $\Sigma^n$  having no consecutive a's.
- $A_0 = \{\lambda\}$
- $A_1 = \Sigma$
- $A_2 = \Sigma^2 \setminus \{aa\}$
- Hence,  $s_0 = 1$ ;  $s_1 = 2$ ;  $s_2 = 4 - 1 = 3$ .
- For  $n \geq 2$ , if a word in  $A_n$  ends in 'b', it can be preceded by any word in  $A_{n-1}$ . Thus, there are  $s_{n-1}$  words in  $A_n$  that end in 'b'. If a word in  $A_n$  ends in 'a', the last two letters of the word must be 'ba' and this can be preceded by any word in  $A_{n-2}$ . Thus, there are  $s_{n-2}$  words in  $A_n$  that end in 'a'.

Hence  $s_n = s_{n-1} + s_{n-2} = \text{FIB}(n+1)$ .

● Example - number of paths connecting a pair of vertices:



$$\mathbf{M} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad \mathbf{M}^2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \quad \mathbf{M}^3 = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} \quad \mathbf{M}^4 = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix} \quad \mathbf{M}^5 = \begin{bmatrix} 8 & 5 \\ 5 & 3 \end{bmatrix}$$

$$\mathbf{M}^n = \begin{bmatrix} \text{FIB}(n) & \text{FIB}(n-1) \\ \text{FIB}(n-1) & \text{FIB}(n-2) \end{bmatrix} \quad \text{for } n \geq 2$$

## Recurrence Relations

- **Theorem to obtain the explicit value of  $s_n$ :**

Consider a recurrence relation of the form

$$s_n = as_{n-1} + bs_{n-2}$$

with **characteristic equation**

$$x^2 - ax - b = 0$$

where  $a$  and  $b$  are nonzero constants.

- If the characteristic equation has two different solutions  $r_1$  and  $r_2$ , then

$$s_n = c_1 r_1^n + c_2 r_2^n$$

for certain constants  $c_1$  and  $c_2$ . If  $s_0$  and  $s_1$  are specified, the constants can be determined by setting  $n=0$  and  $n=1$  and solving the simultaneous equations for  $c_1$  and  $c_2$ .

- If the characteristic equation has only one solution  $r$ , then

$$s_n = c_1 r^n + c_2 \cdot n \cdot r^n$$

for certain constants  $c_1$  and  $c_2$ . As above,  $c_1$  and  $c_2$  can be determined if  $s_0$  and  $s_1$  are known.

- **Example:**

$$s_n = s_{n-1} + 2s_{n-2}; s_0 = s_1 = 3.$$

$$x^2 - x - 2 = 0$$

Two solutions:  $r_1 = 2$  and  $r_2 = -1$

$$s_n = c_1 \cdot 2^n + c_2 \cdot (-1)^n$$

$$s_0 = 3 = c_1 \cdot 2^0 + c_2 \cdot (-1)^0 = c_1 + c_2$$

$$s_1 = 3 = c_1 \cdot 2^1 + c_2 \cdot (-1)^1 = 2c_1 - c_2$$

$$c_1 = 2; c_2 = 1$$

$$s_n = 2 \cdot 2^n + (-1)^n = 2^{n+1} + (-1)^n \text{ for } n \in \mathbb{N}$$

## Divide-and-Conquer Algorithms

- **Theorem:**

Let  $(s_n)$  be a sequence that satisfies a recurrence relation of the form:

$$s_{2n} = 2s_n + f(n) \quad \text{for } n \in P.$$

Then

$$s_{2^m} = 2^m \left[ s_1 + \frac{1}{2} \sum_{i=0}^{m-1} \frac{f(2^i)}{2^i} \right] \quad \text{for } m \in \mathbb{N}$$

In particular, if

$$s_{2n} = 2s_n + A + B \cdot n$$

for constants A and B, then

$$s_{2^m} = 2^m \cdot s_1 + (2^m - 1) \cdot A + \frac{B}{2} \cdot 2^m \cdot m$$

Thus, if  $n = 2^m$  we have

$$s_n = n s_1 + (n - 1)A + \frac{B}{2} \cdot n \cdot \log_2 n$$

- **Example:**

Find the largest number in a set of numbers by dividing the set into two disjoint subsets of approximately equal size, finding the largest number in each subset, and then comparing those two to get the largest number.

The time  $T(n)$  such an algorithm takes to process an input of size  $n$  is of the form:

$$T(n) = T(n/2) + T(n/2) + F(n) \text{ or}$$

$$T(2n) = 2T(n) + A \quad (A = \text{time to compare two winners})$$

$$T(2^m) = 2^m T(1) + (2^m - 1)A \quad (\text{from theorem})$$

$$T(n) = nT(1) + (n - 1)A \quad (\text{substitute } n = 2^m)$$

## More Induction

- **First Principle of Mathematical Induction:**

Let  $m$  be an integer and let  $p(n)$  be a sequence of propositions defined on the set  $\{n \in \mathbb{Z} : n \geq m\}$ . If

(B)  $p(m)$  is true and

(I) for  $k > m$ ,  $p(k)$  is true whenever  $p(k-1)$  is true,

then  $p(n)$  is true for every  $n \geq m$ .

- **Second Principle of Mathematical Induction:**

Let  $m$  be an integer and let  $p(n)$  be a sequence of propositions defined on the set  $\{n \in \mathbb{Z} : n \geq m\}$ . and let  $j$  be a nonnegative integer. If

(B)  $p(m), \dots, p(m+j)$  are true and

(I) for  $k > m+j$ ,  $p(k)$  is true whenever

$p(m), \dots, p(k-1)$  are all true,

then  $p(n)$  is true for every  $n \geq m$ .

- **Example:**

Given:

$$b_0 = b_1 = 1; \quad b_n = 2b_{n-1} + b_{n-2} \text{ for } n \geq 2.$$

Prove:

$p(n) = "b_n \text{ is an odd number}"$  is true.

Proof:

(B)  $p(0)$  and  $p(1)$  are obviously true.

For  $k \geq 2$ , assume:  $b_n$  is odd for  $0 \leq n < k$ .

(I)  $b_k = 2b_{k-1} + b_{k-2}$

$b_k$  must be odd since it is the sum of an even number and an odd number. (The odd number is  $b_{k-2}$ ).

## Second Principle of Mathematical Induction (more detailed)

Let  $m$  be an integer and let  $p(n)$  be a sequence of propositions defined on the set  $\{n \in \mathbb{Z} : n \geq m\}$ . and let  $j$  be a nonnegative integer. If

(B)  $p(m), \dots, p(m+j)$  are true and

(I) for  $k > m+j$ ,  $p(k)$  is true whenever

$p(m), \dots, p(k-1)$  are all true,

i.e.,  $p(m) \wedge \dots \wedge p(k-1) \rightarrow p(k)$  for  $m+j < k$

then  $p(n)$  is true for every  $n \geq m$ .

- (B) is called the **basis**.
- (I) is called the **inductive step**.

$p(m) \wedge \dots \wedge p(m+j)$	basis
$p(m) \wedge \dots \wedge p(m+j) \rightarrow p(m+j+1)$	inductive step
$p(m+j+1)$	modus ponens
$p(m) \wedge \dots \wedge p(m+j) \wedge p(m+j+1)$	conjunction
$p(m) \wedge \dots \wedge p(m+j+1) \rightarrow p(m+j+2)$	inductive step
$p(m+j+2)$	modus ponens
$p(m) \wedge \dots \wedge p(m+j+1) \wedge p(m+j+2)$	conjunction
...	
$p(k-1)$	modus ponens
$p(m) \wedge \dots \wedge p(k-1)$	conjunction
$p(m) \wedge \dots \wedge p(k-1) \rightarrow p(k)$	inductive step
$p(k)$	modus ponens
$p(m) \wedge \dots \wedge p(k-1) \wedge p(k)$	conjunction
...	
$p(m) \wedge \dots \wedge p(m+j) \dots \wedge p(k) \wedge \dots$	conjunction