MAT2540, Classwork3, Spring2026

5.4 Recursive Algorithms

1. Definition of **Recursive Algorithms**.

An algorithm is called recursive if a function calls itself to solve smaller parts of a given problem. It continues until a base condition is met to stop further calls.

2. Find gcd(91, 287) by using the Euclidean algorithm.

$$\gcd(91,287)=\boxed{\gcd(287 \bmod 91, 91)}=\gcd(14,91)=\boxed{\gcd(91 \bmod 14, 14)}$$
$$=\gcd(7,14)=\boxed{\gcd(14 \bmod 7, 7)}=\gcd(0,7)=7$$

3. A Recursive Algorithm for Computing **gcd**(a, b).

> **procedure** *gcd*(a, b: nonnegative integers with a < b)
>
> **if** $a = 0$ **then return** b
>
> **else return** *gcd*(b mod a , a)
>
> {output is gcd(a, b)}

4. Find the value of 2^{31} **mod** 3.

$$2^{31} \bmod 3 = (2^{30} \bmod 3)\cdot(2^1 \bmod 3) = \boxed{(2^{15} \bmod 3)^2\cdot(2 \bmod 3)}$$
$$2^{15} \bmod 3 = (2^{14} \bmod 3)\cdot(2^1 \bmod 3) = (\boxed{2^7 \bmod 3})^2\cdot(2 \bmod 3)$$
$$2^7 \bmod 3 = (2^6 \bmod 3)\cdot(2 \bmod 3) = (\boxed{2^3 \bmod 3})^2\cdot(2 \bmod 3)$$
$$2^3 \bmod 3 = (2^2 \bmod 3)\cdot(2 \bmod 3) = (\boxed{2 \bmod 3})^2\cdot(2 \bmod 3)$$
$$2^1 \bmod 3 = (2^0 \bmod 3)^2\cdot(2 \bmod 3)$$
$$2^0 \bmod 3 = 1$$

$$2^1 \bmod 3 = (1)^2\cdot 2 \bmod 3 = 2$$
$$2^3 \bmod 3 = (2)^2\cdot 2 \bmod 3 = 2$$
$$2^7 \bmod 3 = 2^2\cdot 2 \bmod 3 = 2$$
$$2^{15} \bmod 3 = 2^2\cdot 2 \bmod 3 = 2$$
$$2^{31} \bmod 3 = 2^2\cdot 2 \bmod 3 = 2$$

5. A Recursive Modular Exponentiation b^n **mod** m

$b^{\frac{n}{2}} \bmod m$

> **procedure** *mpower*$(b, n, m$: integers with $b > 0$, and $m \geq 2, n \geq 0)$
>
> base condition → **if** $n = 0$ **then return** 1
>
> **else if** n is even **then return** $\left(mpower(b, \frac{n}{2}, m)\right)^2 \bmod m$
>
> when n is odd → **else return** $\left(\left(mpower(b, \lfloor \frac{n}{2} \rfloor, m)\right)^2 \bmod m \cdot (b \bmod m)\right) \bmod m$
>
> {output is b^n **mod** m }

$\lfloor x \rfloor$: floor

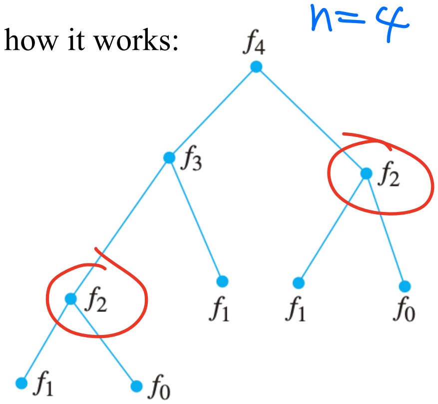
$x-1 <$ an integer $< x$

$f_0, f_1, f_2 = f_1 + f_0, f_3 = f_2 + f_1 \cdots$

6. A Recursive Algorithm for Fibonacci Numbers and an example (f_4) showing how it works:

$n = 4$

> **procedure** *fibonacci*(n, : nonnegative integer)
>
> base condition → **if** $n = 0$ **then return** 0
>
> **else if** $n = 1$ **then return** 1
>
> **else return** *fibonacci*(n−1)+ *fibonacci*(n−2)
>
> {output is *fibonacci*(n)}



7. Definition of **Iterative Algorithms**.

An algorithm is called iterative if it is repeatedly executing a set of instructions using loops like for do-while, while. It continues until a specified condition becomes false

8. Why Iterative Algorithms?

- less computation than a recursion procedure
- less ram.?
- But the code might be longer.

9. An Iterative Algorithm for Computing Fibonacci Numbers and an example finding f_4.

procedure *iterative fibonacci*(n: nonnegative integer)
if $n = 0$ **then return** 0
else $x := 0$; $y := 1$;
for $i := 1$ to $n - 1$
$z := x+y$; $x := y$; $y := z$ ($i = i+1$)
return y
{output is the nth Fibonacci number}

Find f_4:

Initialization: $n = 4$

Since $n \neq 0$, then $x = 0, y = 1, i = 1$

	Round	z	x	y	i
$i=1$	1	1	1	1	2
$i=2$	2	2	1	2	3
$i=3$	3	3	2	3	STOP

Return $y=3$ (which means $f_4 = 3$)

10. An Algorithm to Merge Two ordered lists L_1 and L_2 to a merged list L:

First, compare the smallest elements in the L_1 and L_2. If, for example, the one from L_1 is smaller than the one from L_2, then put the smaller one at the beginning of the merged list L and remove it from the list L_1.

Next, repeat this process until all the elements in both lists are placed in L.

Here we use an example to explain the algorithm:

Merge the two lists 2, 3, 5, 6 and 1, 4.

Algorithm:

Initialization: $L = \{\}$, empty set, $L_1 = \{2, 3, 5, 6\}$, and $L_2 = \{1, 4\}$.

First Round:
The first element in L_1: 2 The first element in L_2: 1
Comparison: $2 > 1$ $\Rightarrow L_1 = \{2, 3, 5, 6\}$, $L_2 = \{4\}$
and $L = \{1\}$
Next round? $L_1 \neq \phi$, $L_2 \neq \phi \Rightarrow$ Yes

Second Round:
The first element in L_1: 2 The first element in L_2: 4
Comparison: $2 < 4$ $\Rightarrow L_1 = \{3, 5, 6\}$, $L_2 = \{4\}$
and $L = \{1, 2\}$
Next round? $L_1 \neq \phi$, $L_2 \neq \phi \Rightarrow$ Yes

Third Round:
The first element in L_1: 3 The first element in L_2: 4
Comparison: $3 < 4$ $\Rightarrow L_1 = \{5, 6\}$, $L_2 = \{4\}$
and $L = \{1, 2, 3\}$
Next round? $L_1 \neq \phi$, $L_2 \neq \phi \Rightarrow$ Yes

Fourth Round:
The first element in L_1: 5 The first element in L_2: 4
Comparison: $5 > 4$ $\Rightarrow L_1 = \{5, 6\}$, $L_2 = \phi$,
and $L = \{1, 2, 3, 4\}$
Next round? $L_2 = \phi \Rightarrow$ NO Next round
$L = \{1, 2, 3, 4, 5, 6\}$