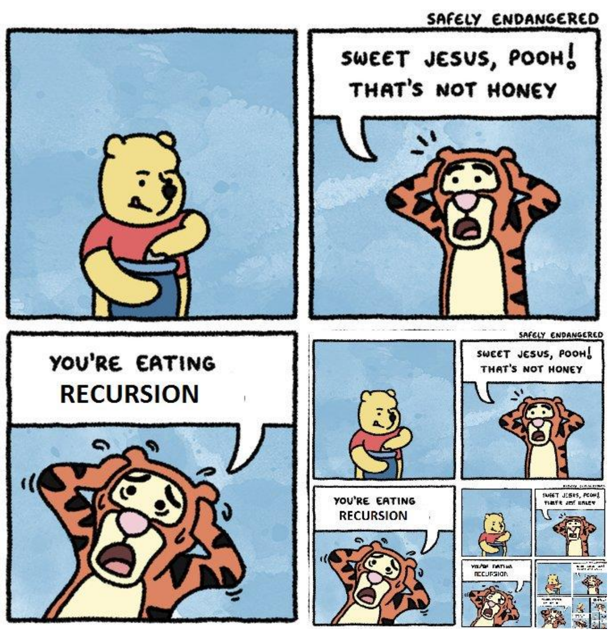
Chapter 3.1
Recursive Definitions

Instructor: Abhishek Santra
Email: abhishek.santra@uta.edu

1



2

What are recursive sequences?

- S1: 1, 2, 3, 4, …
  - *Is the current term, 1 + the previous term?*
- S2: 1, 2, 4, 8, 16, …
  - *Is the current term, 2 multiplied by the previous term?*

Recursive Sequences (Intro)

- Inductive or recursive definition
  - A definition in which the **item being defined appears as part of the definition**
- Two parts of recursive definition
  - A basis
    - Some **simple cases** of the item being defined are **explicitly given**
  - An inductive or recursive step
    - New cases of the **item being defined are given in terms of previous cases**
- Construct new cases from the basis then construct other cases from these new ones
- Similar to **proof by induction**

Recursive Sequence

- A sequence is *defined recursively*,
  - by **explicitly** naming the **first value (or the first few values)** in the sequence,
    - E.g., $S(1) = 1$
  - and then **defining later values** in the sequence **in terms of earlier values**.
    - E.g., $S(n) = S(n-1) + 1$, for $n \geq 2$
  - **What is the final sequence?**
    - *Sequence S* $\Rightarrow$ *1, 2, 3, 4, …*

Recursive Sequence Examples

- **Geometric Sequence Example**
  - $S(1) = 2$
  - $S(n) = 2S(n-1)$ for $n \geq 2$
    - Sequence $S \Rightarrow 2, 4, 8, 16, 32, \ldots$
- **Arithmetic Sequence**
  - $T(1) = 1$
  - $T(n) = T(n-1) + 3$ for $n \geq 2$
    - Sequence $T \Rightarrow 1, 4, 7, 10, 13, \ldots$

Recursive Sequence (Fibonacci)

- **Fibonacci Sequence**
  - $F(1) = 1$
  - $F(2) = 1$
  - $F(n) = F(n\text{-}1) + F(n\text{-}2)$ for $n > 2$
    - Sequence $F \Rightarrow$ 1, 1, 2, 3, 5, 8, 13,….
- **Fibonacci Sequence in Nature**
  - Number of petals in daisies is a Fibonacci number

Recursive Sequence (Fibonacci)

- Can you use the basics of Fibonacci sequence to show that **F(n+4) = 3F(n+2) – F(n) for all n >= 2** ?
- **Solution:**

F(n+4) = **F(n+3)** + F(n+2)

= F(n+2) + F(n+1) + F(n+2) *(rewriting F(n+3))*

Trying to replace F(n+1)

F(n+2) = F(n+1) + F(n)

F(n+1) = F(n+2) – F(n) (1)

Thus,

F(n+4) = F(n+2) + **F(n+2) – F(n)** + F(n+2) *(rewriting F(n+1) using (1))*

= 3F(n+2) – F(n), for all n >= 2

Hence, proved!

Recursively defined operations

- **Exponential operation**
  - $a^0 = 1$
  - $a^n = (a^{n-1})a$ for $n \geq 1$
    - E.g. $3^3 = (\mathbf{3^2})3 = (\mathbf{3^1})3.3 = (\mathbf{3^0})3.3.3 = 1.3.3.3 = 27$
- **Multiplication operation** (for two positive integers, m and n)
  - $m(1) = m$
  - $m(n) = m(n\text{-}1) + m \quad$ for $n \geq 2$
    - E.g. 5 * 3 = 5(3) = **5(2)** + 5 = **5(1)** + 5 + 5 = 5 + 5 + 5 = 15
- **Factorial Operation**
  - $F(0) = 1$
  - $F(n) = n.F(n\text{-}1)$ for $n \geq 1$
    - E.g. F(4) = 4.F(3) = 4.3.F(2) = 4.3.2.F(1) = 4.3.2.1.F(0) = 24
    - Also, represented as **4!**

Recursively defined algorithms

- If a recurrence relation exists for an operation, the **algorithm for such a relation** can be written either iteratively or recursively
- **Iterative way:** (2, 4, 8, 16, 32,….)

  $S(1) = 2$

  $S(n) = 2S(n\text{-}1)$ for $n \geq 2$

```
S(integer n) //function that iteratively computes the value S(n)
Local variables:
    integer i ;//loop index
    CurrentValue ;
if n =1 then
    return 2
else
    i = 2
    CurrentValue = 2
    while i ≤ n
        CurrentValue = CurrentValue *2
        i = i+1
    end while
    return CurrentValue
end if
end function S
```

Recursively defined algorithms

- Recursive way: function **calls** itself

Sequence S is 2, 4, 8, 16, …

$S(1) = 2$

$S(n) = 2S(n-1)$ for $n \geq 2$

```
S(positive integer n)
//function that recursively computes the value S(n)
   if n =1 then
      return 2
   else
      return 2 * S(n-1)
   end if
end function S
```

S(3) = 2 * **S(2)**

= 2 * (2 * **S(1)**)

= 2 * (2 * **2**)

= 8

Recursive algorithm for Selection Sort

- Algorithm to **sort recursively** an **input sequence of numbers** in **increasing or decreasing order**

```
Function sort(List s,Integer n)      //This function sorts in increasing order
   if n =1 then
          output "sequence is sorted"   //base case
   end if
   max_index = 1                        //assume s_1 is the largest
   for j = 2 to n do
          if s[j] > s[max_index] then
                 max_index = j          //found larger, so update
          end if
   end for
   exchange/swap s[n] and s[max_index]   //move largest to the end
   return(sort(s,n-1))
end function sort
```

Selection Sort Example

- Sequence *S* to be sorted by increasing order:
  - *S*: 23 12 9 -3 89 54, n = 6
- Before 1st recursive call, the sequence is:
  - Swap 89 and 54
  - *S*: 23 12 9 -3 54 89, n = 5
- After 1st recursive call, nothing is swapped:
  - *S*: 23 12 9 -3 54 89, n = 4
- After 2nd recursive call, the sequence is:
  - Swap 23 and -3
  - *S*: -3 12 9 23 54 89, n = 3
- After 3rd recursive call, the sequence is:
  - Swap 12 and 9
  - *S*: -3 9 12 23 54 89, n = 2
- After 4th recursive call, nothing is swapped:
  - *S*: -3 9 12 23 54 89, n = 1
- After 5th recursive call, *we meet base case, since n=1*

- **Final sorted array *S*: -3 9 12 23 54 89**

Recursive algorithm for binary search

- Looks for a **value *x* in a sorted sequence**

```
Function binary_search(list L, int i, int j, itemtype x)
// Here L is assumed to be sorted in increasing order
// searches list L from L[i] to L[j] for item x
   if i > j then              //not found
     write ("not found")
   else
     find the index k of the middle item in the list L[i]-L[j]
     if x = middle item then
       write ("found")
     else
       if x < middle item then
         binary_search(L,i,k-1,x)          // lower half
       else
         binary_search(L,k+1,j,x)          // upper half
       end if
     end if
   end if
end binary_search
```

Binary Search Example

- Search for **81** in the sequence **3 6 18 37 76 81 92**
  - Sequence has 7 elements.
  - Calculate middle point: (1 + 7)/2 = 4.
  - Compares 81 to 37 (4^{th} sequence element): *no match*
  - Search in the second half since 81 > 37
  - New sequence for search: 76 81 92
  - Calculate midpoint: (5 + 7)/2 = 6
  - 6^{th} Element: 81 Compares 81 to 81: *perfect match*
  - **Element 81 found as the 6^{th} element of the sequence**

Class Exercise

- **Ackermann function**

$$A(m,n) = \begin{cases} n+1, & m = 0 \\ A(m-1, 1), & \text{for } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)), & \text{for } m > 0 \text{ and } n > 0 \end{cases}$$

- **Find the terms $A(1,1)$, $A(1,2)$, $A(2,1)$**
  - $A(1,1) = A(0, A(1,0)) = A(0, A(0,1)) = A(0,2) = 3$
  - $A(1,2) = A(0, A(1,1)) = A(0, 3) = 4$
  - $A(2,1) = A(1, A(2,0)) = A(1, A(1,1)) = A(1, 3) = A(0, A(1,2)) = A(0, 4) = 5$

Class Exercise

- What does the following function calculate? **mystery(n) = ?**

Function *mystery*(integer *n*)
if *n* = 1 then
 return 1
else
 return (*mystery*(*n*-1)+1)
end if
end function *mystery*

- **Solution**
  - Mystery(n) = n, for all n >=1

Class Exercise

- Write the recursive algorithm for the recursive definition of the following sequence
  - *a*, *b*, *a*+*b*, *a*+2*b*, 2*a*+3*b*, 3*a*+5*b*

- **Solution**

Function *S(integer n)*
if n = 1
  return *a*
else
  if n = 2
    return *b*
  else
    return *S(n-1) + S(n-2)*
  endif
endif
end Function *S*

Discussion