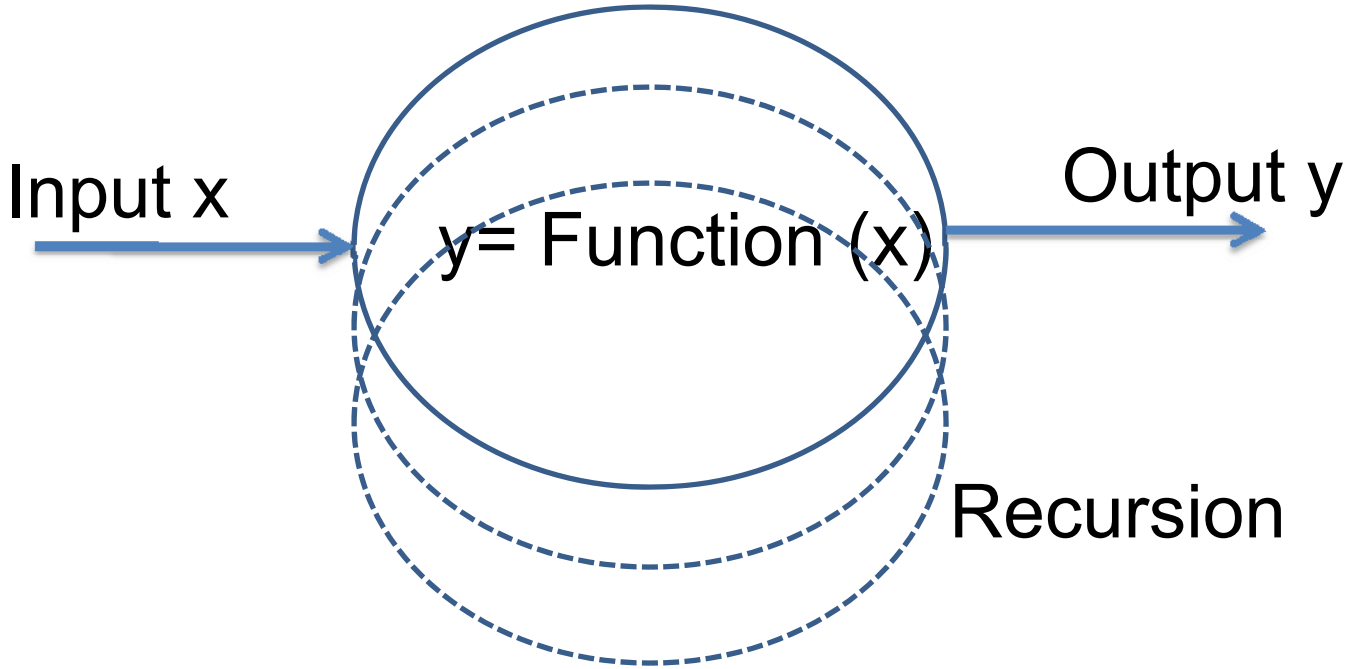
CSE 20 Lecture 11
Function, Recursion & Analysis

CK Cheng
UC San Diego

Motivation



- Verification
- Complexity of the execution

OUTLINE

- Definition of Function
- Formulation of Recursive Functions
- Induction (Verification)
- Counting (Analysis)

I. Definition

- A function f: A → B maps elements in domain A to codomain B such that for each a ∈ A, f(a) is exact one element in B.
- f: A → B

  A: Domain

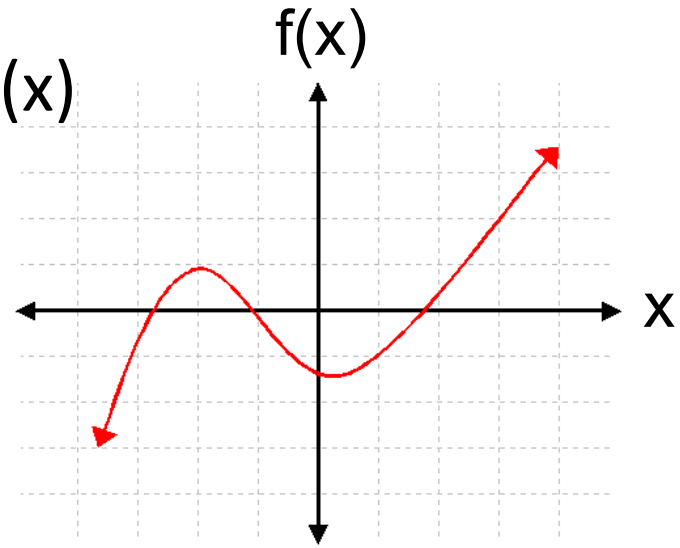
  B: Codomain

  f(A): range or image of function f(A)⊂B

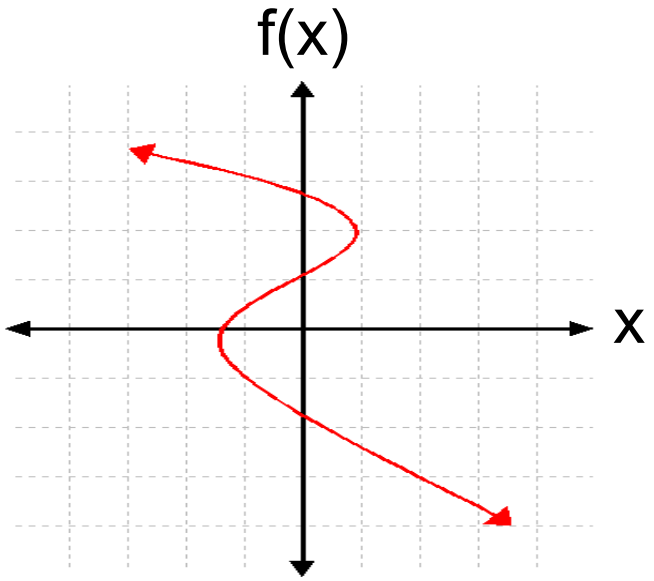
Examples

(1) $f(x)=x^2$, $x \in \mathbb{R}$

(2) f(x)

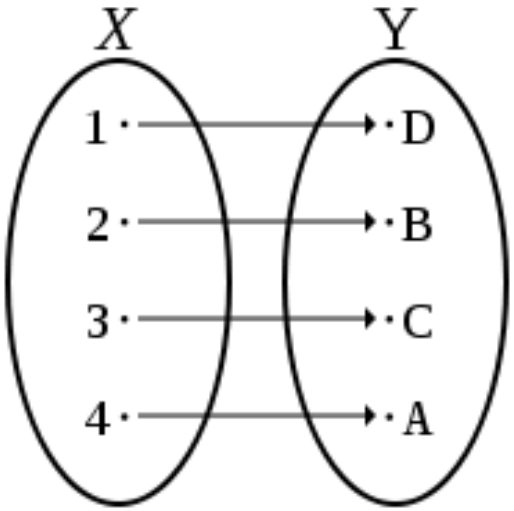


(3) f(x) NOT a function

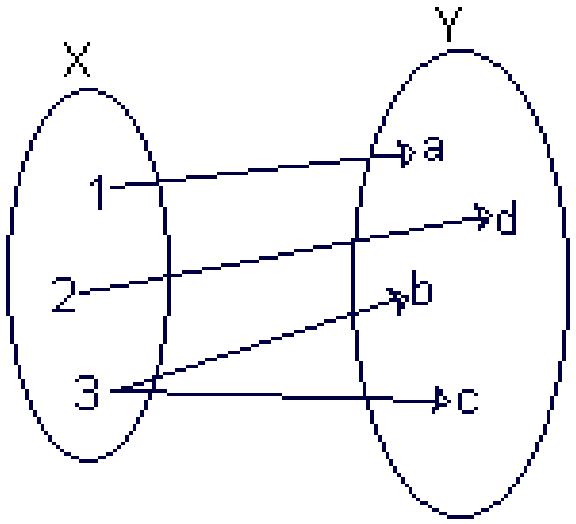


Examples

(4) Mapping from x to y



(5) NOT a function



When domain A is an integer set Z, we may denote f(x) as f_x, ie. f_0, f_1, f_2.

Function: iClicker

Let X={1,2,3,4}. Determine which of the following relation is a function

A. f={(1,3), (2,4), (3,3)}

B. g={(1,2), (3,4), (1,4), (2,3)}

C. h={(1,4), (2,3), (3,2), (4,1)}

D. w={(1,1), (2,2), (3,3), (4,4)}

E. Two of the above.

Formulation of Recursive Functions

1. Overview of Recursive Functions
2. Examples
    1. Fibonacci Sequence
    2. The Tower of Hanoi
    3. Merge Sort

Formulation of Recursive Function: overview

1. Recursion: A function calls itself.
2. Reduction: For each call, the problem size becomes smaller.
3. Base Case: The smallest sized problem.
    - The derivation of the base case is defined.

Recursive Functions: iClicker

A. We can convert a recursive function into a program with iterative loops.
B. Recursive function takes much less lines of codes for some problems.
C. Recursive function always runs slower due to its complexity.
D. Two of the above
E. None of the above.

Fibonacci Sequence: Problem

Fibonacci sequence: Start with a pair of rabbits. For every month, each pair bears a new pair, which becomes productive from their second month.

Calculate the number of new pairs in month i, $i \geq 0$.

Fibonacci Sequence: Algorithm

- Algorithm

array n(i) (# new born pairs), a(i) (# adult pairs)

Init n(0)=0, a(0)=1

For i=1, i=i+1, i< #MaxIter

{ n(i)=a(i-1)

a(i)=a(i-1)+n(i-1)

}

Index: 0 1 2 3 4 5 6 …

a(i): 1 1 2 3 5 8 13 …

n(i): 0 1 1 2 3 5 8 …

Fibonacci Sequence: iClicker

The following is Fibonacci sequence:

- A. 0 1 1 2 3 5 8 13 21 34
- B. 0 1 1 2 3 5 8 13 18 31
- C. 0 1 1 2 3 5 8 13 18 36
- D. 0 1 1 2 3 5 8 13 17 30
- E. None of the above

Fibonacci Sequence: Recursion

Function Fibonacci(n)

If n< 2, return n

Else return(Fibonacci(n-1)+Fibonacci(n-2))

Index: 0 1 2 3 4 5 6 …

F(i): 0 1 1 2 3 5 8 …

Fibonacci Sequence: Complexity

- Fibonacci Sequence

$$f_0 = 0,\ f_1 = 1,\ f_n = f_{n-1} + f_{n-2} \quad \forall\, n > 1$$

Index	0	1	2	3	4	5	6
Sequence	0	1	1	2	3	5	8

Fibonacci sequence: Golden Ratio

$$X_n = \frac{f_n}{f_{n-1}} \qquad X_{n\to\infty} = \frac{1+\sqrt{5}}{2}$$

- Derivation:

$$\frac{f_n}{f_{n-1}} = \frac{f_{n-1}+f_{n-2}}{f_{n-1}} = 1+\frac{f_{n-2}}{f_{n-1}} = 1+\frac{1}{\frac{f_{n-1}}{f_{n-2}}}$$

- Let $X = X_{n\to\infty}$
- We have: $X = 1+1/X$

Fibonacci Sequence and the golden ratio

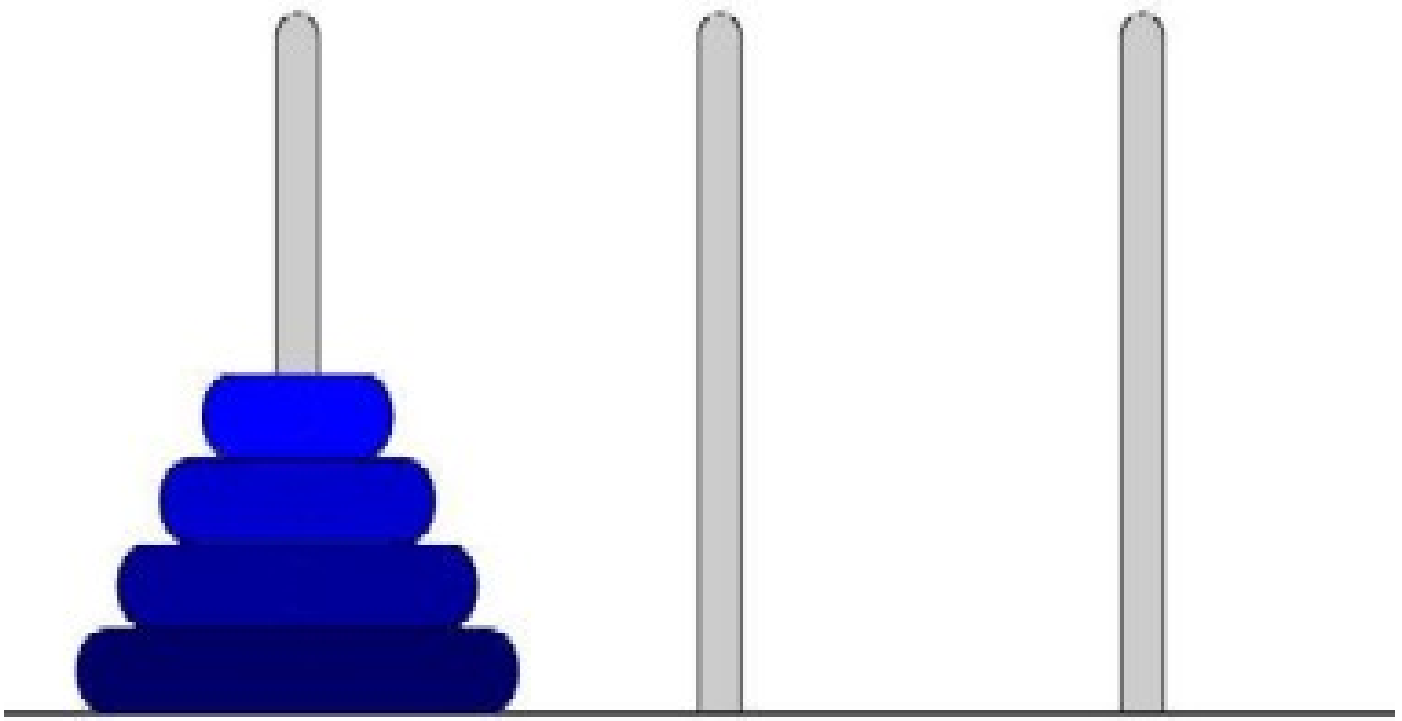
	0	1	2	3	4	5	6	7	8	9
	0	1	1	2	3	5	8	13	21	34
$\frac{f_n}{f_{n-1}}$			1	2	1.5	1.6	1.61	1.625	1.615	1.619

Tower of Hanoi: Problem

- Setting: Three poles and a stack of disks on the leftmost pole in order of size with the largest at bottom.
- Obj: Move the entire pile of disks to the rightmost pole.
- Rules:
  - Each move involves only one top disk on one of three poles.
  - The top disk may be moved to either of the other two poles.
  - A disk must never be on top of a smaller disk.

Tower of Hanoi: Problem

Hanoi (4, 1, 3): Move 4 disks from pole 1 to pole 3



Tower of Hanoi: Function

Setting: n, A=1, B=3 (move n disks from pole1 to pole 3)

Output: a sequence of moves (a, b)

Function Hanoi(n, A, B)

If n=1, return (A, B),

Else {C <= other pole(not A nor B)

return Hanoi(n-1, A, C), (A, B), Hanoi(n-1, C, B)

}

Tower of Hanoi: Example

Function Hanoi(n, A, B)

If n=1, return (A, B),

Else {C<= other pole(not A nor B)

  return Hanoi(n-1, A, C), (A, B), Hanoi(n-1, C, B)

}

Example: Hanoi(2, 1, 3)

  =Hanoi(1, 1, 2), (1, 3), Hanoi(1, 2, 3)

  =(1,2), (1,3), (2,3)

Tower of Hanoi: Complexity

Number of moves for Hanoi(n, A, B)

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

n	1	2	3	4	5	6	7	8	9	10
F(n)	1	3	7	15	31	63	127	255	511	1023

Merge Sort: Problem

- Given an array of numbers, sort the numbers from small to large.

id	0	1	2	3	4	5	6	7
items	5	2	4	7	1	3	2	6

Merge Sort: Function

Initial: Array A from p=0 to r=n-1

Function Merge-Sort(A, p, r)

If p < r

Then { q= Floor((p+r)/2)

    Merge-Sort(A, p, q)

    Merge-Sort(A, q+1, r)

    Merge(A, p, q, r)

}

Merge Sort: Function

Function Merge(A, p, q, r)

Create and copy arrays L=A[p, q], R=A[q+1, r]

Set initial i=j=0, L[q-p]=R[r-q]=inf

for k=p to r

{ if L[i]≤R[j]

   then A[k]=L[i]

        i=i+1

   else A[k]=R[j]

        j=j+1

}

Merge Sort: Complexity

Complexity: Let F(n) be the number of L[i]≤R[j] comparisons in Merge-Sort(A, 0, 2^n-1).

- F(0)= 0
- F(1)= 2
- F(2)= 2F(1)+2^2
- F(n)= 2F(n-1)+2^n