DATA STRUCTURE AND ALGORITHM

Dr. Khine Thin Zar
Professor
Computer Engineering and Information Technology Dept.
Yangon Technological University

Lecture 2

Mathematical Preliminaries

Outlines of Class (Lecture 2)

- Set
- Equivalence Relation
- Miscellaneous Notation
- Mathematical Proof Techniques
  - ✓ Direct Proof
  - ✓ Proof by Contradiction
  - ✓ Mathematical Induction

Sets

- A set is a collection of objects and it is applied the mathematical concepts in computer science.
- Notations and techniques of set theory are commonly used when describing and implementing algorithms.
- A set is a collection of distinguishable members or elements.
- The members are typically drawn from some larger population known as the base type
- Eg. P's members are 7, 11, and 42, and the base type is integer.

Rules

- Elements should not be repeated.
- Each element should have a reason to be in the set.

Null Set (Empty Set)

- The set with no element is called null set or empty set.

For example:

- A={x | x is a prime number, where 24<x<29}
- This set can't contain a element.
- A={ }

Basic Operations

- set union (set1,set2)
- set intersection (set1,set2)
- set difference (set1,set2)
- int subset (set1,set2)

Set Notation

$\{1, 4\}$	A set composed of the members 1 and 4
$\{x \mid x \text{ is a positive integer}\}$	A set definition using a **set former** Example: the set of all positive integers
$x \in \mathbf{P}$	x is a member of set **P**
$x \notin \mathbf{P}$	x is not a member of set **P**
$\emptyset$	The null or empty set
$\lvert\mathbf{P}\rvert$	Cardinality: size of set **P** or number of members for set **P**
$\mathbf{P} \subseteq \mathbf{Q}, \mathbf{Q} \supseteq \mathbf{P}$	Set **P** is included in set **Q**, set **P** is a subset of set **Q**, set **Q** is a superset of set **P**
$\mathbf{P} \cup \mathbf{Q}$	Set Union: all elements appearing in **P** OR **Q**
$\mathbf{P} \cap \mathbf{Q}$	Set Intersection: all elements appearing in **P** AND **Q**
$\mathbf{P} - \mathbf{Q}$	Set difference: all elements of set **P** NOT in set **Q**

Set Notation (Example)

- $P = \{2,3,5\}$, $Q=\{5,10\}$
- $|P|=3$ and $|Q|=2$
- $P \cup Q = Q \cup P = \{2,3,5,10\}$
- $P \cap Q = Q \cap P = \{5\}$
- $P-Q \neq Q-P$; $P-Q = \{2,3\}$, $Q-P = \{10\}$

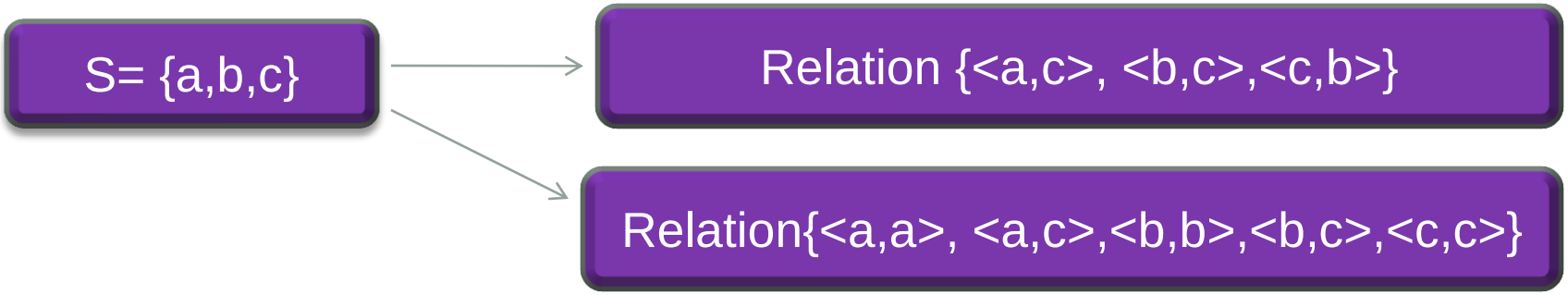
Sets

- Powerset of a set S
  - ✓ The set of all possible subsets for S
  - ✓ The powerset of S={a,b,c} is: {Ø ,{a},{b},{c},{a,b},{a,c},{b,c},{a,b,c}}
- Bag
  - ✓ A collection of elements with no order, but with duplicate-valued elements
  - ✓ [3,4,5,4]
  - ✓ eg. bag [3,4,5,4]is distinct from bag [3,4,5]
- Sequence
  - ✓ A collection of elements with an order, and which may contain duplicate-valued elements
  - ✓ is also sometimes called a tuple or a vector
  - ✓ Sequence <3,5,4,4>is distinct from sequence <3,4,5,4>
  - ✓ Both are distinct from sequence <3,4,5>.

Contd.

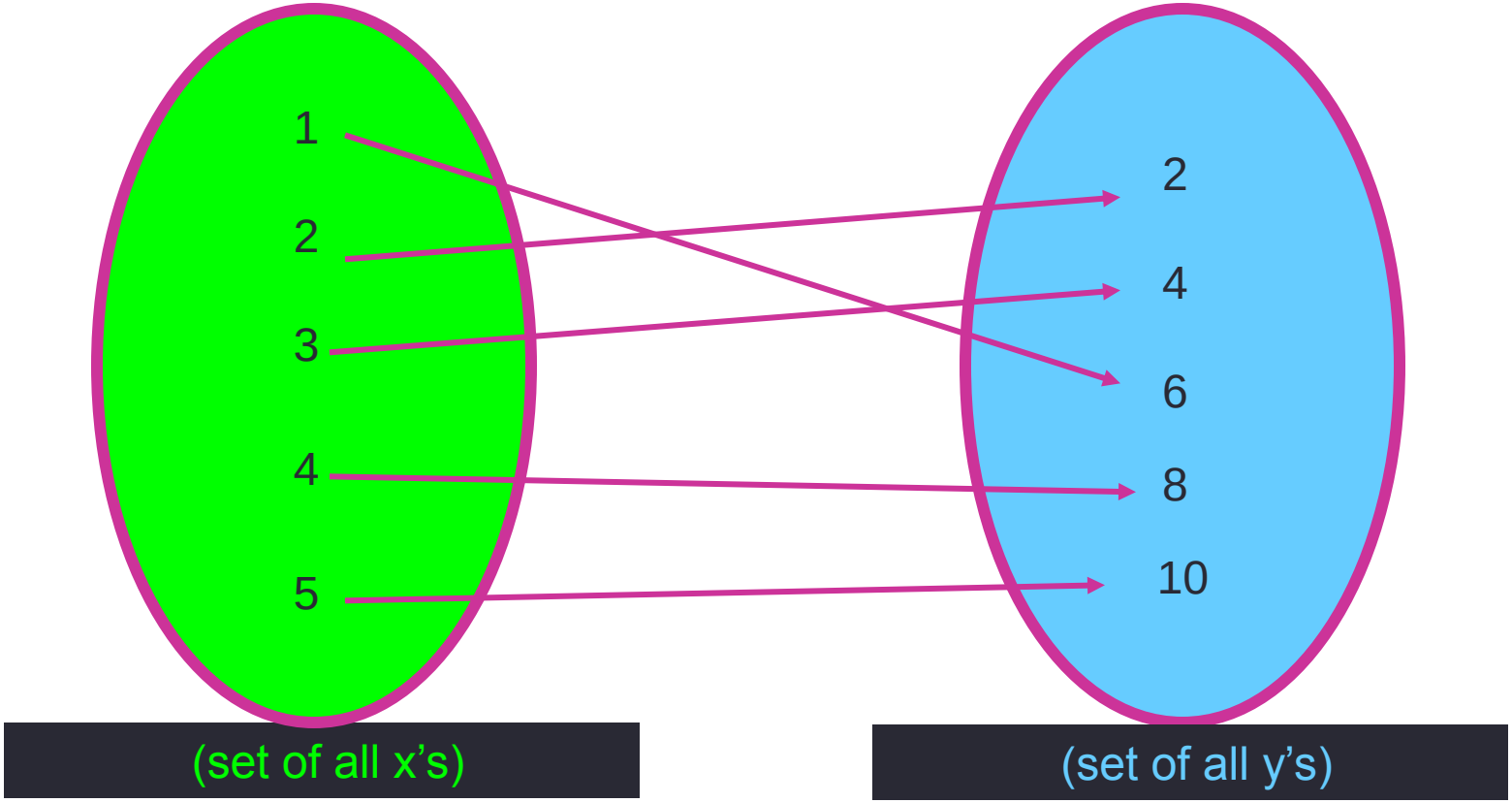
Introduction to Relation

- A **relation** is a set of ordered pairs, (*a*, *b*), where *a* is related to *b* by some rule.



- If tuple <x,y> is in relation R, the notation is xRy.

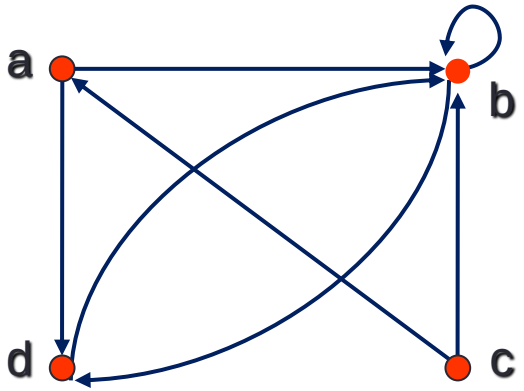
A relation assigns the x's with y's



This relation can be written as {<1,6>, <2,2>, <3,4>, <4,8>, <5,10>}.

Representing Relations Using Digraphs

- A directed graph, or digraph, consists of a set V of vertices(nodes), a set E of ordered pairs of elements of V called edges(arcs).
- a is called the initial vertex of the edge (a,b)
- b is called the terminal vertex of this edge
- An edge of the form (b,b) is called a loop.
- Example: Digraph with:
  - ✓ Vertices a, b, c, d;
  - ✓ Edges (a,b), (a,d), (b,b), (b,d), (c,a), (c,b), (d,b)



Properties of Relations

- *Reflexive*: Each element is related to itself.
  - **If** aRa , for all $a \in S$.
- *Symmetric*: If any one element is related to any other element, then the second element is related to the first.
  - **If** aRb **, then** bRa , for all $a,b \in S$.
- *Antisymmetric*: if a is related to b and b is related to a, then it must be the case that $a = b$.
  - **if** aRb **and** bRa **then** $a = b$ for all $a,b \in S$.
- *Transitive*: If any one element is related to a second and that second element is related to a third, then the first element is related to the third.
  - **If** aRb **and** bRc **then** aRc , for all $a,b,c \in S$.

Equivalence Relation

- R is an equivalence relation on set S if it is reflexive, symmetric and transitive.
- a and b are equivalent to each other  $a \equiv b$

Example 1

- For the integers, = is an equivalence relation that partitions each element into a distinct subset. In other words, for any integer a, three things are true.

  1. a = a,
  2. if a = b then b = a, and
  3. if a = b and b = c, then a = c.

- An equivalence relation *R* satisfies three properties:
  - ✓ Reflexive: *a R a* for all *a* in *S*.
  - ✓ Symmetric: *a R b* if and only if *b R a*.
  - ✓ Transitive: If *a R b* and *b R c* then *a R c*.

Example 2

- Modulus function defines an equivalence relation on the integers.
- Use the modulus function to define a binary relation such that two numbers x and y are in the relation if and only if x mod m = y mod m. (e.g, for m = 4, <1,5> is in the relation)
- This relation can be used to partition the integers into m equivalence classes. This relation is an equivalence relation because

  1. x mod m = x mod m for all x;
  2. if x mod m = y mod m, then y mod m = x mod m; and
  3. if x mod m = y mod m and y mod m = z mod m, then x mod m = z mod m.

Example 3

Consider the relation R on a set {1,2,3,4,5}.

- R = {(1,1), (1,3), (1,5), (2,2), (2,4), (3,1), (3,3), (3,5), (4,2), (4,4), (5,1), (5,3), (5,5)} is an equivalence relation because:
  - ✓ R is reflexive because (1,1), (2,2), (3,3), (4,4), (5,5) are in R.
  - ✓ R is symmetric because whenever (x,y) is in R, (y,x) is in R as well.
  - ✓ R is transitive because whenever (x,y) and (y,z) are in R, (x,z) is in R as well.
- Consider the relation R on a set {1,2,3,4}.
- R = {(1,1), (1,2), (1,3), (1,4), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)} is NOT an equivalence relation because R is not symmetric.

Miscellaneous Notation

- Units of Measure
  - B for byte
  - b for bits
  - KB for kilobytes (2^{10} = 1024 bytes)
  - MB for megabytes (2^{20} bytes)
  - GB for gigabyte (2^{30} bytes)
  - ms for milliseconds (1/1000 of a second)
  - 2000 bits (2Kb = 2048 bits)
- Factorial Function
  - n an integer greater than 0 is the product of the integers between 1 and n, inclusive.
  - 5! =1 · 2 · 3 · 4 · 5 = 120, 0! = 1
- Permutation
  - The members of S arranged in some order
  - A permutation of the integers 1 through n would be those values arranged in some order.
  - n distinct members, n! different permutations for the sequence

Miscellaneous Notation (Cont.)

- Boolean variables
  - ✓ Two values true and false (1 and 0)

- Floor and Ceiling
  - ✓ The floor of x ($\lfloor x \rfloor$): takes real value x and returns the greatest integer ≤ x.
    $\lfloor 3.4 \rfloor = 3$, $\lfloor -3.4 \rfloor = -4$ $\lfloor -3.0 \rfloor = -3$
  - ✓ Ceiling of x ($\lceil x \rceil$) takes real value x and returns the least integer ≥ x.
    $\lceil 3.4 \rceil = 4$, $\lceil -3.4 \rceil = \lceil -3.0 \rceil = -3$

- Modulus Operator
  - ✓ mod function returns the remainder of an integer division.
  - ✓ n mod m (Mathematical expressions)
  - ✓ n % m (C++ modulus operator)

-

Logarithms

- A logarithm of base b for value y $\Rightarrow$ x = $\log_b y$
- If x = $\log_b y$ then $b^x = y$, and $b^{\log_b y} = y$.
- Logarithms are used frequently by programmers.
- For any positive values m, n and r, and any positive integers a and b
  - ✓ $\log(nm) = \log n + \log m$
  - ✓ $\log(n/m) = \log n - \log m$
  - ✓ $\log (n^r) = r \log n$
  - ✓ $\log_a n = \log_b n / \log_b a$

Example 4

- QUES: Many programs require an encoding for a collection of objects. What is the minimum number of bits needed to represent n distinct code values?
- ANS: $\lceil \log_2 n \rceil$ bits
- For example, if you have 1000 codes to store, you will require at least $\lceil \log_2 1000 \rceil = 10$ bits to have 1000 different codes (10 bits provide 1024 distinct code values).

Example 5

- QUES: Consider the binary search algorithm for finding a given value within an array sorted by value from lowest to highest. Binary search first looks at the middle element and determines if the value being searched for is in the upper half or the lower half of the array. The algorithm then continues splitting the appropriate subarray in half until the desired value is found. How many times can an array of size n be split in half until only one element remains in the final subarray?
- ANS: $\lceil \log_2 n \rceil$ times

Summations

- Most programs contains with loop constructs.
- When analyzing running time costs for programs with loops, we need to add up the costs for each time the loop is executed.
- Summations are simply the sum of costs for some function applied to a range of parameter values.

$$\sum_{i=1}^{n} f(i)$$

Summations

- Recurrence Relation
- A function by means of an expression that includes one or more (smaller) instances of itself.
- A classic example (Factorial Function)
  - ✓ n! =n ·(n-1)! For n>1;
  - ✓ 1! = 0! =1.
- Another standard example (Fibonacci Sequence)
  - ✓ Fib(n) = Fib(n-1) + Fib (n-2) for n>2;
  - ✓ Fib(1) = Fib(2) =1.
  - ✓ 1,1,2,3,5,8,13,…

Contd.

Example 6

- If we expand the recurrence T(n) =T(n-1)+1, for n>1, T(0)=T(1)=0
- T(n) = T(n-1) + 1

= (T(n-2) + 1) + 1 (substitude n=n-1)

= T(n-2) + 2

if we expand the recurrence again,

T(n) = T(n-2) + 2 = T(n-3) + 3 = …….

T(n) = T(n-i) + i

To conclude,

T(n) = T(n-(n-1)) + (n-1) (substitude i=n-1)

= T(1) + n-1

= n-1

Contd.

Example 7

- A slightly more complicated recurrence is

  $T(n) = T(n-1) + n; \qquad T(1) = 1.$

- Expanding this recurrence a few steps,

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

  $= T(n-2) + (n-1) + n$

  $= T(n-3) + (n-2) + (n-1) + n, \ldots..$

This recurrence appears to have a pattern:

$T(n) = T(n-(n-1)) + (n-(n-2)) + \cdots + (n-1) + n$

$= 1 + 2 + \cdots + (n-1) + n,$

equivalent to $\sum_{i=1}^{n} i$

Contd.

Recursion

- Recursion
  - ✓ A function calls itself
- Base case
  - ✓ Handles a simple input that can be solved without resorting a recursive call
- Recursive Case (Inductive Case)
  - ✓ Calls the function again, but on different arguments
  - ✓ Formally, the arguments in recursive calls must be "smaller" in some way (shorter list, closer to zero, etc.)

Recursion

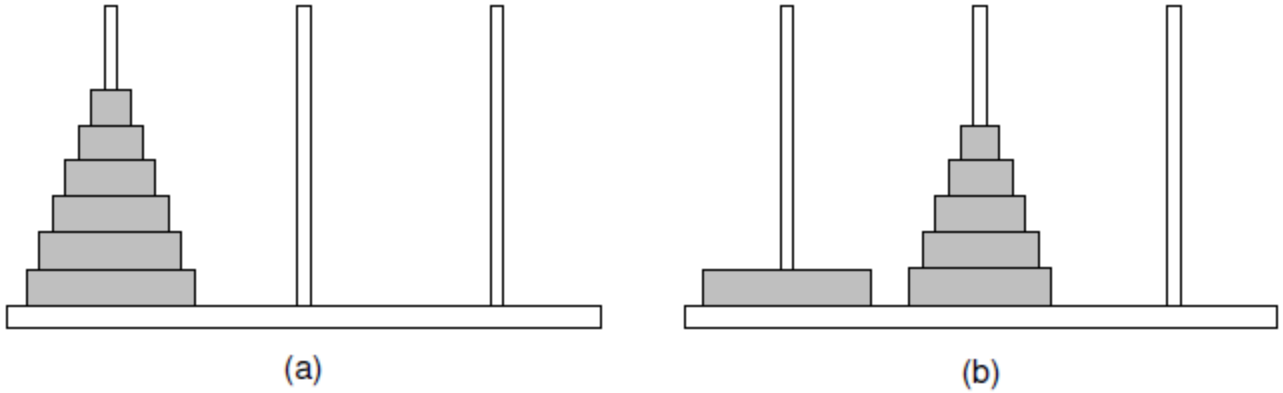


Figure: Towers of Hanoi example.
(a) The initial conditions for a problem with six rings.
(b) A necessary intermediate step on the road to a solution.

- Moving of disks from towers of hanoi should follow the following rules:
  - ✓ Only one disk can be moved at a time.
  - ✓ A disk can only be moved if it is the uppermost disk on a stack.
  - ✓ No disk may be placed on top of a smaller disk.
- Number of moves = $2^n - 1$; n= number of disks

Contd.

Mathematical Proof Techniques

- Solving any problem
  - ✓ Investigation: engage the problem, and working through until finding a solution
  - ✓ Argument: give the answer to the client and need to be able to make the argument in a way that gets the solution, involves good technical writing skills
- Proof Techniques
  - ✓ Direct Proof
  - ✓ Proof by Contradiction
  - ✓ Mathematical Induction

Direct Proof

- Logical explanation
  - ✓ logic notation $P \Rightarrow Q$
  - ✓ written in English with words such as "if ... then,"

- Eg. To prove P and Q are equivalent
  - ✓ First prove $P \Rightarrow Q$ and then $Q \Rightarrow P$

Direct Proof

Prove: If n is an odd integer, then n^2 is an odd integer.

Proof:

Suppose n is an odd integer, exists a such that $n = 2a + 1$.

$$n^2 = (2a + 1)^2$$
$$= 4a^2 + 4a + 1$$
$$= 2(a^2+2a) + 1$$

for $b = a^2+2a$,

$$n^2 = 2b + 1 \text{ (odd)}$$

so n is odd --> n^2 is odd

Contd.

Proof By Contradiction

- Disprove a theorem or statement
  - ✓To find a counterexample
- To prove a theorem by contradiction, first assume that the theorem is false, then find a logical contradiction stemming from this assumption.
- The assumption that the theorem is false must be incorrect. That is, conclude that the theorem must be true.

- Eg. To prove $P \Rightarrow Q$ by proofing (not Q) $\Rightarrow$ (not P).

Proof By Contradiction

- Theorem 2.1 Theorem is no largest integer.

- Proof: Proof by contradiction.
  - ✓ Step1. Contrary assumption: Assume that there is a largest integer. Call it B (for "biggest").
  - ✓ Step 2. Show this assumption leads to a contradiction:
  - ✓ Consider C = B +1. C is an integer because it is the sum of two integers. Where C > B.

Contd.

Proof by Mathematical Induction

- Mathematical Induction
  - ✓ Provides a useful way to think about algorithm design
  - ✓ Solves a problem by building up from simple subproblems
  - ✓ The same process as recursion

- Let Thrm be a theorem to prove, and express Thrm in terms of a positive integer parameter n.

- Mathematical induction states that Thrm is true for any value of parameters n (for $n \geq c$, where c is some constant)

  1. Base Case: Thrm holds n=c,
  2. Induction step: If Thrm holds for n-1,

     then Thrm holds for n.

Proof by Mathematical Induction

Example 8: Here is a sample proof by mathematical induction.
Call the sum of the first n positive integers S(n).

Theorem 2.2 $S(n) = n(n+1)/2$.

Proof: The proof is by mathematical induction.

1. **Check the base case.** For $n = 1$, verify that $\mathbf{S}(1) = 1(1+1)/2$. $\mathbf{S}(1)$ is simply the sum of the first positive number, which is 1. Because $1(1+1)/2 = 1$, the formula is correct for the base case.
2. **State the induction hypothesis.** The induction hypothesis is

$$\mathbf{S}(n-1) = \sum_{i=1}^{n-1} i = \frac{(n-1)((n-1)+1)}{2} = \frac{(n-1)(n)}{2}.$$

Contd.

Proof by Mathematical Induction

3. **Use the assumption from the induction hypothesis for $n-1$ to show that the result is true for n.** The induction hypothesis states that $\mathbf{S}(n-1) = (n-1)(n)/2$, and because $\mathbf{S}(n) = \mathbf{S}(n-1) + n$, we can substitute for $\mathbf{S}(n-1)$ to get

$$\begin{aligned}\sum_{i=1}^{n} i &= \left(\sum_{i=1}^{n-1} i\right) + n = \frac{(n-1)(n)}{2} + n \\ &= \frac{n^2 - n + 2n}{2} = \frac{n(n+1)}{2}.\end{aligned}$$

Thus, by mathematical induction,

$$\mathbf{S}(n) = \sum_{i=1}^{n} i = n(n+1)/2.$$

Contd.

Proof by Mathematical Induction

Theorem 2.3 $\sum_{i=1}^{n}(2i-1) = n^2$.

Proof: The base case of $n = 1$ yields $1 = 1^2$, which is true. The induction hypothesis is

$$\sum_{i=1}^{n-1}(2i-1) = (n-1)^2.$$

$$\begin{aligned}
\sum_{i=1}^{n}(2i-1) &= \left[\sum_{i=1}^{n-1}(2i-1)\right] + 2n - 1 \\
&= [(n-1)^2] + 2n - 1 \\
&= n^2 - 2n + 1 + 2n - 1 \\
&= n^2.
\end{aligned}$$

Thus, by mathematical induction, $\sum_{i=1}^{n}(2i-1) = n^2$.

Contd.

Proof by Mathematical Induction

Theorem 2.4 The recurrence relation $T(n) = T(n-1)+1; T(1) = 0$ has closed-form solution $T(n) = n-1$.

Proof:

- To prove the base case, we observe that $T(1) = 1-1 = 0$.
- The induction hypothesis is that $T(n-1) = n-2$.
- Combining the definition of the recurrence with the induction hypothesis, we see immediately that

$$T(n) = T(n-1) + 1 = n-2 + 1 = n-1 \text{ for } n > 1$$

Thus, we have proved the theorem correct by mathematical induction.

Contd.

Estimation

- One of the most useful life skills gained from the computer science is the ability to perform quick estimates.
- Estimation can be formalized by the following three-step process:

    1. Determine the major parameters that affect the problem.

    2. Derive an equation that relates the parameters to the problem.

    3. Select values for the parameters, and apply the equation to yield an estimated solution.

- Should decide on acceptable error bounds before doing an estimate

Next Week Lecture (Week 3)

Lecture 3: Algorithm Analysis

Thank you!