Data Structure

Lecture#4: Mathematical Preliminaries

U Kang
Seoul National University

In This Lecture

- Set concepts and notation
- Logarithms
- Summations
- **Recurrence Relations**
- **Recursion**
- **Induction Proofs**

Recurrence Relation

- Recurrence relation defines a function by an expression that includes itself
  - Factorial: $n! = (n-1)! * n$ for $n > 1$; $1! = 0! = 1$
  - Fibonacci sequence
    - $Fib(n) = Fib(n-1) + Fib(n-2)$ for $n > 2$; $Fib(1) = Fib(2) = 1$
- Closed form solution: solution without recurrence
  - E.g., Factorial $n! = n * (n-1) * \ldots * 1$
  - How to get a closed form solution? Expansion.

Closed Form for Recurrence

- $T(n) = T(n-1) + 1$ for $n > 1$; $T(0) = T(1) = 0$
  - Closed form solution: $T(n) = ?$

- $T(n) = T(n-1) + n$; $T(1) = 1$
  - Closed form solution: $T(n) = ?$

Recursion

- An algorithm is recursive if it calls itself to do part of its work.

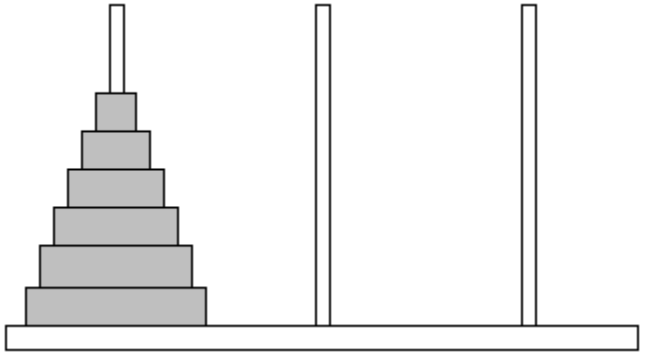
- Example:
  - 1. Compute n!
  - 2. Hanoi puzzle

Recursion: n!

```
/** Recursively compute and return n! */
static long fact(int n) {
  if (n <= 1)  return 1;   // Base case: return base solution
  return n * fact(n-1);    // Recursive call for n > 1
}
```

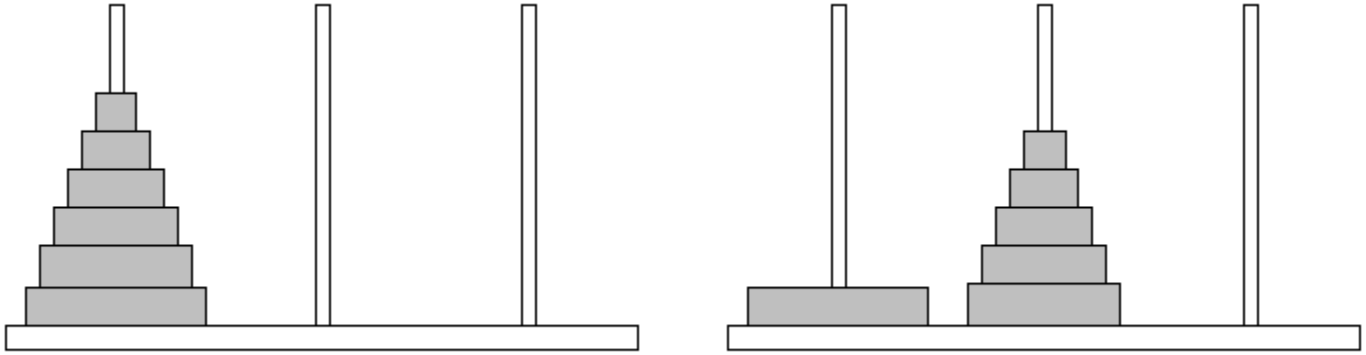
- When solving recursion, don't *worry* about how the recursive call solves the subproblem. Simply *accept* that it will solve it correctly.

Tower of Hanoi



- Problem: move the rings in the leftmost pole to the rightmost pole
  - A ring can never be moved on top of a smaller ring
- Key to solution: use *recursive* function

Tower of Hanoi



- Assume that a function X can move top n-1 rings to pole 2 (Fig (b))

```
/** Compute the moves to solve a Tower of Hanoi puzzle.
    Function move does (or prints) the actual move of a disk
    from one pole to another.
    @param n The number of disks
    @param start The start pole
    @param goal The goal pole
    @param temp The other pole */
static void TOH(int n, Pole start, Pole goal, Pole temp) {
  if (n == 0) return;           // Base case
  TOH(n-1, start, temp, goal); // Recursive call: n-1 rings
  move(start, goal);           // Move bottom disk to goal
  TOH(n-1, temp, goal, start); // Recursive call: n-1 rings
}
```

Mathematical Proof

- Three templates for mathematical proof
  1. Deduction
  2. Proof by Contradiction
  3. Proof by mathematical induction

Proof Technique 1 - Deduction

- Deduction
  - Argument in terms of logic
  - If we want to prove P => R, then we may prove P => Q, then Q=>R
  - If we want to prove that P and Q are equivalent, then we can first prove P => Q, and then Q=> P

Proof Technique 2 - Contradiction

- The simplest way to *disprove* a theorem: use counterexample!
  - E.g., everyone in the earth is male
- But, no number of examples supporting a theorem is sufficient to *prove* it
- Proof by contradiction (e.g., P => Q)
  - Assume the theorem is false (Q is false)
  - Then a logical contradiction stems from it
  - Thus we can conclude that the theorem is true

Proof Technique 2 - Contradiction

- Example
  - Theorem: No integer is the largest.
  - Proof
    - Assume x is the largest integer
    - y = x+1 is an integer, since it is the sum of two integers (x and 1)
    - But then, y is the largest integer which contradicts the assumption

Proof Technique 3 – Mathematical induction

- We want to prove that theorem X is true for all integers n >= c
  - Base case: prove that X holds for n = c
  - Induction step: If X holds for n-1, then X holds for n

    This is called "Induction Hypothesis"
  - An alternative induction step: If X holds for all c <= i <= n-1, then X holds for n

Proof Technique 3 – Mathematical induction

- E.g., Thrm: S(n) = 1+2+…+n = n(n+1)/2 for n >=1
  - Base case: 1 = 1*(1+1)/2
  - Induction Hypothesis (I.H.): S(n-1) = (n-1)n/2
  - Use I.H. to show that Thrm is true for S(n)
    - S(n) = S(n-1) + n = (n-1)n/2 + n = n(n+1)/2

Proof Technique 3 – Mathematical induction

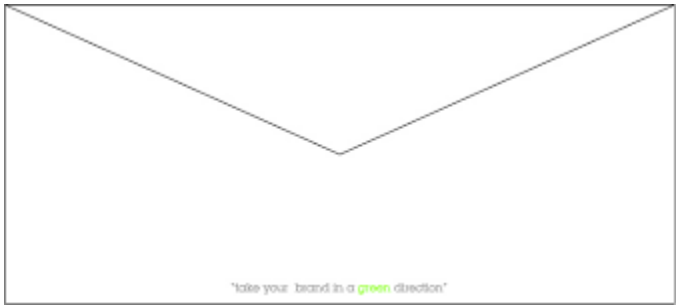
```
/** Recursively compute and return n! */
static long fact(int n) {
  if (n <= 1)  return 1;   // Base case: return base solution
  return n * fact(n-1);    // Recursive call for n > 1
}
```

- E.g., Thrm: function fact(n) will terminate for any n>=1
  - Base case: fact(1) terminates (returns 1)
  - I.H. : fact(n-1) terminates
  - Use I.H. to show that Thrm is true for n
    - fact(n) multiplies n with fact(n-1), and thus it will terminate as well

Estimation Techniques

Known as “back of the envelope” or “back of the napkin” calculation

1. Determine the major parameters that effect the problem.
2. Find an equation that relates the parameters to the problem.
3. Select values for the parameters, and apply the equation to yield estimated solution.



Estimation Example

How many library bookcases does it take to store books totaling one million pages?

Estimate:

- A 500-pages book requires ~ 2.5 cm
- => 1 million pages require ~ 5000 cm = 50 m of shelf space
- => A shelf is ~ 1.2 m wide, thus ~ 42 shelves are needed
- => A bookcase contains 5 shelves, thus 9 library bookcases needed

What you need to know

- Recurrence
  - Express problems using recurrence relations
  - Find closed form solution for recurrence
  - Write functions with recursion
- Mathematical proof
  - Deduction, by contradiction, by mathematical induction
  - Prove using contradiction
  - Prove using mathematical induction
- Be familiar with estimation techniques

Questions?