CS171 Introduction to Computer Science II

Recursion

Li Xiong
Recursion

• Recursion concept
• Examples
  – Factorial
  – Fibonacci
  – Recursive graph Htree
• Divide and conquer programming technique
  – Binary search
  – Tower of Hanoi
• Cost analysis of recursive algorithms
Recursion - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Recursion
Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested ...

Did you mean: recursion

Recursion (computer science) - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Recursion_(computer_science)
Recursion in computer science is a method where the solution to a problem depends on solutions to smaller instances of the same problem. The approach can ...

Recursion -- from Wolfram MathWorld
mathworld.wolfram.com
A recursive process is one in which objects are defined in terms of other objects of the same type. Using some sort of recurrence relation, the entire class of ...
What is recursion?
What is recursion? When one function calls itself directly or indirectly.

Why learn recursion?
- A new mode of thinking.
- Powerful programming paradigm.

Many computations are naturally self-referential.
- Mergesort, FFT, gcd, depth-first search.
- Linked data structures.
- A folder contains files and other folders.

Closely related to mathematical induction.
Factorial

\[ N! = N \times (N-1) \times (N-2) \times \ldots \times 2 \times 1 \]

```c
int fact (int N)
{
    if (N==0)
        return 1;
    else
        return (N * fact (N-1));
}
```
Recursive Method

• A method that calls itself (direct recursion)

```java
void recursiveMethod() {
    ... ...
    recursiveMethod();
}
... ...
```
Recursive Method

- A method that calls itself (direct recursion)
- Every recursive method must have a base case that is not recursive

```java
void recursiveMethod() {
    ...
    if (base case) {
        ...
    }
    else {
        ...
        recursiveMethod();
        ...
    }
}
```
Better version of recursion definition

Recursion
n. If you still don't get it, see Recursion.
Recursion

• A method calls itself
  – Calls a “clone” of itself to solve a **smaller** problem
  – Buck Passing

• Must have a base case
  – The buck stops here! (does **not** call the method)
Example: Fibonacci Numbers

- Recursive formula:

\[ F(n) = F(n-1) + F(n-2) \]
\[ F(0) = 0, \quad F(1) = 1 \]

- 0, 1, 1, 2, 3, 5, 8, 13, ....
Fibonacci Numbers: Java Code

```java
int F(int n)
{
    if (n==0)
        return 0;
    else if (n==1)
        return 1;
    else
        return F(n-1)+F(n-2);
}
```
Visual Recursion
Fractals
**H-tree of order n.**

- Draw an H.
- Recursively draw 4 H-trees of order n-1, one connected to each tip.
public class Htree {
    public static void draw(int n, double sz, double x, double y) {
        if (n == 0) return;
        double x0 = x - sz/2, x1 = x + sz/2;
        double y0 = y - sz/2, y1 = y + sz/2;
        StdDraw.line(x0, y, x1, y);
        StdDraw.line(x0, y0, x0, y1);
        StdDraw.line(x1, y0, x1, y1);
        draw(n-1, sz/2, x0, y0);
        draw(n-1, sz/2, x0, y1);
        draw(n-1, sz/2, x1, y0);
        draw(n-1, sz/2, x1, y1);
    }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        draw(n, .5, .5, .5);
    }
}
Recursion

- Recursion concept
- Examples
  - Factorial
  - Fibonacci
  - Recursive graph Htree
- Divide and conquer programming technique
  - Binary search
  - Tower of Hanoi
- Cost analysis of recursive algorithms
Divide and Conquer

- Solve two or more problems that are smaller than the original problem
- Solve original problem by solutions of smaller problems

Divide problem into smaller (easier) problems

Conquer: Write algorithm for each sub-problems
Recursion for Divide and Conquer

- Solve two or more problems that are smaller than the original problem
- To solve the smaller problem, divide it to even smaller problems until it can be solved as a base case
- Solve original problem by solutions of smaller problems
Binary Search in Ordered Array

- Compares the middle element with search key and reduces the search range by half in each iteration
Binary Search – Recursive solution

```java
public int find(long searchKey) {
    return recFind(searchKey, 0, nElems-1);
}

private int recFind(long searchKey, int lowerBound, int upperBound){
    int curIn;
    curIn = (lowerBound + upperBound) / 2;
    if(a[curIn]==searchKey) // found it
        return curIn;
    else if(lowerBound > upperBound) // can’t find it
        return nElems;
    else if(a[curIn] < searchKey) // upper half
        return recFind(searchKey, curIn+1, upperBound);
    else // lower half
        return recFind(searchKey, lowerBound, curIn-1);
}
```
Recursion vs. Iteration

• Recursion: calls itself one or more times until a condition is met
• Iteration: repeating for a specified number of times or until a condition is met
• Every recursive function can be transformed to an iterative function using a stack and vice versa
• Recursion is an elegant way to solve many practical problems but usually sacrifices memory and computational efficiency
The Towers of Hanoi

Move all the discs from the leftmost peg to the rightmost one.
- Only one disc may be moved at a time.
- A disc can be placed either on empty peg or on top of a larger disc.

Is world going to end (according to legend)?
- 40 golden discs on 3 diamond pegs.
- World ends when certain group of monks accomplish task.

Edouard Lucas (1883)
The Towers of Hanoi

• How do we move the disks to achieve the goal?

• We also want to know in general, how many steps it takes to move N disks.

• Let’s play with it to get some intuition:
  – N=1, N=2, N=3, N=4

http://www.mazeworks.com/hanoi/
The Towers of Hanoi

- Let’s call the initial pyramid-shaped arrangements of disks on column A a tree.
- We call a smaller set of the disks a subtree.
- It turns out that the intermediate steps in the solution involves moving a subtree.
The Towers of Hanoi

- Idea:
  - Assume, for now, that you have a (magical) way of moving a subtree from A to B via C;
  - Then you move A to C;
  - Finally move the subtree from B to C via A.
  - This is similar to the 2-disk case, except the top disk is now a subtree.

- How can we move the subtree?
- What’s the base case?
a) Move subtree to B

b) Move bottom disk to C

c) Move subtree to C

d) Puzzle is solved!
The Towers of Hanoi

• Suppose you want to move \( n \) disks from a source tower \( S \) to a destination tower \( D \), via an intermediate tower \( I \).

• Initially
  – Source tower is \( A \)
  – Destination tower is \( C \)
  – Intermediate tower is \( B \)
The Towers of Hanoi

1. Move the subtree consisting of the top $n-1$ disks from S to I;
2. Move the remaining (largest) disk from S to D;
3. Move the subtree from I to D.
Towers of Hanoi: Implementation

```java
public class Hanoi {

    static void hanoi(int n, char from, char temp, char to) {
        if (n == 0) return;
        hanoi(n-1, from, to, temp);
        System.out.println("Move disc " + n);
        System.out.println(" from " + from + " to " + to);
        hanoi(n-1, temp, from, to);
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        hanoi(N, 'A', 'B', 'C');
    }
}
```
The Tower of Hanoi

• How many steps does the solution take to solve a N-disk problem?

• N=1:
• N=2:
• N=3:
• N=4:

• When is the world going to end (N=64)?
The Tower of Hanoi

• How many steps does the solution take to solve a N-disk problem?
  • N=1: 1 step
  • N=2: 3 steps
  • N=3: 7 steps
  • N=4: 15 steps

• When is the world going to end (N=64)?
Using Recurrence Relation for Cost Analysis

- Define the cost function $T(N)$ using recurrence relation and define base cases
- Solve the recurrence relation
- Derive Big O function
Recurrence Relation

• A recurrence relation is an equation that recursively defines a sequence: each term of the sequence is defined as a function of the preceding terms

• Examples:
  – $T(n) = T(n-1) + 1$
Solving Recurrence Relations

• Rewrite $T(N)$, $T(N-1)$, $T(N-2)$, ..., with the recurrence formula
• Discover the patterns and find an expression
• Check the correctness
  – Substitute solution in initial conditions
  – Substitute solutions in the recurrence relation
Example

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

- Expansion
  - \( T(n) = T(n-1) + 1 = T(n-2) + 1 + 1 = T(n-3) + 1 + 1 + 1 = \ldots \)

- Discover pattern and solution
  - \( T(n) = T(1) + n - 1 = n \)

- Verification
  - \( T(1) = 1 \)
  - \( T(n) = T(n-1) + 1 \)
Binary Search Example

- Binary Search Cost Function
- $T(n) = T(n/2) + 1$
- $T(1) = 1$
Binary Search Example: Solution

- Binary Search Cost Function
- $T(n) = T(n/2) + 1$
- $T(1) = 1$

- $N=2^k$
- $T(2^k) = T(2^{k-1}) + 1 = T(2^{k-2}) + 1 + 1 + \ldots$
- $T(2^k) = T(2^0) + k = 1 + k$
- $T(N) = 1 + \lg N$
The Tower of Hanoi

• How many steps does the solution take to solve a N-disk problem?
• Use recursive formula
  \[ T(N) = T(N-1) + 1 + T(N-1) = 2 \times T(N-1) + 1 \]
• So how to solve this?

\[
T(N) = 2^{N-1} + 2^{N-2} + 2^{N-3} + \ldots + 1 = 2^N - 1
\]

• So it takes an exponential number of steps!
When is the world going to end?

- Takes 585 billion years for $N = 64$ (at rate of 1 disc per second).
Summary

• Recursion: powerful tool allows for elegant solutions
  – But, computational overhead is high
• Cost analysis: Recurrence relations