Logistics

- **Homewroks 2 due tonight @ 11:59pm**
- **Homeworks 3 released, due at Oct 21st 11:59pm**

Logistics

Midterm Exam: Next Monday, Oct 14th 2024

- 4:50pm - 6:05pm @ MCH 201
- 10 multiple-choice questions (20 pts)
- 10 short answer questions (30 pts)
- 3 programming questions (50 pts)
- Paper exam, closed-book, no cheat sheet, no electronic devices (phone, tablet, laptop, calculator *etc.*)
- Covers from Introduction to C++ to Advanced Fucntions (include unix we have learned so far)

Lecture 12

Advanced Functions - Recursion

Shibo Li

shiboli@cs.fsu.edu

Department of Computer Science
Florida State University

What is Recursion

- A recursive function is a function that calls **itself** in order to solve a smaller instance of the same problem.
- A problem is divided into smaller **sub-problems** until it reaches a **base case**, which is directly solvable.

Components of a Recursive Function

- **Base Case:** The condition under which the recursion ends. It prevents infinite recursion.
- **Recursive Case:** The part of the function that calls itself with a modified parameter, moving the problem closer to the base case.

```
function recursiveFunction(parameters) {
   if (base case condition)
      return base_case_value;
   else
      return recursiveFunction(modified_parameters);
}
```

Why Recursions

- **Simpler Code for Certain Problems:** Recursive solutions can be more intuitive and shorter for problems like factorials, Fibonacci sequence, and tree traversals.
- **Divide and Conquer Approach:** Recursion naturally fits problems that can be divided into similar sub-problems (e.g., merge sort, quick sort).

Example

- Calculate the factorial of a number n.
- $n! = n \times (n-1)!$, where $0! = 1$

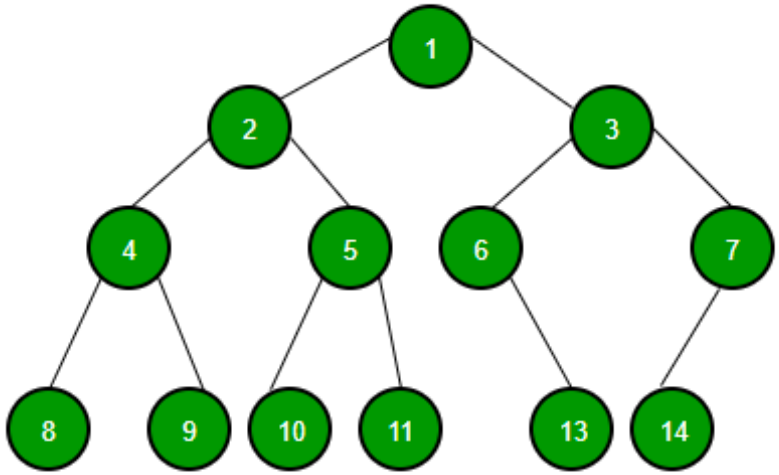
```
int factorial(int n) { // recursion
   if (n == 0) // Base case
      return 1;
   else
      return n * factorial(n - 1); // Recursive case
}
```

```
int factorial(int n) { // iteration
   int result = 1;
   for (int i = 1; i <= n; ++i) {
      result *= i;
   }
   return result;
}
```

Pros and Cons of Recursion

Pros:

- Simplicity and readability for problems that fit naturally into recursive patterns.
- Useful for tasks that involve tree structures or backtracking.



Pros and Cons of Recursion

Cons:

- Higher memory usage due to call stack overhead.
- Can lead to stack overflow if the recursion depth is too high (possible exponential growth).
- Slower performance for problems with many overlapping subproblems

Pros and Cons of Recursion

Cons:

- Higher memory usage due to call stack overhead.
- Can lead to stack overflow if the recursion depth is too high (possible exponential growth).
- Slower performance for problems with many overlapping subproblems

Avoiding Pitfalls

- **Base Case Issues:** Ensure a proper base case is defined to prevent infinite recursion.
- **Performance Considerations:** Consider using dynamic programming or memoization

Real-world Applications of Recursion

- Sorting Algorithms (Quick Sort, Merge Sort)
- Tree Traversal (Pre-order, In-order, Post-order)
- Graph Traversal (Depth-First Search)
- Solving Puzzles (e.g., Towers of Hanoi)