Tutorial 7
Recursion

Jonathan Kristian Tjandra (USTF)
(SDS, 122040024)

Concept

- Very powerful concept and can be used in many scenarios
- Divides problems into sub-problems **of the same form** (can be solved using the same method or function)
- In implementation, usually the function will call itself until the solution is found
- Example:

```
int linearSum(int a){
        if(a == 1) return 1;
        return a + linearSum(a-1);
}
```

How does it work?

- In every recursion, there will be **base case(s)**.
- Base cases are very important since these are the ones preventing the recursion from infinitely looping
- Recursion function’s parameters must converge to the base cases

Recursion Example

Description: John wants to fill a bottle with the size of n liters using 1-liter, 2-liter, and 3-liter bottles. N is guaranteed to be at least 1 liter. How many possible ways to fill the entire bottle with the 3 bottles that John has?

Think recursively:

- Recursion sub-problem: How to fill a bottle with size n liters using 1-liter, 2-liter, and 3-liter bottles
- Designing the recursion method/function: **How?**
- **What to put in return? What to use as base cases?**

Designing Recursion Method

In each step, we can do the following operations:

1. Choose one of the 1L, 2L, and 3L bottle
2. Fill the big bottle with the chosen bottle
3. Find the number of ways to fill the rest of the bottle

Let F(n) be the function that computes all the possible way to fill a bottle of size n with 1L, 2L, and 3L bottles. Suppose we choose the 1L bottle in the first step. The number of possible ways to fill the rest of the bottle is **F(n-1)**

If we choose 2L, then the number of possible ways to fill the rest of the bottle is F(n-2)

If we choose 3L, then it is F(n-3)

Recursive Function

We need to combine all the information we have above

Recursive function:

return F(n-1) + F(n-2) + F(n-3)

Base cases:

F(0) = 0

F(1) = 1

F(2) = 2

Final Function

```
int FillBottle(int n){
        if(n == 0) return 0;
        if(n == 1) return 1;
        if(n == 2) return 2;
        return FillBottle(n – 3) + FillBottle(n – 2) + FillBottle(n – 1);
}
```

Mutual Recursion

- Recursion that involves **more than one** function
- Example: Hofstadter Female and Male sequences
- For most cases, this is inefficient and unnecessary (there are solutions that don't use mutual recursion and are better/easier to read)

```
bool isOdd(unsigned int n) {
   return !isEven(n);
}

bool isEven(unsigned int n) {
   if (n == 0) {
      return true;
   } else {
      return isOdd(n - 1);
   }
}
```

```
bool isOdd(unsigned int n) {
   if(n == 0) return false;
   if(n == 1) return true;
   return isOdd(n - 2);
}
```

Recursive Strategies

- For some problems, the recursive function might not be so clear at first sight
- Try to play with the problem and see what pattern shows up
- Try starting from small problem size and expand to bigger numbers

Example:

Hanoi tower with n = 2 (move from A to C)

Solution:

Move from A to B → Move from A to C → Move from B to C

Recursive Strategies

Example:

Hanoi tower with n = 3 (move from A to C)

Solution:

Move from A to C → Move from A to B → Move from C to B → Move from A to C → Move from B to A → Move from B to C → Move from A to C

From this we can deduct general solution for the tower of Hanoi

Move n-1 disks to auxiliary pole, move the last disk to the target pole, and move n-1 disks from the auxiliary pole to the target pole

Inclusion-Exclusion Pattern

- Very common recursion strategy
- Use case: finding subsets
- Recursion method: Find the solution that includes an element and also find the solution that excludes that element

Case Example 1: Binary Conversion

Description: Given an integer, you should output the binary conversion of said integer

Input format: First line is the number of test cases. For each line after that, the input is an integer that you need to convert

Example input & output:

Input:	**Output:**
2 2 7	10 111
3 0 1 73	0 1 1001001

Case Example 2: Odd-Even Factorial

Description: Description: John has found a new sequence, named the odd-even factorial. To calculate this sequence, the formula is as follows. Let f(x) denote the factorial of x (x!) and f'(x) denote the odd-even factorial of x. If x is an odd number, then f'(x) = f(x) (the odd-even factorial is the same as the regular factorial). If x is an even number, then f'(x) = f(x)/2 (the odd-even factorial is half of the regular factorial).

Example input & output:

Input:	**Output:**
2 5 0	30 0
3 1 6 10	1 90 113400

Case Example 3: Reverse Linked-List

Description: Given a linked list (list1). Your task is to reverse the linked list. Use recursion to solve this problem.

No need to read input (will be stored in a variable in the main program manually)

Examples:

Input:	**Output:**
[1, 2, 3, 4, 5]	5 -> 4 -> 3 -> 2 -> 1
[2, 3]	3 -> 2

Q&A