School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

Recursion

Overview

- In this lecture we will discuss:
  - How to improve the efficiency of recursion
    - Tail recursion
    - Memoization
    - Helper functions

Think Recursively

- Many of the problems we talked before can be solved using recursion if we think recursively.

- Consider the palindrome problem in prac 1.

Example

```
bool isPalindrome(string s){
        //base
        if(s.length()<=1)
                return true;
        //remove non-alphabet
        if(!isalpha(s[0]))
                return isPalindrome(s.substr(1,s.length()));
        if(!isalpha(s[s.length()-1]))
                return isPalindrome(s.substr(0,s.length()-1));
        //recursion
        if(tolower(s[0]) != tolower(s[s.length()-1]))
                return false;
        return isPalindrome(s.substr(1,s.length()-2));
}
```

Recursive Helper Functions

- This implementation of isPalindrome() is not efficient. Why?
  - It creates a new string for every recursive call
  - What about checking whether a substring is a palindrome or not?

- It is a common design technique in recursive programming to declare a second function that receives additional parameters.

```
int isPalindrome(string s, int start, int end)
```

Example

```
bool isPalindrome(string s){
		isPalindromHelper(s, 0, s.length-1);
}

bool isPalindromeHelper(string s, int start, int end){
		//base
		if(end==-1 || start=end)
				return true;
		//remove non-alphabet
		if(!isalpha(s[start+0]))
				return isPalindromeHelper(s, start+1, end);
		if(!isalpha(s[end]))
				return isPalindromeHelper(s,start, end-1);
		//recursion
		if(tolower(s[start]) != tolower(s[end]))
				return false;
		return isPalindromeHelper(s,start+1, end-1);
}
```

Stack use for recursive isPalindrome

- How does it work?

Hanoi Tower

- The Towers of Hanoi problem can be solved easily using recursion, but is difficult to solve without using recursion.

- The problem involves moving a specified number of disks of distinct sizes from one tower to another while observing the following rules:
  - Only one more tower can be used other these two towers
  - No disk can be on top of a smaller disk at any time
  - All the disks are initially placed on one tower
  - Only one disk can be moved at a time and it must be the top disk on the tower.

Summary

- Recursion is a useful tool for understanding problems and producing solutions, but:
  - You can always solve it iteratively
  - It can be inefficient and space hungry
  - Analysing recursive code can get tricky quickly

We hope you enjoy this course