ACA Semester Project
Task List

Anuj Nagpal

January 29, 2017

Algorithms

Binary Trees

I hope you have studied the lecture slides on binary trees and found it pretty interesting. As promised, here is a problem on binary tree:
Given a Binary tree and a sum S, print all the paths, starting from root, that sums upto the given sum. Note that here path doesnt need to end on a leaf node. Code your solution in C++. Here are some sample test cases:

```
Input : sum = 8,
        Root of tree
          1
        /   \
      20     3
            /   \
           4     15
          /  \   /  \
         6    7 8    9
Output :
Path: 1 3 4

Input : sum = 38,
        Root of tree
          10
        /    \
      28      13
             /   \
           14     15
          /  \   /  \
         21  22 23   24
Output :
Path found: 10 28
Path found: 10 13 15
```

Stacks

Ever thought how your computer parses and evaluates a expression like:

$$3 + (2 * 3 + x = 5 + (x + y)) + 5 * x + y$$

Time to learn another data structure. Go through lecture 10 and lecture 11 which are there in the onedrive folder (You can ignore the initial binary tree portion in lecture 10 and start from the slide where stack portion begins). Once you do that, you will try to implement an extension of algorithm for arithmetic expression evaluation. I acknowledge Prof. Surender Baswana for this problem. It will seem tough initially but you will have fun solving it :) Please avoid use of STL:

We know that $3 + 5^6$ is an arithmetic expression whereas neither 3*+6-5 nor 2*(3+5)) is an arithmetic expression. Can we give a precise definition of an arithmetic expression? Think over it. Indeed it is possible through recursion. The following is a recursive definition of an arithmetic expression:

- Each number is an arithmetic expression and its value is the same as the number. Likewise, a variable of type number is also an arithmetic expression and its value is the latest value assigned to it.
- if α is an arithmetic expression then (α) is also an arithmetic expression whose value is the same as the value of α.
- if α and β is an arithmetic expression, then $\alpha o \beta$ is also an arithmetic expression where o is any arithmetic operator (+ , - , * , / , $\wedge$).

The above definition can be used to determine if an expression is a valid arithmetic expression. However, as a careful reader might have noticed that the above definition of arithmetic expressions is ambiguous. Indeed, the above definition may lead to multiple possible values of an expression. However, if we specify all the precedence and associativity of the operators, each expression will have a unique value. [In the course on Compilers, you will study a more elegant mechanism of removing the ambiguity by defining the grammar of expression suitably]. We shall now extend the domain of arithmetic expression.

In addition to the usual binary operators (+, - , * , /, $\wedge$) We introduce an additional operator = called assignment operator. Its details are described as follows. If x is a variable and α is an expression, then $x = \alpha$ is also an expression. The value of expression $x = \alpha$ is defined as the value of expression α. The evaluation of $x = \alpha$ involves evaluation of α and assigning this value to variable x. Notice that = is right associative.

Design and implement an algorithm to evaluate a valid generic expression. The challenge lies in incorporating the assignment operator and variables in a seamless manner instead of resorting to case analysis. Your algorithm must make a single scan from left to

right. One subtle issue is the following. When you encounter a variable, how to detect whether this variable is to be considered like a number here or as a left hand side of an assignment expression. Think carefully: there is an easy way to detect it. Note that if a variable is to be considered as a number, its value is the latest value assigned to it in the past.

You may assume that only two variables, x and y are allowed in an expression. However, they may appear multiple times in an expression. The initial value of x as well as y is 1. Your output must consist of three lines: value of x in the first line, the value of y in the second line, and the value of the expression in the third line. You may assume that input expression is a valid arithmetic expression and it terminates with #.

Sample input and output:

1. input: 3+5^(2*3-(x+y))#
   output: 1 1 628
2. input: 3*6+2^x=2+(x+1)*(3-y)#
   output: 6 1 82
3. input: 100+2*x+x=2^4+3*y=3*(2+4^2)-22#
   output: 112 32 214.
4. input: 3+(2*3+x=5+(x+y))+5*x+y#
   output: 7 1 52

Intro to Graphs

Combinatorial graphs provide a natural way to model connections between different objects. They are very useful in depicting communication networks, social networks and many other kind of networks. Here, graphs mean combinatorial graphs as opposed to graphs of functions etc. which you might have learnt previously :D Graphs have become such an important tool that a complete field, Graph Theory, is devoted to learning about the properties of graphs. To study graph algorithms, you need to learn some basics of graph theory. Your Task: You can find lecture 22 in one drive - Please go through that