

1 Optimizations Comments

Instead, think of this: Fibonacci sequences are exhibited by dynamic programming. 1, 1, 2, 3, 5, 8, ... Each element is dependent on the two previous. These can be represented in #1: a recursive form, #2: in an array (like ICA6), or #3: in a closed-form as a single equation.

Would you rather consider implementing ICA6 as a recursive problem, like #1?
Would you rather consider implementing ICA6 as a closed form problem, like #3?

Which applications would benefit from one of the above two optimizations?

Please give me an optimization in one of those two ways.

$$F_i = \frac{\phi^i - (1-\phi)^i}{\sqrt{5}}$$
$$\phi = 1.61803...$$

1.1 Type #1: Recursion

Slow, little space usage.

$$T_i = T_{i-1} + T_{i-2}$$
$$T_0 = 1$$
$$T_1 = 1$$

```
int fib( int i )
{
    if( i == 0 )
        return 1;
    else if( i == 1 )
        return 1;
    else
        return fib( i-1 ) + fib( i-2 );
}
```

```
fib(5) = fib(4) + fib(3);
fib(4) = fib(3) + fib(2);
fib(3) = fib(2) + fib(1);
fib(2) = fib(1) + fib(0);
return 1;
return 1;
return 2;
return 1;
return 3;
fib(2) = fib(1) + fib(0);
return 1;
return 1;
return 2;
return 5;
fib(3) = fib(2) + fib(1);
fib(2) = fib(1) + fib(0);
return 1;
return 1;
return 2;
return 1;
return 3;
return 8;
```

Look at how many times we had to redundantly call fib(3), fib(2), fib(1), and fib(0).

1.2 Type #2: Dynamic Cache

Fast, large space usage.

1 1 2 3 5 8

```
fib(5) = fib(4) + fib(3);
fib(4) = fib(3) + fib(2);
fib(3) = fib(2) + fib(1);
fib(2) = fib(1) + fib(0);
return 1;
return 1;
return 2;
return 1;
return 3;
return 2;
return 5;
return 3;
return 8;
```

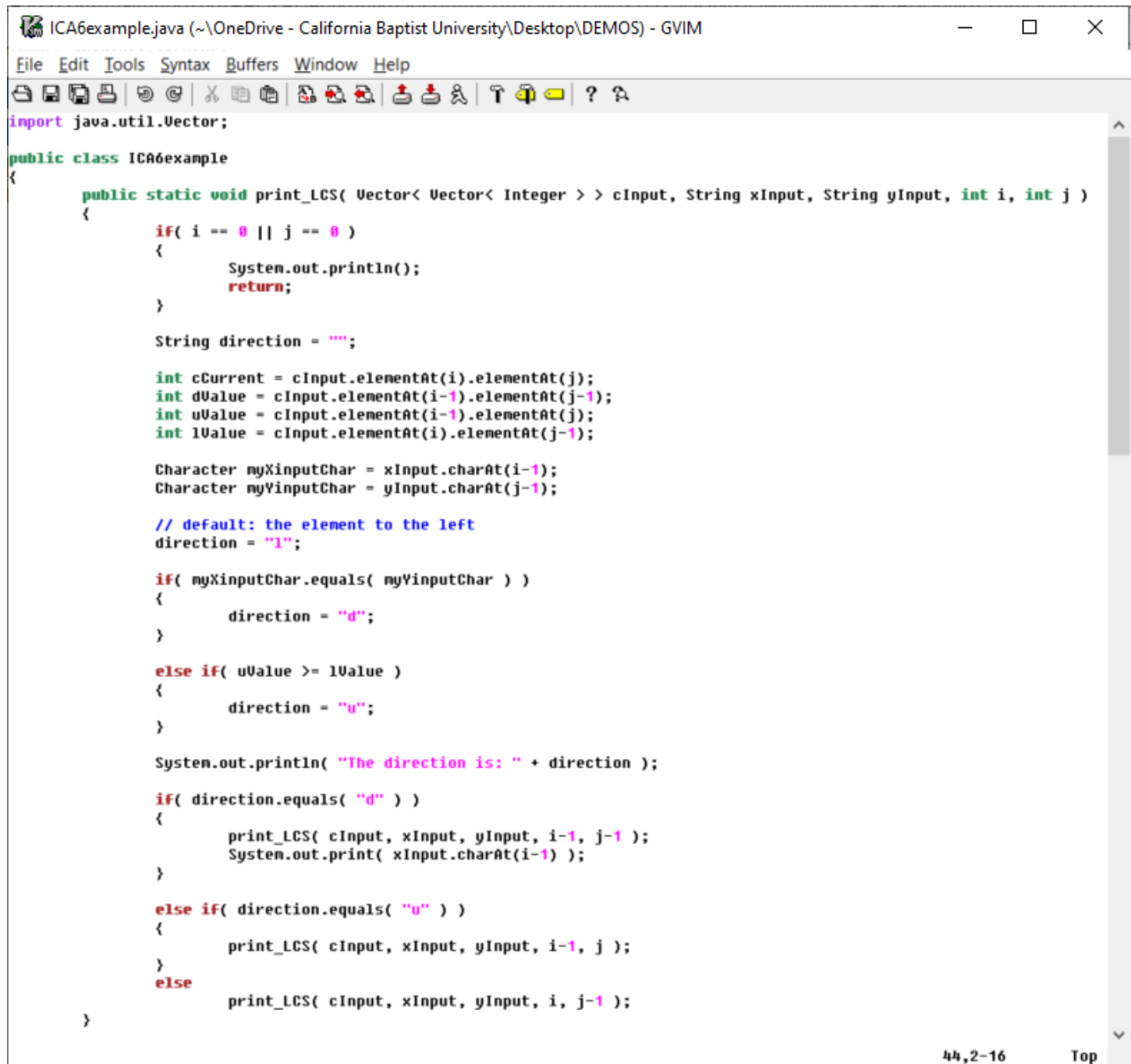
1.3 Type #3: Closed Form

Fast, little space usage, inaccurate for small numbers.

$$F_i = \frac{\phi^i - (1-\phi)^i}{\sqrt{5}}$$

$$\phi = 1.61803...$$

2 Constant Optimizations Example



```
ICA6example.java (~\OneDrive - California Baptist University\Desktop\DEMOS) - GVIM
File Edit Tools Syntax Buffers Window Help
import java.util.Vector;

public class ICA6example
{
    public static void print_LCS( Vector< Vector< Integer > > cInput, String xInput, String yInput, int i, int j )
    {
        if( i == 0 || j == 0 )
        {
            System.out.println();
            return;
        }

        String direction = "";

        int cCurrent = cInput.elementAt(i).elementAt(j);
        int dValue = cInput.elementAt(i-1).elementAt(j-1);
        int uValue = cInput.elementAt(i-1).elementAt(j);
        int lValue = cInput.elementAt(i).elementAt(j-1);

        Character myXinputChar = xInput.charAt(i-1);
        Character myYinputChar = yInput.charAt(j-1);

        // default: the element to the left
        direction = "l";

        if( myXinputChar.equals( myYinputChar ) )
        {
            direction = "d";
        }

        else if( uValue >= lValue )
        {
            direction = "u";
        }

        System.out.println( "The direction is: " + direction );

        if( direction.equals( "d" ) )
        {
            print_LCS( cInput, xInput, yInput, i-1, j-1 );
            System.out.print( xInput.charAt(i-1) );
        }

        else if( direction.equals( "u" ) )
        {
            print_LCS( cInput, xInput, yInput, i-1, j );
        }
        else
        {
            print_LCS( cInput, xInput, yInput, i, j-1 );
        }
    }
}
```

44,2-16 Top

ICA6example.java (~\OneDrive - California Baptist University\Desktop\DEMOS) - GVIM

File Edit Tools Syntax Buffers Window Help

```
public static void main( String[] args )
{
    String Xinput = "ABCBDAB";
    String Yinput = "BDCABA";

    Vector< Vector< Integer > > cTable = new Vector< Vector< Integer > > ();

    // Initialize the tables
    for( int i = 0; i <= Xinput.length(); i++ )
    {
        Vector< Integer > cRow = new Vector< Integer > ();

        for( int j = 0; j <= Yinput.length(); j++ )
        {
            cRow.addElement( 0 );
        }

        cTable.addElement( cRow );
    }

    // consider only C Table
    for( int i = 1; i < cTable.size(); i++ )
    {
        for( int j = 1; j < cTable.elementAt(i).size(); j++ )
        {
            Character xInputChar = Xinput.charAt(i-1);
            Character yInputChar = Yinput.charAt(j-1);

            if( xInputChar.equals( yInputChar ) )
                cTable.elementAt(i).set( j, cTable.elementAt(i-1).elementAt(j-1) + 1 );
            else if( cTable.elementAt(i-1).elementAt(j) >= cTable.elementAt(i).elementAt(j-1) )
                cTable.elementAt(i).set( j, cTable.elementAt(i-1).elementAt(j));
            else
                cTable.elementAt(i).set( j, cTable.elementAt(i).elementAt(j-1) );
        }
    }

    print_LCS( cTable, Xinput, Yinput, Xinput.length(), Yinput.length() );
}
}
```

57,2-16 Bot