CS 6810/7810: Wavelets and Wavelet Algorithms
Assignment 7
2D Ordered Inverse HWT

Vladimir Kulyukin
Department of Computer Science
Utah State University

March 25, 2017

Learning Objectives

1. 2D Ordered Inverse HWT

Introduction

This will be the last assignment on 2D HWT. You will implement the 2D Ordered Inverse HWT.

Problem

Extend your class `TwoDHWT.java` from Assignment 6 with the following static method that applies the 2D Ordered Inverse HWT to the 2D Haar Wavelet Transform of some 2D signal `sig_hwt`. The second argument `num_inv_iters` specifies the number of inverse iterations we will apply to `sig_hwt`. The third argument `num_fwd_iter` specifies the number of forward iterations of the 2D Ordered HWT to the original signal. The four argument `dbg_flag` is the argument I use to turn on or off various debugging messages.

```
TwoDHWT.ordInvDWTForNumIters(sig_hwt, num_inv_iters, num_fwd_iters, dbg_flag) {}
```

You can implement the following test method:

```
static void test_ordFwdInvHWT2(double[][] sig, int num_fwd_iters, int num_inv_iters, boolean dbg_flag) {
    TwoDHWT.ordFwdDWTForNumIters(sig, num_fwd_iters, dbg_flag);
    if ( dbg_flag ) {
       System.out.println("================");
       System.out.println("Forward HWT Transform");
       display2DArray(sig, sig.length, sig[0].length);
    }
    TwoDHWT.ordInvDWTForNumIters(sig, num_inv_iters, num_fwd_iters, dbg_flag);
    if ( dbg_flag ) {
        System.out.println("Inverse HWT Transform");
        display2DArray(sig, sig.length, sig[0].length);
    }
}
```

Create the class `WaveletAlgos_S17_HW07.java` with several 2D signals, as defined below. You can use these signals for debugging your implementation of the 2D Ordered Inverse HWT. Note that the method `TwoDHWT.ordFwdDWTForNumIters` has a third boolean argument set to `false`.

```
  public class WaveletAlgos_S17_HW07 {

     static double[][] signal_4x4 = {
        {9,7,6,2},
        {5,3,4,4},
        {8,2,4,0},
        {6,0,2,2}
     };
     static double[][] signal_8x8 = {
        {8,  5,  4,  8,  6, 8, 10,  8},
        {8, 10, 10,  4, 10, 4,  8,  2},
        {6, 10,  2,  4,  2, 6,  1,  6},
        {0,  8,  0, 10, 10, 6,  6, 10},
        {8,  8,  8,  0,  4, 8,  6,  2},
        {10, 6,  2,  2,  6, 6,  6,  8},
        {2,  4, 10, 10, 10, 4,  6, 10},
        {10, 6, 10,  6,  6, 4,  4,  4}
    };

    public static double[][] img_sample_16x16_01 = {
        {255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255}
    };

    public static double[][] img_sample_16x16_02 = {
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
    };
  }
```

Let us run a few tests. Here is the first test with its output right below it.

```
public static void main(String[] args) {
   test_ordFwdInvHWT2(signal_4x4, 2, 1, true);
}
```

```
================
Forward HWT Transform
4.0 1.0 1.0 1.0
1.0 0.0 3.0 1.0
2.0 0.0 0.0 1.0
1.0 0.0 0.0 1.0

Inverse HWT Transform
6.0 1.0 4.0 1.0
```

```
2.0 0.0 0.0 1.0
4.0 3.0 2.0 1.0
1.0 0.0 0.0 1.0
```

Here is the second test with its output right below it.

```
public static void main(String[] args) {
   test_ordFwdInvHWT2(signal_4x4, 2, 2, true);
}
```

```
================
Forward HWT Transform
4.0 1.0 1.0 1.0
1.0 0.0 3.0 1.0
2.0 0.0 0.0 1.0
1.0 0.0 0.0 1.0

Inverse HWT Transform
9.0 7.0 6.0 2.0
5.0 3.0 4.0 4.0
8.0 2.0 4.0 0.0
6.0 0.0 2.0 2.0
```

You can also test your implementation of the 2D Ordered Inverse HWT on the images from the previous assignment.

What To Submit

Submit your `WaveletAlgos_S17_HW07.java` and `TwoDHWT.java` with your implementation of `ordInvDWTForNumIters()` or their Py equivalents via Canvas.

Happy Hacking!