Subband Coding

- Most source outputs exhibit a combination of various characteristics, which makes it difficult to select a compression scheme exactly suited to the source output
- Subband coding
  - To decompose a signal into constituent parts
    - adopt the coding technique best suited to each constituent to improve compression performance
    - Furthermore, each constituent may have different perceptual characteristics
- Example :
  - $x(n)$: 10 14 10 12 14 8 14 12 10 8 10 12
  - To decompose $x(n)$ into two subsequence $y_0(n)$ and $y_1(n)$

    $y_0(n) = [x(2n) + x(2n+1)]/2$: 12 11 11 13 9 11

    $y_1(n) = [x(2n) - x(2n+1)]/2$: -2 -1 3 1 1 -1
  - To recover $x(n)$ from $y_0(n)$ and $y_1(n)$

    $x(2n) = y_0(n) + y_1(n)$

    $x(2n+1) = y_0(n) - y_1(n)$

Linear Time Invariant System

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \mathrm{n} \neq 0 \end{cases} \qquad \delta(n-k) = \begin{cases} 1 & n = k \\ 0 & \mathrm{n} \neq \mathrm{k} \end{cases} \qquad L : \delta(n) \rightarrow h(n)$$

where $\delta(n)$: impulse sequence and $h(n)$: impulse response sequence

$$\Rightarrow f(n) = \sum_{k=-\infty}^{\infty} f(k)\delta(n-k)$$

$\mathcal{L}(ax+y)$

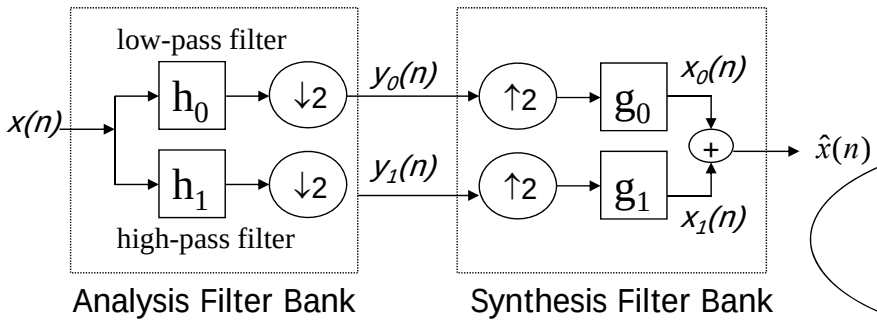
$= a\mathcal{L}(x) + \mathcal{L}(y)$

$$\Rightarrow g(n) = L\{f(n)\} = L\{\sum_{k=-\infty}^{\infty} f(k)\delta(n-k)\}$$

$$\overset{\text{linear}}{=} \sum_{k=-\infty}^{\infty} f(k)L\{\delta(n-k)\} \overset{\text{time invariant}}{=} \sum_{k=-\infty}^{\infty} f(k)h(n-k) \equiv f(n) * h(n)$$

linear convolution

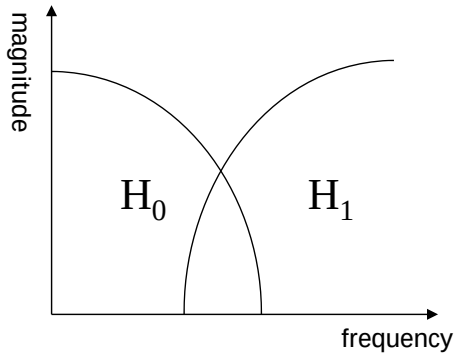
Two-Channel Filter Bank (1)



$$y(n) = x(n) * h(n) = \sum_{l=-\infty}^{\infty} x(l)h(n-l)$$

$$\begin{cases} y_0(k) = x(n) * h_0(n)\,|_{n=2k} = \sum_{l=-\infty}^{\infty} x(l)h_0(2k-l) \\ y_1(k) = x(n) * h_1(n)\,|_{n=2k} = \sum_{l=-\infty}^{\infty} x(l)h_1(2k-l) \end{cases}$$

$$\hat{x}(n) = x_0(n) + x_1(n)$$
$$= \sum_{k=-\infty}^{\infty} y_0(k)g_0(n-2k) + \sum_{k=-\infty}^{\infty} y_1(k)g_1(n-2k)$$



Two-Channel Filter Bank (2)

Example :

$$h_0(n) = \begin{cases} 1/2 & n = -1,0 \\ 0 & \text{otherwise} \end{cases} \qquad g_0(n) = \begin{cases} 1 & n = 0,1 \\ 0 & \text{otherwise} \end{cases}$$

$$h_1(n) = \begin{cases} -1/2 & n = -1 \\ 1/2 & n = 0 \\ 0 & \text{otherwise} \end{cases} \qquad g_1(n) = \begin{cases} 1 & n = 0 \\ -1 & n = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{cases} y_0(n) = \sum_{l=-\infty}^{\infty} x(l)h_0(2n-l) = x(2n+1)h_0(-1) + x(2n)h_0(0) = \dfrac{x(2n)+x(2n+1)}{2} \\ y_1(n) = \sum_{l=-\infty}^{\infty} x(l)h_1(2n-l) = x(2n+1)h_1(-1) + x(2n)h_1(0) = \dfrac{x(2n)-x(2n+1)}{2} \end{cases}$$

$$\begin{cases} x_0(2n) = \sum_{k=-\infty}^{\infty} y_0(k)g_0(2n-2k) = y_0(n)g_0(0) = y_0(n) \\ x_0(2n+1) = \sum_{k=-\infty}^{\infty} y_0(k)g_0(2n+1-2k) = y_0(n)g_0(1) = y_0(n) \end{cases}$$

$$\begin{cases} x_1(2n) = \sum_{k=-\infty}^{\infty} y_1(k)g_1(2n-2k) = y_1(n)g_1(0) = y_1(n) \\ x_1(2n+1) = \sum_{k=-\infty}^{\infty} y_1(k)g_1(2n+1-2k) = y_1(n)g_1(1) = -y_1(n) \end{cases}$$

$$\begin{cases} \hat{x}(2n) = x_0(2n) + x_1(2n) \\ \quad = y_0(n) + y_1(n) = x(2n) \\ \hat{x}(2n+1) = x_0(2n+1) + x_1(2n+1) \\ \quad = y_0(n) - y_1(n) = x(2n+1) \end{cases}$$

perfect reconstruction

TABLE 13.1 Coefficients for the 8-tap Johnston low-pass filter.

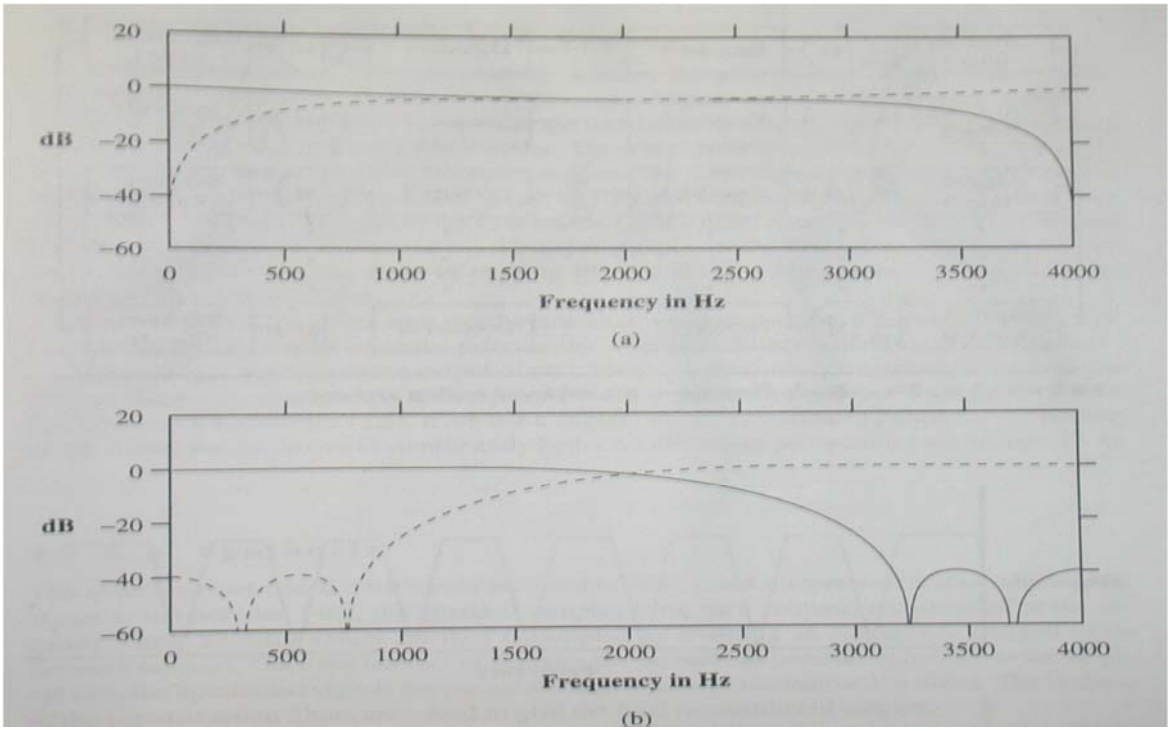
h_0, h_7	0.00938715
h_1, h_6	0.06942827
h_2, h_5	−0.07065183
h_3, h_4	0.48998080

$h_0(n)$, n= 0,…,7

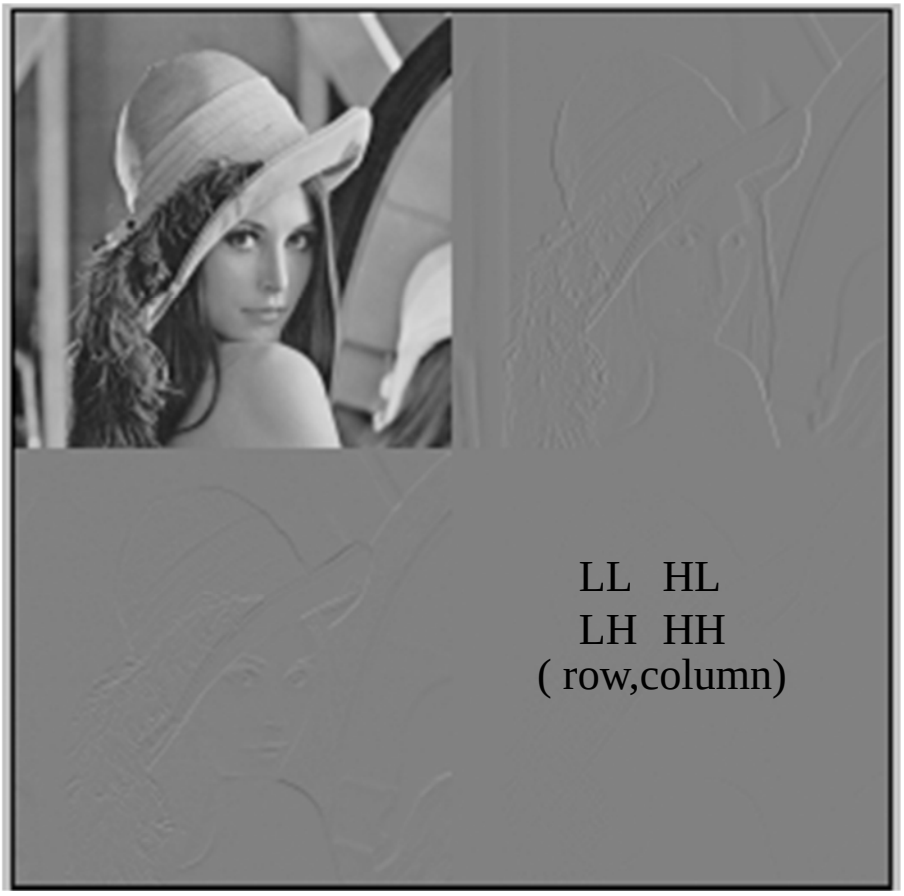
TABLE 13.4 Coefficients for the eight-tap Smith-Barnwell low-pass filter.

h_0	0.0348975582178515
h_1	−0.01098301946252854
h_2	−0.06286453934951963
h_3	0.223907720892568
h_4	0.556856993531445
h_5	0.357976304997285
h_6	−0.02390027056113145
h_7	−0.07594096379188282

$h_0(n)$, n= 0,…,7



Magnitude transfer functions of the (a) eight-tap Johnston and (b) eight-tap Smith-Barnwell filters


LL HL
LH HH
(row,column)

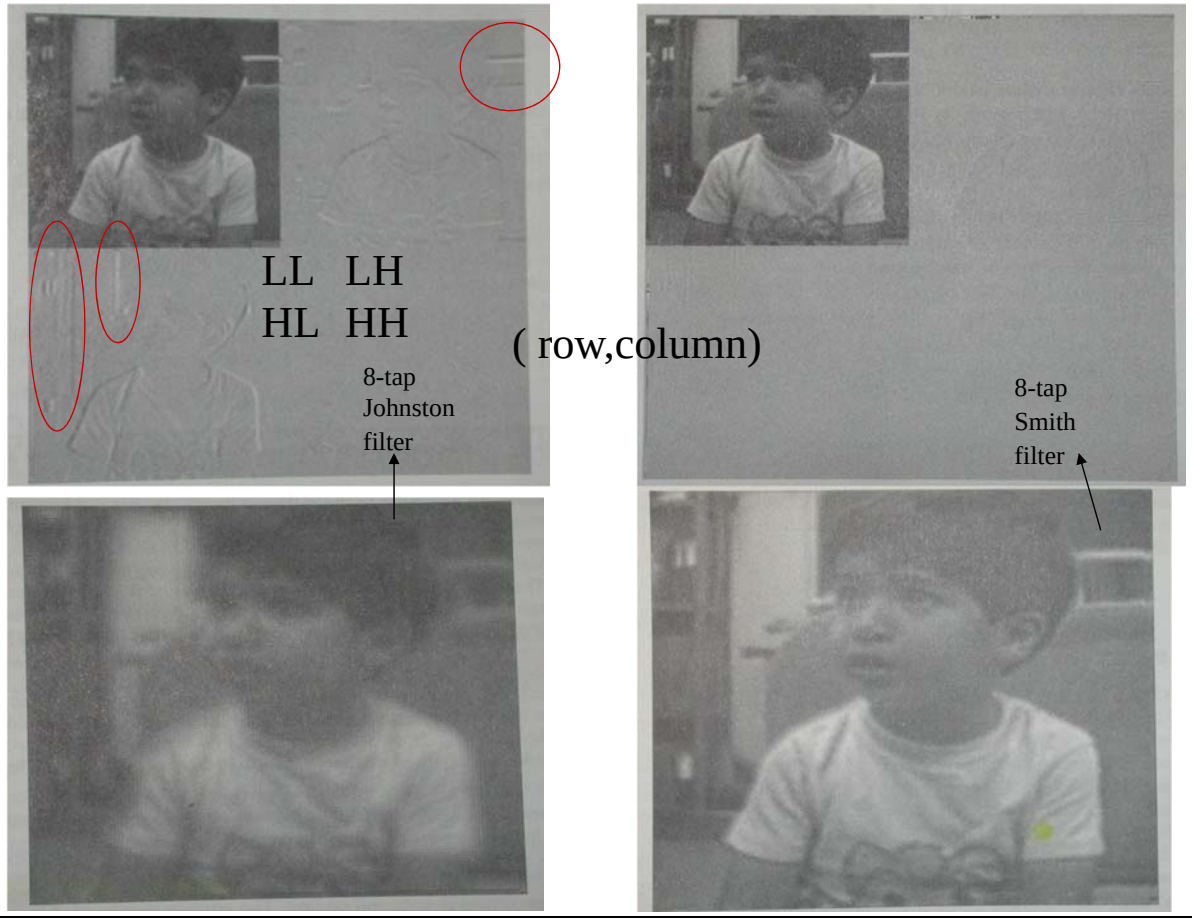


TABLE 14.1 Coefficients for the 4-tap Daubechies low-pass filter.

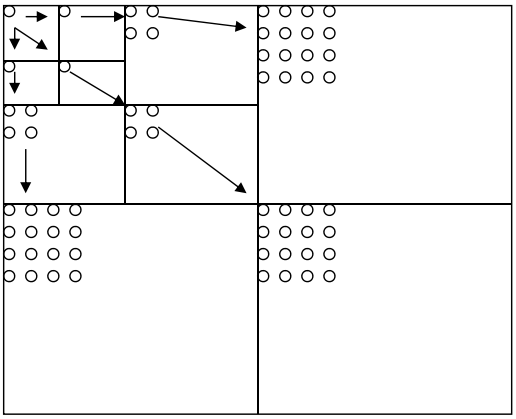
h(n), n= 0,…,3

h_0	0.4829629131445341
h_1	0.8365163037378079
h_2	0.2241438680420134
h_3	−0.1294095225512604

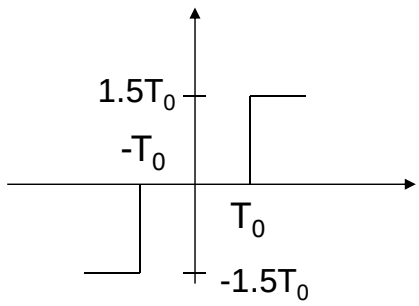
Embedded Zerotree Wavelet (EZW) (1)

- Embedded coding
  - the bits are ordered in importance
  - the codec can terminate at any point thereby allowing a target rate or distortion metric to be met exactly
- Zerotree structure
  - if a coefficient has a magnitude less than a given threshold, all its descendants might have magnitudes less than that threshold
  - significant coefficient with respect to threshold T : if $|x| \geq T$.
- Symbols
  - zerotree root (zr) : all descendants (including itself) are insignificant
  - isolated zero (iz) : the coefficient is insignificant but has some significant descendant
  - significant positive (sp)
  - significant negative (sn)

Embedded Zerotree Wavelet (EZW) (2)

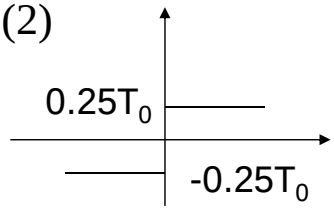


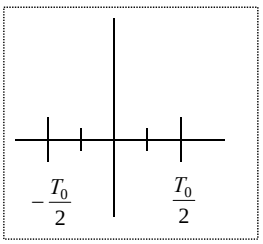
(1)



three-level midtread quantizer for sp (sn)

(2)





two-level midrise quantizer for refinemet

$$T_0 = 2^{\lfloor \log_2 C_{\max} \rfloor}$$

- C_{max}: the value of the largest coefficient
- $C_{max} \in [T_0, 2T_0)$

Embedded Zerotree Wavelet (EZW) (3)

Example :

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

Initial threshold $T_0 = 2^{\lfloor \log_2 26 \rfloor} = 16$

Pass 1: sp zr zr zr ← send 8 bits

∵ $1.5\ T_0 = 24$, $26-24 \geq 0$, ← send 1 bit

and $T_0/4=4$

∴ 收端得到 24+4=28 Q

Pass 2: iz zr zr sp sp iz iz ← send 14 bits

$T_1 = 0.5\ T_0 = 8$

∵ $1.5\ T_1 = 12$, $T_1/4=2$ send 3 bits

and 26-28<0, 13-12 ≥0, 10-12<0

∴ 收端得到 28-2=26, 12+2=14, 12-2=10

Ls={26}

*	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

28		

Ls={26,13,10}

*	6	*	*
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

26		14 10

Embedded Zerotree Wavelet (EZW) (4)

Pass 3: sp sn sp sp sp sp sn iz iz sp iz iz iz ← send 26 bits

- $T_2 = 0.5\ T_1 = 4$
- $1.5\ T_2 = 6$ and $T_2/4=1$

*	6	*	*
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

*	*	*	*
*	*	*	*
*	*	*	-3
2	-2	-2	0

Ls={26, 13, 10, 6, -7, 7, 6, 4, 4, -4, 4}

Lr={26, 14, 10, 6, -6, 6, 6, 6, 6, -6, 6} ← Emulate receiver

Ls-Lr={ 0, -1, 0, 0, -1, 1, 0, -2, -2, 2, -2}

send 11 bits {1,0,1,1,0,1,1,0,0,1,0}

Lr={26, 14, 10, 6, -6, 6, 6, 6, 6, -6, 6}

Ld={ 1, -1, 1, 1, -1, 1, 1, -1, -1, 1, -1}

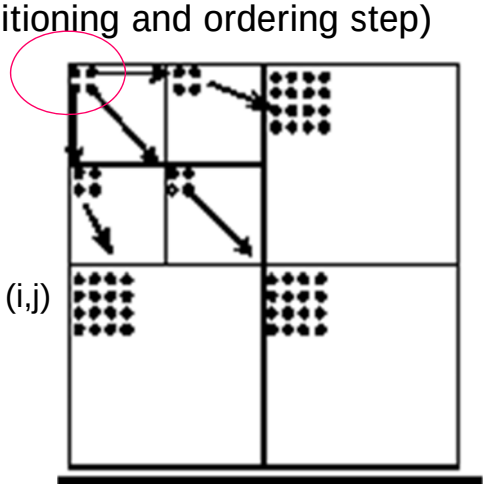
Lr+Ld={27, 13, 11, 7, -7, 7, 7, 5, 5, -5, 5}

26		14 10

27	7	13 11
-7	7	7 5
5	-5	5

SPIHT (1)

- Set Partitioning in Hierarchical Trees
  - Use a partitioning of the *spatial orientation trees* in a manner than tends to keep insignificant coefficients together in a larger subsets
  - Partitioning decision -> binary decision
    Provide a significance map encoding that is more efficient than EZW
  - Each pass consists of 2 steps
    - Significance map encoding (set partitioning and ordering step)
    - Refinement
- Partitioning types
  - O(i,j)
    Set of coordinates of the offsprings at (i,j)
  - D((i,j)
    Set of all descendants of the coefficients at (i,j)
  - H
    Set of all root nodes
  - L (i,j)
    L (i,j) = D (i,j) - O (i,j)



SPIHT (2)

- Significant
  - A set D(i,j) or L(i,j) is significant is any coefficient in the set has a magnitude greater than the threshold
- Three lists are maintained
  - LIP (list of insignificant pixels)
    Is Initialized with the set H
  - LIS (list of insignificant sets)
    Contain the coordinates of the roots of types D or L
  - LSP (list of significant pixels)
    Is initially empty
- At each pass
  - Significant map encoding step
    - Process the members of LIP
      - If the coefficient is significant => transmit (1 + sign), move to LSP
      - Otherwise => transmit 0
    - Then process the members of LIS
      - Insignificant set => 0
  - Refinement step
    - Process the elements of LSP

SPIHT (3)

- Example
  - Initialize

    $$n = \lfloor \log_2 c_{\max} \rfloor = \lfloor \log_2 26 \rfloor = 4$$

    - T = 16
  - 1st pass
    - LIP { (0,0)-> 26, (0,1)->6, (1,0)-> -7, (1,1)-> 7}
    - LIS { (0,1)D, (1,0)D, (1,1)D }
    - LSP { }

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

    - LIP
      - 26 => 1 + 0 => move to LSP
      - 6 = > 0
      - -7 = > 0
      - 7 = > 0
    - LIS
      - (0,1)D = > 0
      - (1,0)D => 0
      - (1,1)D => 0

SPIHT(4)

- Example (cont.)
  - 2nd pass
    - n-- => n = 3, T = 8
    - LIP { (0,1)->6, (1,0)-> -7, (1,1)-> 7}
    - LIS { (0,1)D, (1,0)D, (1,1)D }
    - LSP { (0,0)-> 26 }

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

    - LIP
      - 6 => 0
      - -7 => 0
      - 7 => 0
    - LIS
      - (0,1)D => 1
        - » 13 => 1+ 0 => move to LSP
        - » 10 => 1 + 0 => move to LSP
        - » 6 => 0 => move to LIP
        - » 4 => 0 => move to LIP
      - (1,0)D => 0
      - (1,1)D => 0
    - LSP
      - 26=$(11010)_2$ => 1

SPIHT (5)

- Example (cont.)
  - 3rd pass
    - n-- => n = 2, T = 4
    - LIP { (0,1)->6, (1,0)-> -7, (1,1)-> 7, (1,2)->6, (1,3)->4 }
    - LIS { (1,0)D, (1,1)D }
    - LSP { (0,0)-> 26, (0,2)->13, (0,3)->10 }

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

    - LIP
      - 6 => 1+0 => move to LSP
      - -7 => 1+1 => move to LSP
      - 7 => 1+0 => move to LSP
      - 6 => 1+0 => move to LSP
      - 4 => 1+0 => => move to LSP
    - LIS
      - (1,0)D => 1
        - » 4 => 1+ 0 => move to LSP
        - » -4 => 1 + 1 => move to LSP
        - » 2 => 0 => move to LIP
        - » -2 => 0 => move to LIP
      - (1,1)D => 1
        - » 4 => 1+ 0 => move to LSP
        - » -3 => 0 => move to LIP
        - » -2 => 0 => move to LIP
        - » 0 => 0 => move to LIP
    - LSP
      - 26=> 0, 13 => 1 , 10=> 0

26=(11010) 26<28

13=(01101) 13>12

10=(01010) 10<12

LIP { (3,0)->2, (3,1)-> -2, (2,3)->-3, (3,2)->-2, (3,3)->0 }

LIS { }

LSP { (0,0)-> 26, (0,2)->13, (0,3)->10, (0,1)->6, (1,0)->-7, (1,1)->7, (1,2)->6, (1,3)->4, (2,0)->4, (2,1)-> -4, (2,2)->4 }

Algorithm II:

1) **Initialization:** output $n = \lfloor \log_2 (\max_{(i,j)}\{|c_{i,j}|\}) \rfloor$; set the LSP as an empty list, and add the coordinates $(i, j) \in \mathcal{H}$ to the LIP, and only those with descendants also to the LIS, as type A entries.
2) **Sorting Pass:**
   2.1) for each entry (i, j) in the LIP do:
      2.1.1) output $S_n(i, j)$;
      2.1.2) if $S_n(i, j) = 1$ then move (i, j) to the LSP and output the sign of $c_{i,j}$;

Type A: *D(i,j)* Type B : *L(i,j)*

2.2) for each entry (i, j) in the LIS do:

2.2.1) if the entry is of type A then

- output $S_n(\mathcal{D}(i, j))$;
- if $S_n(\mathcal{D}(i, j)) = 1$ then
  - for each $(k, l) \in \mathcal{O}(i, j)$ do:
    - output $S_n(k, l)$;
    - if $S_n(k, l) = 1$ then add (k, l) to the LSP and output the sign of $c_{k,l}$;
    - if $S_n(k, l) = 0$ then add (k, l) to the end of the LIP;
  - if $\mathcal{L}(i, j) \neq \emptyset$ then move (i, j) to the end of the LIS, as an entry of type B, and go to Step **2.2.2);** otherwise, remove entry (i, j) from the LIS;

2.2.2) if the entry is of type B then

- output $S_n(\mathcal{L}(i, j))$;
- if $S_n(\mathcal{L}(i, j)) = 1$ then
  - add each $(k, l) \in \mathcal{O}(i, j)$ to the end of the LIS as an entry of type A;
  - remove (i, j) from the LIS.

3) **Refinement Pass:** for each entry (i, j) in the LSP, except those included in the last sorting pass (i.e., with same n), output the nth ~~most significant~~ bit of $|c_{i,j}|$;

4) **Quantization-Step Update:** decrement n by 1 and go to Step **2.**

(11010)

0th bit

3th bit

63(0,0)	-34(0,1)	49(0,2)	10(0,3)	7(0,4)	13(0,5)	-12(0,6)	7(0,7)
-31(1,0)	23(1,1)	14(1,2)	-13(1,3)	3(1,4)	4(1,5)	6(1,6)	-1(1,7)
15(2,0)	14(2,1)	3(2,2)	-12(2,3)	5(2,4)	-7(2,5)	3(2,6)	9(2,7)
-9(3,0)	-7(3,1)	-14(3,2)	-8(3,3)	4(3,4)	-2(3,5)	3(3,6)	2(3,7)
-5(4,0)	9(4,1)	-1(4,2)	47(4,3)	4(4,4)	6(4,5)	-2(4,6)	2(4,7)
3(5,0)	0(5,1)	-3(5,2)	2(5,3)	3(5,4)	-2(5,5)	0(5,6)	4(5,7)
2(6,0)	-3(6,1)	6(6,2)	-4(6,3)	3(6,4)	6(6,5)	3(6,6)	6(6,7)
5(7,0)	11(7,1)	5(7,2)	6(7,3)	0(7,4)	3(7,5)	-4(7,6)	4(7,7)

63(0,0)	-34(0,1)	49(0,2)	10(0,3)	7(0,4)	13(0,5)	-12(0,6)	7(0,7)
-31(1,0)	23(1,1)	14(1,2)	-13(1,3)	3(1,4)	4(1,5)	6(1,6)	-1(1,7)
15(2,0)	14(2,1)	3(2,2)	-12(2,3)	5(2,4)	-7(2,5)	3(2,6)	9(2,7)
-9(3,0)	-7(3,1)	-14(3,2)	-8(3,3)	4(3,4)	-2(3,5)	3(3,6)	2(3,7)
-5(4,0)	9(4,1)	-1(4,2)	47(4,3)	4(4,4)	6(4,5)	-2(4,6)	2(4,7)
3(5,0)	0(5,1)	-3(5,2)	2(5,3)	3(5,4)	-2(5,5)	0(5,6)	4(5,7)
2(6,0)	-3(6,1)	6(6,2)	-4(6,3)	3(6,4)	6(6,5)	3(6,6)	6(6,7)
5(7,0)	11(7,1)	5(7,2)	6(7,3)	0(7,4)	3(7,5)	-4(7,6)	4(7,7)