HW #2 Quad Trees四元樹

Spatial Structures

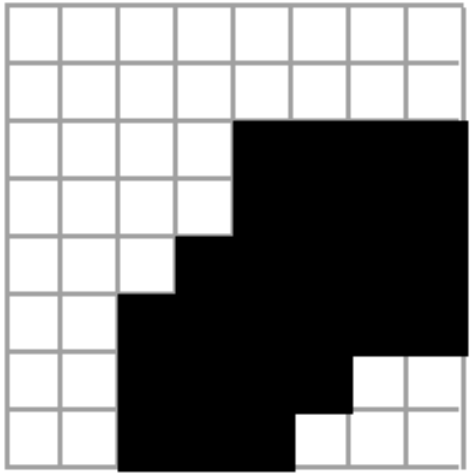
- **Computer graphics, image processing, and GIS (geographic information systems) all make use of a data structure called a quadtree.**
- **Quadtrees represent regional or block data efficiently and support efficient algorithms for operations like the *union* and *intersection* of images.**
- **A quadtree for a black and white image is constructed by successively dividing the image into four equal quadrants. If all the pixels in a quadrant are the same color (all black or all white) the division process for that quadtree stops.**

HW #2 (2)

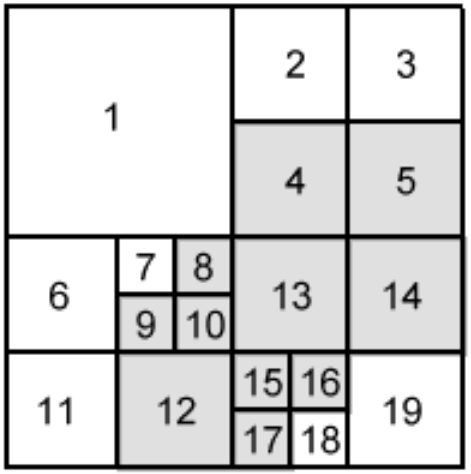
- **Quadrants that contain both black and white pixels are subdivided into four equal quadrants and this process continues until each subquadrant consists of either all black or all white pixels. It is entirely possible that some subquadrants consist of a *single* pixel.**

- **For example, using 0 for white and 1 for black, the region on the left (see next slide) is represented by the matrix of zeros and ones in the middle. The matrix is divided into subquadrants as shown on the right where gray squares represent subquadrants that consist entirely of black pixels.**

HW #2 (3)



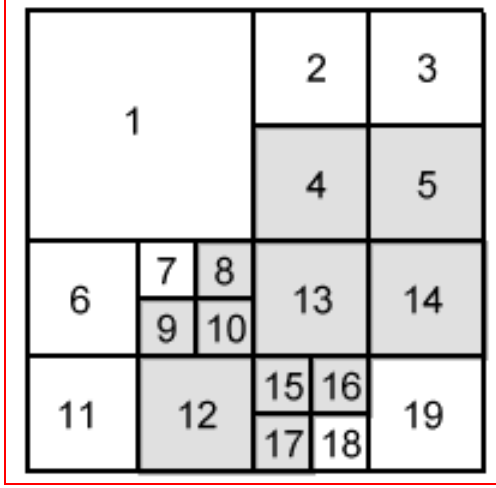
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 0 0
0 0 1 1 1 0 0 0

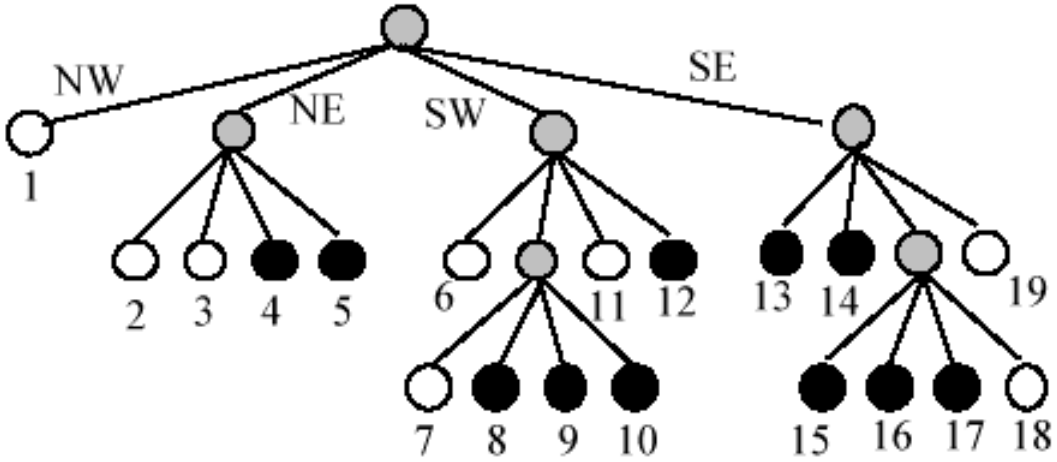


HW #2 (4)

- **A quadtree is constructed from the block structure of an image. The root of the tree represents the entire array of pixels.**
- **Each non-leaf node of a quadtree has four children, corresponding to the four subquadrants of the region represented by the node.**
- **Leaf nodes represent regions that consist of pixels of the same color and thus are not subdivided.**
- **For example, the image shown before, with the block structure on the right, is represented by the quadtree in the next slide.**

HW #2 (5)


1
2
3
4
5
6
7
8
9
10
13
14
11
12
15
16
17
18
19

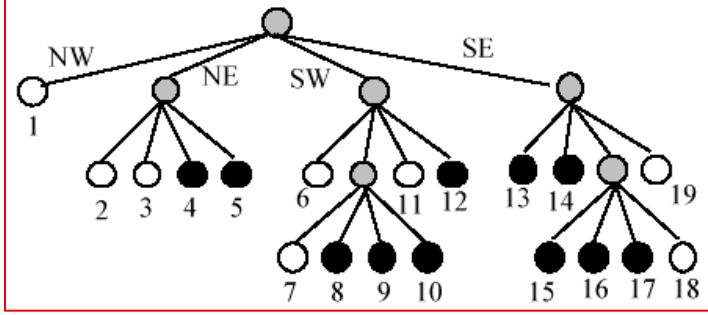

NW
NE
SW
SE
1
2
3
4
5
6
11
12
13
14
19
7
8
9
10
15
16
17
18

HW #2 (6)

- **Leaf nodes are white if they correspond to a block of all white pixels, and black if they correspond to a block of all black pixels. In the tree, each leaf node is numbered corresponding to the block it represents in the diagram shown before.**

- **The branches of a non-leaf node are ordered from left-to-right as shown for the *northwest*, *northeast*, *southwest*, and *southeast* quanrants (or upper-left, upper-right, lower-left, lower-right).**

HW #2 (7)



- **A tree can be represented by a sequence of numbers representing the root-to-leaf paths of black nodes.**
- **Each path is a base five number constructed by labeling branches with 1, 2, 3, or 4 with *NW* = 1, *NE* = 2, *SW* = 3, *SE* = 4, and with the least significant digit of the base, five number corresponding to the branch from the root.**
- **For example, the node labeled 4 has path NE, SW, which is 32_5 (base 5) or 17_{10} (base 10); the node labeled 12 has path SW, SE or $43_5 = 23_{10}$; and the node labeled 15 has path SE, SW, NW, or $134_5 = 44_{10}$.**

HW #2 (8)

- **The entire tree is represented by the sequence of number (in base 10)**

  9 14 17 22 23 44 63 69 88 94 113

- **All images are *black and white*. All images are *square* and the size is a power of two, e.g., 4x4, 8x8, 16x16.**

- **Write a function (in C++,or Java if you prefer) that converts images into root-to-leaf paths and a function that converts root-to-leaf paths into images.**

HW #2 (9)

Input

- **The input contains one or more images. Each image is square, and the data for an image starts with an integer *n*, where |*n*| is the length of a side of the square (always a power of two, with |*n*| < 64) followed by a representation of the image.**

- **A representation is either a sequence of n^2 zeros and ones comprised of |*n*| lines of |*n*| digits per line, or the sequence of numbers that represent the root-to-leaf paths of each *black* node in the quadtree that represents the image.**

HW #2 (10)

- If *n* is positive, the zero/one representation follows; if *n* is negative, the sequence of black node path numbers (in base 10) follows.
- The sequence is terminated by the number –1.

- A one-node tree that represents an all-black image is represented by the number 0. A one-node tree that represents an all-white image is represented by an empty sequence (no numbers).
- The end of data is signaled by a value of 0 for *n*.

HW #2 (11)

Output

- **For each image in the input, first output the number of the image, as shown in the sample output. Then output the alternate form of the image.**
- **If the image is represented by zeros and ones, the output consists of root-to-leaf paths of all black nodes in the quadtree that represents the image. The value should be based 10 representation of the base 5 path numbers, and the value should be printed in sorted order. If there are more than 12 black nodes, print a newline after every 12 nodes. The total number of black nodes should be printed after the path numbers.**

HW #2 (12)

- **If the image is represented by the root-to-leaf paths of black nodes, the output consists of an ASCII representation of the image with the character "." used for white/zero and the character "*" used for black/one. There should be *n* characters per line for an *n* x *n* image.**

HW #2 (13)

Sample Input

8

00000000

00000000

00001111

00001111

00011111

00111111

00111100

00111000

HW #2 (14)

-8

9 14 17 22 23 44 63 69 88 94 113 –1

2

00

00

-4

0 –1

0

HW #2 (15)

Output for the Sample Input

Image 1

9 14 17 22 23 44 63 69 88 94 113

Total number of black nodes = 11

HW #2 (16)

Image 2

```
........
........
...****
...****
..*****
.******
.****..
.***...
```

HW #2 (17)

Image 3

Total number of black nodes = 0

Image 4

****

****

****

****

Two last test images

8

00110000

01111000

11001100

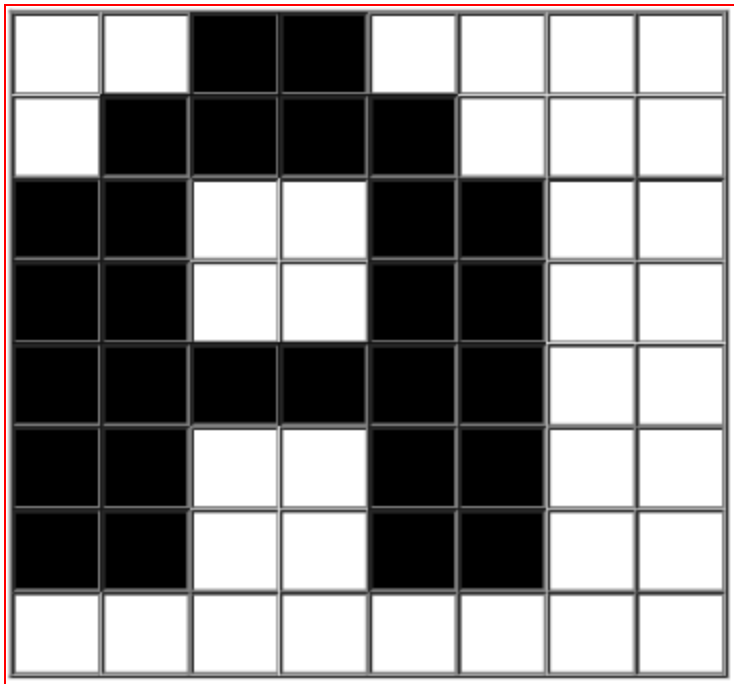
11001100

11111100

11001100

11001100

00000000



16
0000011111110000
0001111111111100
0011111111111110
0011111111111110
0111111111111111
0111111111111111
0111111111111111
1111111111111111
0111111111111111
0111111111111111
0111111111111111
0011111111111110
0011111111111110
0001111111111100
0000011111110000
0000000010000000