2nd LGE Code Jam 2019 - Problem A (English)

Created by 박건형 sunnyun.park, last modified by 강우람 woolam.kang on 2019/07/19

<Array Rotation>

Problem description

The purpose of problem is moving the elements of the given array according to the rotation rule.

It is not required to use special algorithms. It can be simply solved by considering how to access each position in a two-dimensional array and how to repeat similar operations.

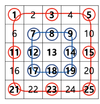
Solutions

The elements that need to be moved can be grouped into several groups. Each group shapes a square.

Since the center the element has not changed, when the array size is n * n, the elements belonging to (n / 2) squares are moved.

The elements belonging to each square are displaced from each other and have no interaction with elements of other squares.

In an example of the problem, the positions of the elements corresponding to two squares ((5/2) = 2) are changed as follows.



Because the rotation for each square is same, let's use it in common and repeat it by the number of squares.

To define the location of the elements belonging to a square, let's number each square first (called "level"). If you are numbering from the outside square, the outermost square will be 0 and the innermost square will be [n / 2] -1.

When each the element is mapped to a[x-coordinate][y-coordinate] (each coordinate starts from 0), each right the element need to be moved is mapped as follows.

...	...	...	...	...
...	a[level][level]	a[n / 2][level]	a[n - level - 1][level]	...
...	a[level][n / 2]	...	a[n - level - 1][n / 2]	...
...	a[level][n - level - 1]	a[n / 2][n - level - 1]	a[n - level - 1][n - level - 1]	...
...	...	...	...	...

Solution 1)

The simple solution is to make two methods that rotate clockwise and counterclockwise by 45 degrees and repeat it as needed.

The pseudocode is as follows.

```
for i=0 to [n/2]-1
    if d>0 then
        for j=i to d/45
            rotate i-th level the elements clockwise by 45 degrees
        end for
    else
        for j=i to -d/45
            rotate i-th level the elements counterclockwise by 45 degrees
        end for
end for
```

The implementation is as follows.

C Java

A-solution.c

```c
#include <stdio.h>

int n;
int d;
int a[500][500];

void input() {
    scanf("%d %d", &n, &d);

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%d", &a[j][i]);
        }
    }
}

void rotate(int level) {
    int t = a[level][level];
    a[level][level] = a[level][n / 2];
    a[level][n / 2] = a[level][n - level - 1];
    a[level][n - level - 1] = a[n / 2][n - level - 1];
    a[n / 2][n - level - 1] = a[n - level - 1][n - level - 1];
    a[n - level - 1][n - level - 1] = a[n - level - 1][n / 2];
    a[n - level - 1][n / 2] = a[n - level - 1][level];
    a[n - level - 1][level] = a[n / 2][level];
    a[n / 2][level] = t;
}

void rotate_inv(int level) {
    int t = a[n / 2][level];
    a[n / 2][level] = a[n - level - 1][level];
    a[n - level - 1][level] = a[n - level - 1][n / 2];
    a[n - level - 1][n / 2] = a[n - level - 1][n - level - 1];
    a[n - level - 1][n - level - 1] = a[n / 2][n - level - 1];
    a[n / 2][n - level - 1] = a[level][n - level - 1];
    a[level][n / 2] = a[level][level];
    a[level][level] = t;
}

void process() {
    int move = d / 45;

    for (int i = 0; i < n / 2; ++i) {
        if (move > 0) {
            for (int j = 0; j < move; ++j) {
                rotate(i);
            }
        } else {
            for (int j = 0; j < -move; ++j) {
                rotate_inv(i);
            }
        }
    }
}

void output() {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
}

int main() {
    int t;
    scanf("%d", &t);

    for (int i = 0; i < t; ++i) {
        input();
        process();
        output();
    }

    return 0;
}
```

Solution 2)

In Solution 1, the rotation of 45 degrees or -45 degrees is repeated up to 7 times. Let's avoid repetition and rotate by d degrees at once.

Let's number the elements from 0 to 7 clockwise from the top left.

...	...	...	...	...
...	a[level][level] → 0	a[n / 2][level] → 1	a[n - level - 1][level] → 2	...
...	a[level][n / 2] → 7	...	a[n - level - 1][n / 2] → 3	...
...	a[level][n - level - 1] → 6	a[n / 2][n - level - 1] → 5	a[n - level - 1][n - level - 1] → 4	...
...	...	...	...	...

The j-th element moves to (j + (360 + d) / 45)% 8 -th position.

If we want to rotate by 135 degrees, the 1st the element moves to 4th place because (1 + (360 + 135) / 45) % 8 = 4.

If we want to rotate by -135 degrees, the 1st the element moves to 6th place because (1 + (360 - 135) / 45) % 8 = 6.

In summary, the pseudocode is as follows.

```
for i=0 to [n/2]-1
    for j=0 to 7
        move j-th element of i-th level to ((j + (360 + d) / 45) % 8)-th place of i-th level
        (backup is required)
end for
```

The implementation is as follows.

C Java

A-solution.c

```c
#include <stdio.h>

int n;
int d;
int a[500][500];

void input() {
    scanf("%d %d", &n, &d);

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%d", &a[j][i]);
        }
    }
}

int get(int level, int index) {
    switch (index) {
        case 0:
            return a[level][level];
        case 1:
            return a[n / 2][level];
        case 2:
            return a[n - level - 1][level];
        case 3:
            return a[n - level - 1][n / 2];
        case 4:
            return a[n - level - 1][n - level - 1];
        case 5:
            return a[n / 2][n - level - 1];
        case 6:
            return a[level][n - level - 1];
        case 7:
            return a[level][n / 2];
        default:
            return -1;
    }
}

void set(int level, int index, int value) {
    switch (index) {
        case 0:
            a[level][level] = value;
            break;
        case 1:
            a[n / 2][level] = value;
            break;
        case 2:
            a[n - level - 1][level] = value;
            break;
        case 3:
            a[n - level - 1][n / 2] = value;
            break;
        case 4:
            a[n - level - 1][n - level - 1] = value;
            break;
        case 5:
            a[n / 2][n - level - 1] = value;
            break;
        case 6:
            a[level][n - level - 1] = value;
            break;
        case 7:
            a[level][n / 2] = value;
            break;
        default:
            break;
    }
}

void process() {
    int move = d / 45;

    for (int i = 0; i < n / 2; ++i) {
        int value[8];
        for (int j = 0; j < 8; ++j) {
            value[j] = get(i, j);
        }
        for (int j = 0; j < 8; ++j) {
            set(i, (8 + j + move) % 8, value[j]);
        }
    }
}

void output() {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
}

int main() {
    int t;
    scanf("%d", &t);

    for (int i = 0; i < t; ++i) {
        input();
        process();
        output();
    }

    return 0;
}
```