Задача BE (Луна)

Олексій Лубинець, Валерія Жовковська 5 курс, ФВЕ

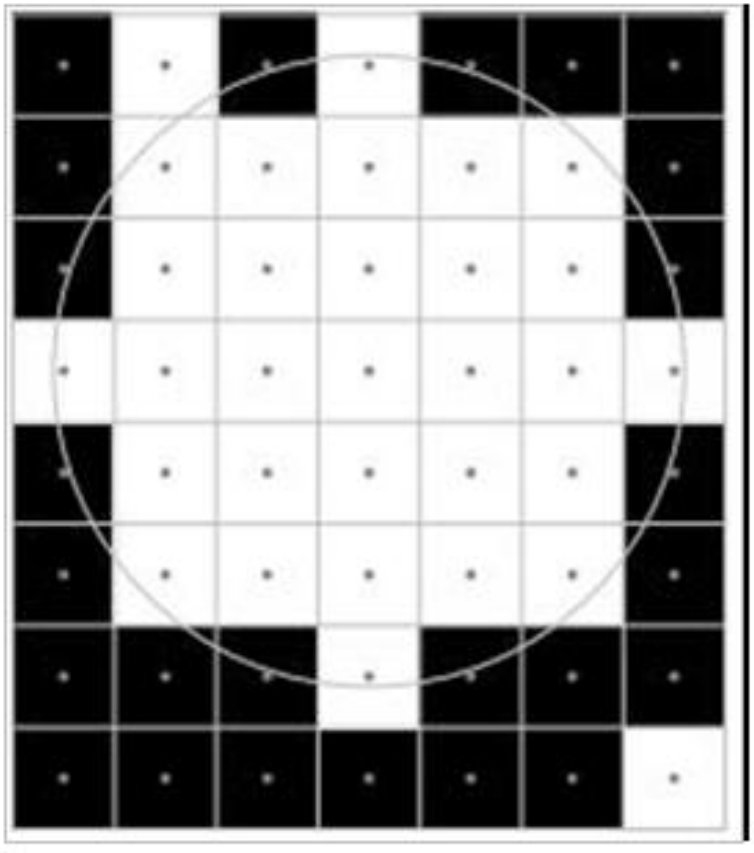
Об'єктно-орієнтоване програмування

25 травня 2018 р.

Умова задачі

На фотографиях ночного неба можно различить только два цвета: черный и белый.
Будем считать, что Луна на снимке выглядит как круг с центром в точке изображения C и целым неотрицательным радиусом r, то есть как множество белых точек, расстояние от центров которых до точки C не больше r. Луна полностью поместилась на снимке.
Также некоторые звезды могут присутствовать на снимке в виде отдельных белых точек. Таких точек, как и звезд на небе, может быть достаточно много.
Напишите программу, которая по изображению найдет наибольший возможный радиус круга, который соответствует Луне.

Приклад



Input-Output

Input format

В первой строке ввода записаны целые числа w и h — горизонтальное и вертикальное разрешение снимка, соответственно (1 <= w, h <= 200). В следующих h строках записано по w символов "."(черная точка) или "*"(белая точка).

Output format

В одной строке выведите натуральное число — максимальный радиус изображения Луны. Гарантируется, что корректный ответ существует.

"Критерій Місяця Метод No.1"

Спосіб гарантувати, що ми знайшли Місяць радіуса R з центром C - перевірити всі пікселі, відстань до яких не перевищує R, чи є вони білими.
Тож пробігаємо всі пікселі, крім "рамочки" на предмет того, чи можуть вони бути центром Місяця і якщо можуть, то якого радіуса. Максимальний із знайдених радіусів і буде відповіддю на задачу.

Визначення радіуса для Місяця з центром у конкретному пікселі

Заздалегідь для кожного “потенційного радіуса“ визначаються точки (а саме зсуви по X та Y), які будуть для Місяця з таким радіусом граничними.

0	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11
2	3	3	4	5	6	7	8	9	10	11
3	4	4	5	5	6	7	8	9	10	11
4	5	5	5	6	7	8	9	9	10	11
5	6	6	6	7	8	8	9	10	11	12
6	7	7	7	8	8	9	10	10	11	12
7	8	8	8	9	9	10	10	11	12	13
8	9	9	9	9	10	10	11	12	13	13
9	10	10	10	10	11	11	12	13	13	14
10	11	11	11	11	12	12	13	13	14	15

Визначення радіуса для Місяця з центром у конкретному пікселі

```
for(int i=0; i<100; i++)
    for(int j=0; j<100; j++)
    {
        r = (int) (sqrt(i*i + j*j)+1);
        if(r - sqrt(i*i + j*j) == 1)
            r--;
        if(r > 99)
            break;
        DX[r][num[r]] = i;
        DY[r][num[r]] = j;
        num[r]++;
    }
```

Пошук по всім пікселям

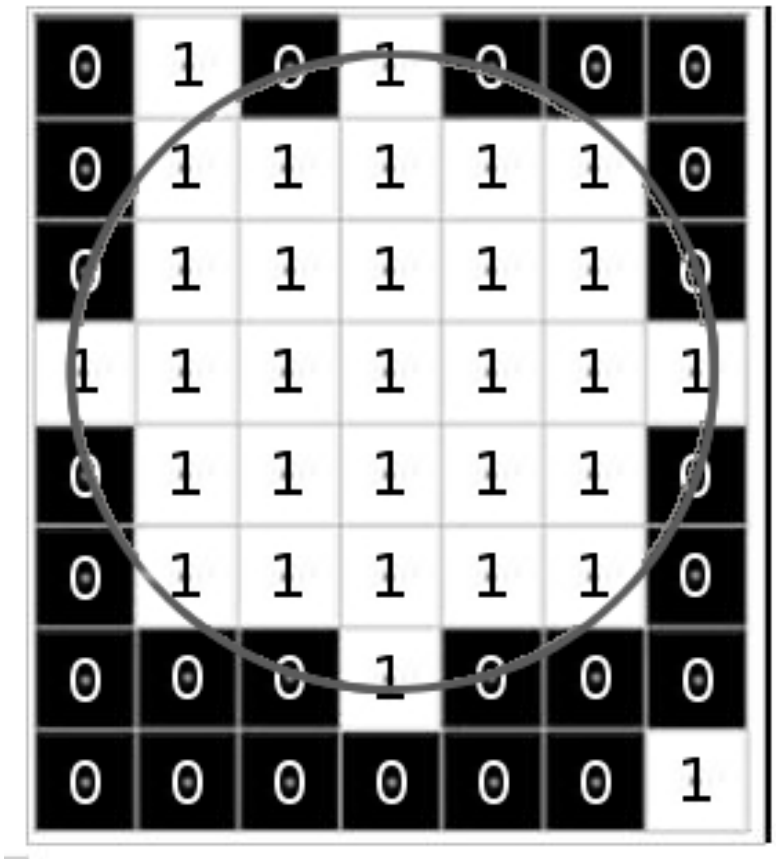
```
int radius = 0;

for(int i=1; i<=w-2; i++)
    for(int j=1; j<=h-2; j++)
        if(c[i][j]=='*')
        {
            bool moon = true;
            for(int R=1; R<=maxRad(i, j, w, h); R++)
            {
                for(int k=0; k<num[R]; k++)
                    if(c[i+DX[R][k]][j+DY[R][k]]=='.' || \
                       c[i+DX[R][k]][j-DY[R][k]]=='.' || \
                       c[i-DX[R][k]][j+DY[R][k]]=='.' || \
                       c[i-DX[R][k]][j-DY[R][k]]=='.')
                    {
                        moon = false;
                        break;
                    }
                if(moon == true && R>radius)
                    radius = R;
            }
        }
```

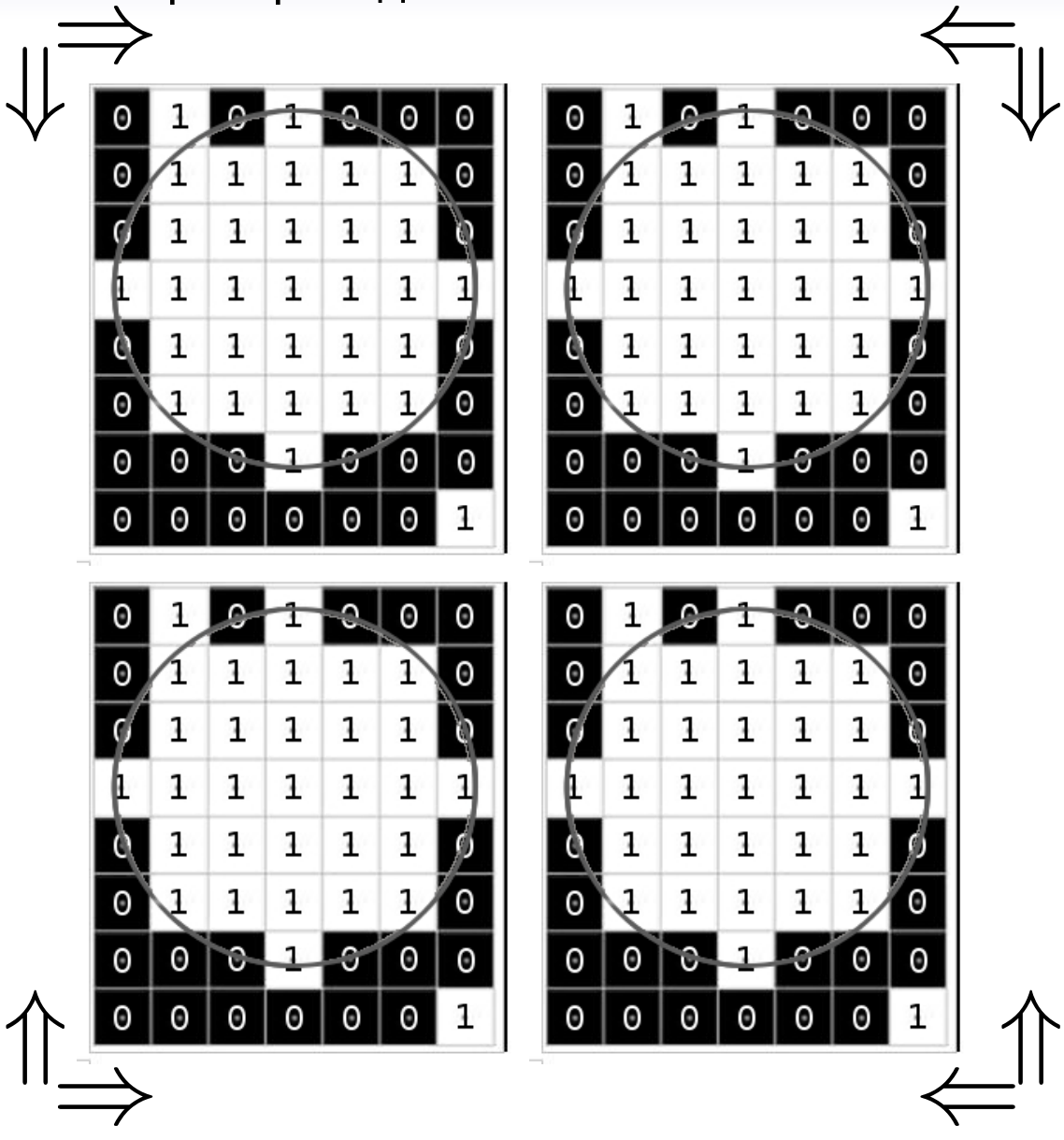
“Критерій Місяця Метод No.2“

Спосіб гарантувати, що ми знайшли Місяць радіуса R з центром C - для кожного пікселя знайти відстань до найближчого чорного пікселя. Піксель, найбільш віддалений від чорних, буде центром кола C. Відстань від краю до визначеного пікселя буде відповідати радіусу кола R.
Тож пробігаємо всі пікселі, крім “рамочки“, визначаючи кількість білих сусідів по вертикалі та горизонталі.

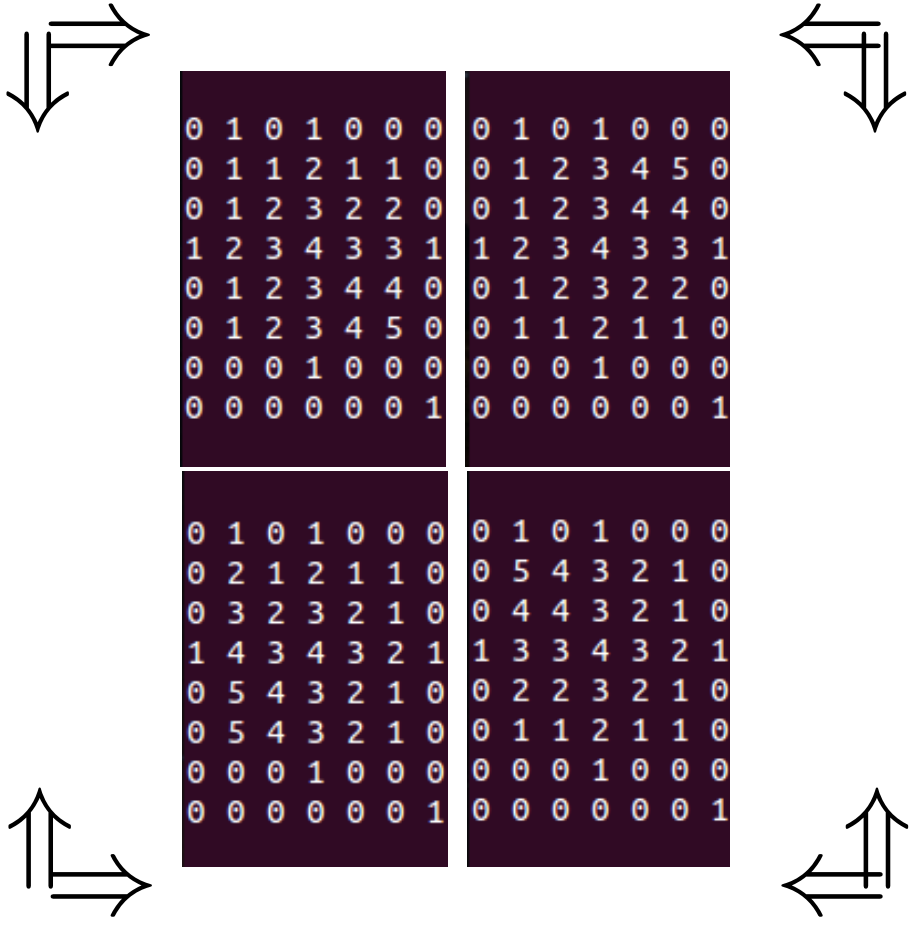
На першому кроці для білої клітинки відстань до найближчої чорної клітинки – **1**, для чорної – **0**

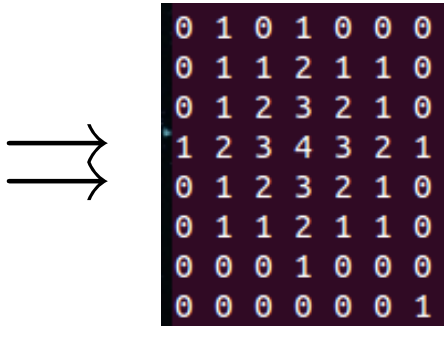


Виконується чотири проходи:



Значення у клітинці відповідає **мінімальному** зі значень у сусідніх **пройдених** клітинках. Фінальне значення - мінімальне з усіх матриць.





Визначення стартових матриць

```
int w, h, iC=-1, jC=-1;
char x;
int maxR=-1;
cin >> w >> h;

int picD[h][w], picU[h][w], picL[h][w], picR[h][w], pic[h][w];
for(int i=0; i < h; ++i) {
    for(int j=0; j < w; ++j) {
        cin >> x;
        if (x == '.'){
            picD[i][j] = 0;
            picU[i][j] = 0;
            picL[i][j] = 0;
            picR[i][j] = 0;
            pic[i][j] = 0; }
        else {
            picD[i][j] = 1;
            picU[i][j] = 1;
            picL[i][j] = 1;
            picR[i][j] = 1;
            pic[i][j] = 1;  }
    }
}
```

Визначення радіуса для Місяця з центром для певного напрямку руху по матриці

```
for(int i=1; i < h; ++i)
    for(int j=0; j < w; ++j)
        if (picU[i][j] != 0){
            picU[i][j] = std::min( picU[i-1][j],picU[i][j-1] )+1;
        }

for(int i=h-2; i > -1; --i)
    for(int j=w-1; j > -1; --j){
        if (picD[i][j] != 0){
            picD[i][j] = std::min( picD[i+1][j],picD[i][j+1] )+1;
        }
    }


for(int i=h-2; i > -1; --i)
    for(int j=1; j < w; ++j) {
        if (picL[i][j] != 0){
            picL[i][j] = std::min( picL[i+1][j],picL[i][j-1] )+1;
        }
    }


for(int i=1; i < h; ++i)
    for(int j=w-1; j > -1; --j){
        if (picR[i][j] != 0){
            picR[i][j] = std::min( picR[i-1][j],picR[i][j+1] )+1;
        }
    }
```

Визначення радіуса для Місяця

```
for(int i=0; i < h; ++i){
    for(int j=0; j < w; ++j){
        pic[i][j] = std::min(picD[i][j], picU[i][j]);
        pic[i][j] = std::min(picL[i][j], pic[i][j]);
        pic[i][j] = std::min(picR[i][j], pic[i][j]);
        std::cout<<pic[i][j]<<" ";
        if (maxR < pic[i][j]){
            maxR = pic[i][j];
            iC = i;
            jC = j;
        }
    }
    std::cout<<std::endl;
}
if (maxR <1) cout<<0;
else cout << maxR-1;
```