Copy-paste je smrť programátora

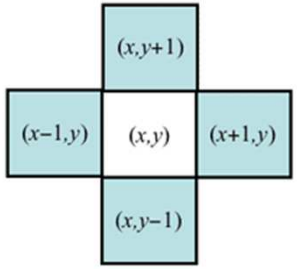
9/10 otvorených kódov Voltage malo tento copy-paste pattern

```
if (j > 0 && k > 0 && n[j - 1][k - 1] % 10 != 0) {
    n[j - 1][k - 1] = (n[j - 1][k - 1] + 1);
    if (n[j - 1][k - 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (j > 0 && n[j - 1][k] % 10 != 0) {
    n[j - 1][k] = (n[j - 1][k] + 1);
    if (n[j - 1][k] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (j > 0 && k < n[0].length - 1 && n[j - 1][k + 1] % 10 != 0) {
    n[j - 1][k + 1] = (n[j - 1][k + 1] + 1);
    if (n[j - 1][k + 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (k < n[0].length - 1 && n[j][k + 1] % 10 != 0) {
    n[j][k + 1] = (n[j][k + 1] + 1);
    if (n[j][k + 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (j < n.length - 1 && k < n[0].length - 1 && n[j + 1][k + 1] % 10 != 0) {
    n[j + 1][k + 1] = (n[j + 1][k + 1] + 1);
    if (n[j + 1][k + 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (j < n.length - 1 && n[j + 1][k] % 10 != 0) {
    n[j + 1][k] = (n[j + 1][k] + 1);
    if (n[j + 1][k] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (j < n.length - 1 && k > 0 && n[j + 1][k - 1] % 10 != 0) {
    n[j + 1][k - 1] = (n[j + 1][k - 1] + 1);
    if (n[j + 1][k - 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
    }
}
if (k > 0 && n[j][k - 1] % 10 != 0) {
    n[j][k - 1] = (n[j][k - 1] + 1);
    if (n[j][k - 1] % 10 == 0) {
        countDracula++;
        zeroCount++;
```

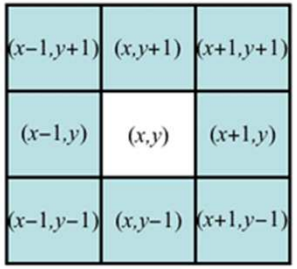
```
if (voltages[i - 1][j] != 0) {
    cellsToUpdate.add((i - 1) + "," + j);
}
if (j != 0) {
    if (voltages[i - 1][j - 1] != 0) {
        cellsToUpdate.add((i - 1) + "," + (j - 1));
    }
}
if (j != voltages[i].length - 1) {
    if (voltages[i - 1][j + 1] != 0) {
        cellsToUpdate.add((i - 1) + "," + (j + 1));
    }
}
}
if (i != voltages.length - 1) {
if (voltages[i + 1][j] != 0) {
    cellsToUpdate.add((i + 1) + "," + j);
}
if (j != 0) {
    if (voltages[i + 1][j - 1] != 0) {
        cellsToUpdate.add((i + 1) + "," + (j - 1));
    }
}
if (j != voltages[i].length - 1) {
    if (voltages[i + 1][j + 1] != 0) {
        cellsToUpdate.add((i + 1) + "," + (j + 1));
    }
}
}
if (j != 0) {
if (voltages[i][j - 1] != 0) {
    cellsToUpdate.add(i + "," + (j - 1));
}
}
if (j != voltages[i].length - 1) {
if (voltages[i][j + 1] != 0) {
    cellsToUpdate.add(i + "," + (j + 1));
}
}
```

```
if (voltages[i - 1][j] != 0) {
    cellsToUpdate.add((i - 1) + "," + j);
}
if (j != 0) {
    if (voltages[i - 1][j - 1] != 0) {
        cellsToUpdate.add((i - 1) + "," + (j - 1));
    }
}
if (j != voltages[i].length - 1) {
    if (voltages[i - 1][j + 1] != 0) {
        cellsToUpdate.add((i - 1) + "," + (j + 1));
    }
}
}
if (i != voltages.length - 1) {
if (voltages[i + 1][j] != 0) {
    cellsToUpdate.add((i + 1) + "," + j);
}
if (j != 0) {
    if (voltages[i + 1][j - 1] != 0) {
        cellsToUpdate.add((i + 1) + "," + (j - 1));
    }
}
if (j != voltages[i].length - 1) {
    if (voltages[i + 1][j + 1] != 0) {
        cellsToUpdate.add((i + 1) + "," + (j + 1));
    }
}
}
if (j != 0) {
if (voltages[i][j - 1] != 0) {
    cellsToUpdate.add(i + "," + (j - 1));
}
}
if (j != voltages[i].length - 1) {
if (voltages[i][j + 1] != 0) {
    cellsToUpdate.add(i + "," + (j + 1));
}
}
```

Skôr ako dámy DÚ3



4-neighbourhood



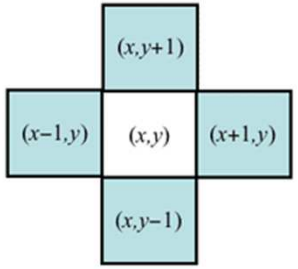
8-neighbourhood

Ak stojím na políčku [x,y] a potrebujem niečo riešiť v 4/8 smeroch

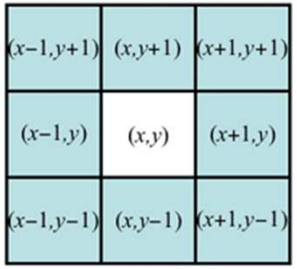
... dva vnorené mini-cykly

```
for (var dx = -1; dx <= 1; dx++) {  // -1, 0 1
      for (var dy = -1; dy <= 1; dy++) { // -1, 0 1
         if (dx == 0 && dy == 0) continue;    // 8 smerov              // 4 smery
                                                  if (Math.abs(dx) == Math.abs(dy)) continue;
         var nx = x +dx;
         var ny = y +dy;
         if (0 <= nx && nx < pole.length &&
            0 <= ny && ny < pole[nx].length) {
            //.. a riešim [nx, ny]
            System.out.println(nx+ "," + ny);
         }
      }
   }
}
```

Skôr ako dámy DÚ3



4-neighbourhood

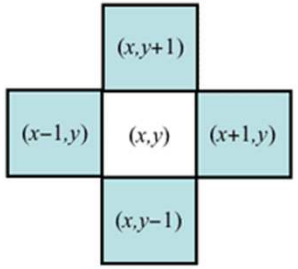


8-neighbourhood

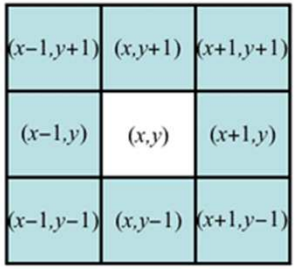
Ak stojím na políčku [x,y] a potrebujem niečo riešiť v 4/8 smeroch … smery mám v dátovej štruktúre, kľudne aj List, ...

```
int[][] directions = {{-1,-1}, {-1,0},  {-1,1},
                      {0, -1},          {0,1},
                      {1,-1},  {1,0},   {1,1}};
for (var dir : directions) {
    var dx = dir[0];
    var dy = dir[1];
    var nx = x+dx;
    var ny = y+dy;
    if (0 <= nx && nx < pole.length &&
        0 <= ny && ny < pole[nx].length) {
        //.. a riesim [nx, ny]
        System.out.println(nx+ "," + ny);
    }
}
```

Skôr ako dámy DÚ3



4-neighbourhood

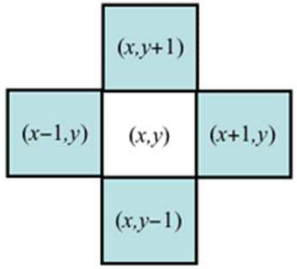


8-neighbourhood

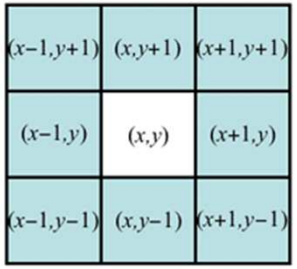
Ak stojím na políčku [x,y] a potrebujem niečo riešiť v 4/8 smeroch ... počul som o výnimkách

```
int[][] directions = {{-1,-1}, {-1,0}, {-1,1},
                      {0, -1},          {0,1},
                      {1,-1},  {1,0},  {1,1}};
for (var dir : directions) {
    var dx = dir[0];
    var dy = dir[1];
    var nx = x+dx;
    var ny = y+dy;
    try {
        //.. a riesim pole[nx, ny]
        System.out.println(pole[nx][ny]);
    } catch (IndexOutOfBoundsException e) {
        // indexoval som mimo
    }
}
```

Skôr ako dámy DÚ3



4-neighbourhood



8-neighbourhood

Ak stojím na políčku [x,y] a potrebujem niečo riešiť v 4/8 smeroch

... a náhodou hľadám niečo ako *piškvorku* dĺžky 5

```
int[][] directions = {{-1,-1}, {-1,0}, {-1,1},   {0, -1},  {0,1},   {1,-1},  {1,0},  {1,1}};
for (var dir : directions) {
    var dx = dir[0];
    var dy = dir[1];
    for (int k = 0; k < 5; k++) {
        var nx = x + k * dx;  // rovnica priamky
        var ny = y + k * dy;
        try {
            //.. a riesim pole[nx, ny]
            System.out.println(pole[nx][ny]);
        } catch (IndexOutOfBoundsException e) {
            // indexoval som mimo
        }
    }
}
```