Rešitev 7. domače naloge (Brezničelni kvadrati)

Osnovna rešitev

Vhodno matriko preberemo v tabelo tipa `boolean[][]`:

```
import java.util.Scanner;

public class BreznicelniKvadrati {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int stranica = sc.nextInt();

        boolean[][] matrika = new boolean[stranica][stranica];
        for (int i = 0;  i < stranica;  i++) {
            for (int j = 0;  j < stranica;  j++) {
                int t = sc.nextInt();
                matrika[i][j] = (t == 1);
            }
        }
        ...
    }
    ...
}
```

Število brezničelnih kvadratov v tabeli `matrika` lahko izračunamo tako, da za vsak element tabele določimo dolžino stranice največjega brezničelnega kvadrata, pri katerem ta element nastopa kot zgornje levo oglišče, in dobljene dolžine seštejemo. V primeru, prikazanem v besedilu naloge, je, denimo, element na koordinatah (1, 1) zgornje levo oglišče brezničelnega kvadrata s stranico dolžine 3:

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

Ker je to največji brezničelni kvadrat, pri katerem je element na koordinatah (1, 1) zgornje levo oglišče, lahko zaključimo, da je element (1, 1) zgornje levo oglišče treh brezničelnih kvadratov, kar pomeni, da k skupnemu številu brezničelnih kvadratov v matriki prispeva vrednost 3.

```
public class BreznicelniKvadrati {

    public static void main(String[] args) {
        ...
        int stKvadratov = 0;
        for (int i = 0;  i < stranica;  i++) {
            for (int j = 0;  j < stranica;  j++) {
                stKvadratov += najvecjiKvadrat(matrika, i, j);
            }
        }
        System.out.println(stKvadratov);
    }
```

```
    ...
}
```

Največji brezničelni kvadrat, pri katerem element na koordinatah (v, s) nastopa kot zgornje levo oglišče, lahko poiščemo enostavno tako, da v zanki povečujemo števec t, dokler kvadrat velikosti $t \times t$ z zgornjim levim ogliščem na koordinatah (v, s) obstaja in sestoji iz samih enic:

```
public class BreznicelniKvadrati {
    ...
    private static int najvecjiKvadrat(
            boolean[][] matrika, int vrstica, int stolpec) {

        int stranica = 1;
        while (sameEnice(matrika, vrstica, stolpec, stranica)) {
            stranica++;
        }
        return (stranica - 1);
    }
    ...
}
```

Preostane nam še metoda, ki preveri, ali kvadrat z zgornjim levim ogliščem na koordinatah (v, s) in stranico dolžine t obstaja in sestoji iz samih enic:

```
public class BreznicelniKvadrati {
    ...
    private static boolean sameEnice(boolean[][] matrika,
            int vrstica, int stolpec, int stranica) {

        // Ali kvadrat stranica × stranica obstaja?
        if (vrstica + stranica > matrika.length) {
            return false;
        }
        if (stolpec + stranica > matrika[vrstica].length) {
            return false;
        }

        // Je kvadrat stranica × stranica sestavljen iz samih enic?
        for (int i = 0;  i < stranica;  i++) {
            for (int j = 0;  j < stranica;  j++) {
                if (!matrika[vrstica + i][stolpec + j]) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

Učinkovitejša rešitev

Rešitev, ki smo jo prikazali, ni najbolj učinkovita. Če kot vhod podamo matriko $n \times n$, sestavljeno iz samih enic (to je sicer najbolj neugoden primer), potrebujemo kar $\sum_{i=1}^{n} i^2(n-$

$i+1)^2 = O(n^5)$ dostopov do posameznih elementov matrike. Sedaj pa bomo prikazali rešitev, ki v vsakem primeru zahteva le $O(n^2)$ osnovnih operacij, saj se po matriki sprehodi le enkrat.

Ključ do rešitve je ločena matrika (imenujmo jo K), ki za vsak element vhodne matrike (označimo jo z V) hrani število brezničelnih kvadratov, pri katerih ta element nastopa kot spodnje desno oglišče. V primeru, prikazanem v besedilu naloge, izgleda matrika K takole:

$$\begin{matrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 1 \end{matrix}$$

Če vrednosti v matriki K seštejemo, dobimo iskano število brezničelnih kvadratov v matriki V.

Matriko K je mogoče zgraditi kar med samim sprehodom po matriki V. Kako izračunamo vrednost elementa na koordinatah (i, j)? Če velja $V[i, j] = 0$, potem je $K[i, j] = 0$, saj ničla seveda ne more biti spodnje desno oglišče brezničelnega kvadrata. V nasprotnem primeru pa je vrednost $K[i, j]$ določena z vrednostmi $K[i-1, j-1]$, $K[i-1, j]$ in $K[i, j-1]$. Če so vse tri vrednosti enake k, potem je $K[i, j] = k+1$ (v obravnavanem primeru, denimo, imamo $K[2, 2] = K[2, 3] = K[3, 2] = 2$ in zato $K[3, 3] = 3$). Če to ne drži, pa odloča najmanjša izmed vrednosti $K[i-1, j-1]$, $K[i-1, j]$ in $K[i, j-1]$:

$$K[i, j] = \min\{K[i-1, j-1],\ K[i-1, j],\ K[i, j-1]\} + 1 \tag{1}$$

Gornja formula pokriva tudi primer $K[i-1, j-1] = K[i-1, j] = K[i, j-1]$, če privzamemo $K[i, j] = 0$ za vse $i < 0$ in $j < 0$, pa velja še za vrednosti $i = 0$ in $j = 0$.

V sledečem programu je matrika V predstavljena s tabelo `vhodna`, matrika K pa s tabelo `kvadrati`. Da se izognemo preverjanju pogoja $i = 0$ oziroma $j = 0$, je tabela `kvadrati` za eno vrstico in en stolpec večja od matrike K. Element `kvadrati[i][j]` potemtakem vsebuje vrednost $K[i-1, j-1]$.

```
import java.util.Scanner;

public class BreznicelniKvadrati {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int stranica = sc.nextInt();

        boolean[][] matrika = new boolean[stranica][stranica];
        for (int i = 0;  i < stranica;  i++) {
            for (int j = 0;  j < stranica;  j++) {
                int t = sc.nextInt();
                matrika[i][j] = (t == 1);
            }
        }

        int[][] kvadrati = new int[stranica + 1][stranica + 1];
        int vsota = 0;
        for (int i = 1;  i <= stranica;  i++) {
            for (int j = 1;  j <= stranica;  j++) {
                if (matrika[i - 1][j - 1]) {
                    kvadrati[i][j] = min3( kvadrati[i - 1][j - 1], kvadrati[i - 1][j],
```

```
                                            kvadrati[i][j - 1]) + 1;
                    vsota += kvadrati[i][j];
                }
            }
        }
        System.out.println(vsota);
    }

    private static int min3(int a, int b, int c) {
        return Math.min(Math.min(a, b), c);
    }
}
```