Dodatne naloge – Razredi in objekti 2

Splošna navodila

Pri nalogi 5 lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek, pri ostalih nalogah pa z množico testnih razredov in pripadajočih izhodnih datotek.

Pri nalogah, ki zahtevajo izdelavo razreda, so navedeni zgolj javno dostopni elementi, ki jih mora razred vsebovati. Seveda pa lahko po potrebi dodajate tudi svoje (privatne in javne) elemente.

1 Štiri v vrsto

Naloga

Pri igri Štiri v vrsto igralca izmenično spuščata žetone (npr. prvi igralec rdeče, drugi pa zelene) v navpični pravokotni okvir z m vrsticami in n stolpci. Igralec na potezi spusti žeton v enega od stolpcev, ki še niso povsem polni.

Vrstice in stolpce igralnega okvirja označujemo z indeksi od 0 naprej. Indeks 0 predstavlja spodnjo (!) vrstico in levi stolpec. Prvi igralec (to je tisti, ki odigra prvo potezo v igri) ima indeks 0, drugi pa 1.

V vseh testnih razredih imajo parametri metod smiselne vrednosti. Na primer, parameter, ki podaja indeks vrstice, ima vrednost med 0 in vključno $m - 1$.

Napišite razred `StiriVVrsto`, ki omogoča simulacijo igre Štiri v vrsto. V razredu definirajte sledeče javno dostopne elemente:

- `public StiriVVrsto(int stVrstic, int stStolpcev)`

  Ustvari objekt, ki predstavlja začetno stanje igre Štiri v vrsto na okvirju s `stVrstic` (m) vrsticami in `stStolpcev` (n) stolpci. V vseh testnih primerih velja $m \in [1, 100]$ in $n \in [1, 100]$.

- `public int vrniSteviloVrstic()`

  Vrne število vrstic igralnega okvirja.

- `public int vrniSteviloStolpcev()`

  Vrne število stolpcev igralnega okvirja.

- `public boolean vrzi(int stolpec)`

  Če je stolpec z indeksom `stolpec` že poln, naj metoda ne naredi ničesar in naj zgolj vrne `false`, v nasprotnem primeru pa naj simulira potezo (met žetona v podani stolpec in predaja poteze nasprotniku) in vrne `true`.

- `public int naPotezi()`

  Vrne indeks igralca, ki je trenutno na potezi. Na začetku igre naj metoda torej vrne vrednost 0.

- `public int vsebina(int vrstica, int stolpec)`

  Če je polje v vrstici z indeksom `vrstica` in stolpcu z indeksom `stolpec` prazno, vrne vrednost -1, sicer pa vrne indeks igralca, ki mu pripada žeton v tem polju.

- `public int najdaljseZaporedje(int igralec)`

  Vrne dolžino najdaljšega strnjenega zaporedja žetonov igralca z indeksom `igralec` v neki vrstici, stolpcu ali diagonali.

- `public int izid()`

  Če je prvi igralec postavil najmanj štiri zaporedne žetone v neki vrstici, stolpcu ali diagonali, drugemu pa to ni uspelo, naj metoda vrne vrednost 0. Če se s tem dosežkom lahko ponaša drugi igralec, prvi pa ne, naj metoda vrne vrednost 1. Če sta najmanj štiri zaporedne žetone postavila oba ali pa nobeden od njiju, naj vrne vrednost -1.

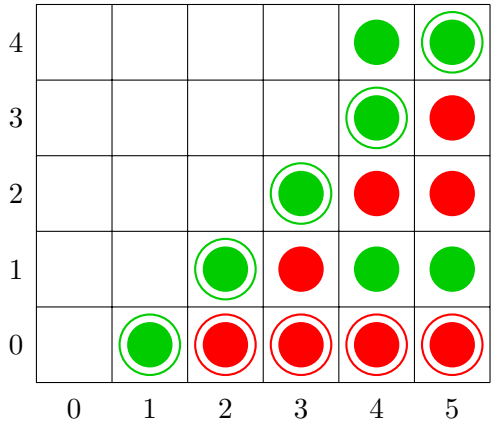
Testni primer 9

Testni razred (in pripadajoči izhod):

```
public class Test09 {

    public static void main(String[] args) {
        StiriVVrsto igra = new StiriVVrsto(5, 6);
        System.out.println(igra.naPotezi());   // 0
        System.out.println(igra.vrzi(2));      // true
        System.out.println(igra.vrzi(1));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(2));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // false
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.naPotezi());   // 1
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.naPotezi());   // 0

        System.out.println(igra.vsebina(0, 0));   // -1
        System.out.println(igra.vsebina(2, 3));   // 1
        System.out.println(igra.vsebina(1, 3));   // 0
        System.out.println(igra.vsebina(3, 3));   // -1

        System.out.println(igra.najdaljseZaporedje(0));   // 4
```

```
        System.out.println(igra.najdaljseZaporedje(1));    // 5
        System.out.println(igra.izid());                   // -1
    }
}
```

Slika 1 prikazuje končno vsebino igralnega okvirja za gornji primer. Označeni sta najdaljši zaporedji žetonov za oba igralca. Ker sta oba igralca postavila najmanj štiri zaporedne žetone v neki vrstici, stolpcu ali diagonali, metoda `izid` vrne vrednost -1.



Slika 1: *Vsebina igralnega okvirja po koncu izvajanja kode v testnem razredu 9.*

2 Obedujoči filozofi

Naloga

Skupina filozofov sedi za okroglo mizo. Vsak filozof ima svoj krožnik s hrano, po dva soseda pa si delita jedilno paličico (slika 2). Filozofi večino časa razmišljajo, občasno pa kdo od njih postane lačen in želi pričeti jesti. Filozof za to opravilo potrebuje obe paličici, levo in desno. Paličici lahko vzame samo, če sta prosti — torej, če njegov levi sosed ne drži svoje desne paličice in če njegov desni sosed ne drži svoje leve paličice. Ko filozof prične jesti, prime obe paličici in ju drži, dokler ne konča. V tem času seveda preprečuje svojemu levemu in desnemu sosedu, da bi pričela s prehranjevanjem. Ko filozof preneha jesti, odloži (in s tem sprosti) obe paličici.[1]

Za predstavitev posameznih filozofov napišite razred `Filozof` s sledečimi elementi:

- `public Filozof(String ime)`:

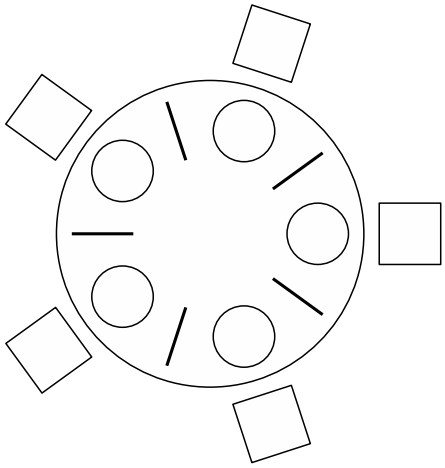
  Ustvari objekt, ki predstavlja filozofa za mizo. Parameter `ime` podaja njegovo ime. Filozof ob svoji »stvaritvi« ne drži niti leve niti desne paličice.

- `public String vrniIme()`:

  Vrne ime filozofa `this`.

- `public void nastaviSoseda(Filozof levi, Filozof desni)`:

[1] Gre za znan problem iz teorije operacijskih sistemov. Filozofi predstavljajo procese, ki tečejo v računalniku, paličice pa vire (npr. datoteke, naprave, procesorski čas, ...), ki jih procesi potrebujejo. Problem obedujočih filozofov tako ilustrira borbo procesov za omejene vire.



Slika 2: *Miza za pet obedujočih filozofov.*

Nastavi oba soseda filozofa `this`. Levi sosed postane filozof `levi`, desni pa filozof `desni`.

- `public boolean jeLeviSosedOd(Filozof f)`:

  Vrne `true` natanko v primeru, če je filozof `this` levi sosed filozofa `f`.

- `public boolean jeDesniSosedOd(Filozof f)`:

  Vrne `true` natanko v primeru, če je filozof `this` desni sosed filozofa `f`.

- `public Filozof[] vrniSoseda()`:

  Vrne tabelo z dvema elementoma. Prvi element je referenca na levega, drugi pa referenca na desnega soseda filozofa `this`.

- `public int kolikoPalicicDrzi()`:

  Vrne število paličic, ki jih trenutno drži filozof `this`. To število je lahko samo 0, 1 ali 2.

- `public boolean primiLevo()`:

  Če filozof `this` že drži paličico na svoji levi, naj se ne zgodi nič, metoda pa naj vrne `true`. Če jo drži njegov levi sosed, naj se prav tako ne zgodi nič, metoda pa naj vrne `false`. Če je paličica prosta, naj jo prime (in s tem zasede) filozof `this`, metoda pa naj vrne `true`.

- `public boolean primiDesno()`:

  Če filozof `this` že drži paličico na svoji desni, naj se ne zgodi nič, metoda pa naj vrne `true`. Če jo drži njegov desni sosed, naj se prav tako ne zgodi nič, metoda pa naj vrne `false`. Če je paličica prosta, naj jo prime (in s tem zasede) filozof `this`, metoda pa naj vrne `true`.

- `public void izpustiLevo()`:

  Če filozof `this` drži paličico na svoji levi, jo izpusti, sicer pa se ne zgodi nič.

- `public void izpustiDesno()`:

  Če filozof `this` drži paličico na svoji desni, jo izpusti, sicer pa se ne zgodi nič.

- `public static int steviloJedcev(Filozof[] filozofi)`:

  Vrne število filozofov v tabeli `filozofi`, ki držijo obe paličici.

- `public int steviloFilozofov()`:

  Vrne število filozofov za mizo, če privzamemo, da je filozof `this` eden od njih. Lahko predpostavite, da sta za mizo vsaj dva filozofa.

  Namig: pričnite s filozofom `this` in potujte po verigi desnih (ali levih) sosedov, dokler ne pridete spet do filozofa `this`.

Pri vseh metodah razen `vrniIme` in `nastaviSoseda` lahko predpostavite, da ima filozof `this` že pravilno (in konsistentno) nastavljena oba soseda. To pomeni, da vam ni treba preverjati, ali levi in desni sosed sploh obstajata.

Testni primer 9

Testni razred (in pripadajoči izhod):

```
public class Test09 {

    public static void main(String[] args) {
        Filozof slavoj = new Filozof("Slavoj");
        Filozof renata = new Filozof("Renata");
        Filozof mladen = new Filozof("Mladen");
        Filozof tine = new Filozof("Tine");
        Filozof spomenka = new Filozof("Spomenka");

        slavoj.nastaviSoseda(spomenka, renata);
        renata.nastaviSoseda(slavoj, mladen);
        mladen.nastaviSoseda(renata, tine);
        tine.nastaviSoseda(mladen, spomenka);
        spomenka.nastaviSoseda(tine, slavoj);

        System.out.println(renata.jeLeviSosedOd(slavoj));     // false
        System.out.println(spomenka.jeLeviSosedOd(slavoj));   // true
        System.out.println(tine.jeDesniSosedOd(mladen));      // true
        System.out.println(spomenka.jeDesniSosedOd(mladen));  // false
        System.out.println("---");

        izpisiSoseda(slavoj);      // Spomenka/Renata
        izpisiSoseda(renata);      // Slavoj/Mladen
        izpisiSoseda(mladen);      // Renata/Tine
        izpisiSoseda(tine);        // Mladen/Spomenka
        izpisiSoseda(spomenka);    // Tine/Slavoj
        System.out.println("---");

        System.out.println(slavoj.primiLevo());     // true
        System.out.println(renata.primiDesno());    // true
        System.out.println(mladen.primiLevo());     // false
        System.out.println(tine.primiDesno());      // true
        System.out.println(spomenka.primiDesno());  // false
```

```
        System.out.println(slavoj.primiLevo());      // true
        System.out.println(slavoj.primiDesno());     // true
        System.out.println(tine.primiLevo());        // true
        renata.izpustiLevo();
        slavoj.izpustiDesno();
        System.out.println(renata.primiLevo());      // true
        System.out.println("---");

        System.out.println(slavoj.kolikoPalicicDrzi());     // 1
        System.out.println(renata.kolikoPalicicDrzi());     // 2
        System.out.println(mladen.kolikoPalicicDrzi());     // 0
        System.out.println(tine.kolikoPalicicDrzi());       // 2
        System.out.println(spomenka.kolikoPalicicDrzi());   // 0
        System.out.println("---");

        Filozof[] filozofi = {slavoj, renata, mladen, tine, spomenka};
        System.out.println(Filozof.steviloJedcev(filozofi));  // 2
        System.out.println("---");

        System.out.println(slavoj.steviloFilozofov());     // 5
        System.out.println(renata.steviloFilozofov());     // 5
        System.out.println(mladen.steviloFilozofov());     // 5
        System.out.println(tine.steviloFilozofov());       // 5
        System.out.println(spomenka.steviloFilozofov());   // 5
    }

    private static void izpisiSoseda(Filozof filozof) {
        Filozof[] soseda = filozof.vrniSoseda();
        System.out.printf("%s/%s%n", soseda[0].vrniIme(), soseda[1].vrniIme());
    }
}
```

3 Krožki

Naloga

Učenci osnovne šole Jožeta Goriška obiskujejo različne krožke. Vsak krožek se izvaja ob fiksnem terminu (enkrat tedensko po dve uri), sprejme pa lahko največ k učencev (vrednost k je od krožka do krožka lahko različna). Učenec se lahko včlani v krožek natanko tedaj, ko so izpolnjeni vsi sledeči pogoji:

1. Učenec je v danem trenutku član manj kot m krožkov. Omejitev m je enaka za vse učence na šoli.
2. Krožek ima v danem trenutku manj kot k članov.
3. Termin krožka se ne prekriva z nobenim od terminov krožkov, ki jih učenec že obiskuje.

Napišite razreda `Ucenec` in `Krozek` tako, da bodo njuni objekti predstavljali posamezne učence oziroma krožke. Razred `Ucenec` naj vsebuje sledeče elemente (seveda lahko po potrebi dodajate tudi svoje):

- `public static void nastaviMaksObremenitev(int maksObremenitev)`

  Nastavi konstanto m (največje možno število krožkov, v katere je lahko včlanjen posamezen učenec) na vrednost `maksObremenitev`. V vsakem testnem primeru, v katerem se vsaj enkrat pokliče metoda `vclani`, se metoda `nastaviMaksObremenitev` pokliče natanko enkrat, in to takoj na začetku izvajanja programa.

- `public Ucenec(String ime, String priimek)`

  Ustvari objekt, ki predstavlja učenca s podanim imenom in priimkom. Učenec ob svoji »stvaritvi« ni član nobenega krožka.

- `public String vrniIP()`

  Vrne ime in priimek učenca `this` kot niz sledeče oblike:

  $ime␣priimek$

- `public boolean vclani(Krozek krozek)`

  Če je učenec `this` že član krožka `krozek`, naj metoda zgolj vrne `true`. Če učenec ne izpolnjuje pogojev za včlanitev v krožek, naj metoda zgolj vrne `false`. Če pa se učenec lahko včlani v krožek, naj metoda to stori in vrne `true`.

- `public void izclani(Krozek krozek)`

  Učenca `this` izpiše iz krožka `krozek`. Če učenec ni član podanega krožka, naj se ne zgodi nič.

- `public int steviloKrozkov()`

  Vrne število krožkov, v katere je včlanjen učenec `this`.

Razred `Krozek` pa naj vsebuje sledeče elemente:

- `public Krozek(String naziv, int dan, int ura, int kvota)`

  Ustvari objekt, ki predstavlja krožek s podanim nazivom. Krožek se izvaja v dnevu `dan` (1: ponedeljek, 2: torek, ...), prične pa se ob uri `ura`. Krožek lahko ima največ `kvota` (k) članov.

- `public String vrniNaziv()`:

  Vrne naziv krožka `this`.

- `public int steviloClanov()`:

  Vrne število članov krožka `this`.

Testni primer 8

Testni razred (in pripadajoči izhod):

```
public class Test08 {

    private static final String LOCILO = "--------------------";

    public static void main(String[] args) {
        Ucenec.nastaviMaksObremenitev(3);
```

```
        Ucenec anja = new Ucenec("Anja", "Antolinc");
        Ucenec bojan = new Ucenec("Bojan", "Bevk");
        Ucenec cvetka = new Ucenec("Cvetka", "Cirnik");
        Ucenec denis = new Ucenec("Denis", "Divjak");
        Ucenec eva = new Ucenec("Eva", "Erlah");

        Krozek matematika = new Krozek("matematika", 1, 13, 2);
        Krozek sah = new Krozek("sah", 1, 14, 3);
        Krozek dramatika = new Krozek("dramatika", 1, 16, 2);
        Krozek ples = new Krozek("ples", 3, 15, 4);
        Krozek nogomet = new Krozek("nogomet", 4, 13, 2);

        Ucenec[] ucenci = {anja, bojan, cvetka, denis, eva};
        Krozek[] krozki = {matematika, sah, dramatika, ples, nogomet};

        System.out.println(anja.vclani(matematika));   // true
        System.out.println(anja.vclani(sah));          // false
        System.out.println(anja.vclani(dramatika));    // true
        System.out.println(anja.vclani(ples));         // true
        System.out.println(anja.vclani(nogomet));      // false
        System.out.println(anja.vclani(matematika));   // true
        System.out.println(LOCILO);

        System.out.println(bojan.vclani(matematika));  // true
        System.out.println(bojan.vclani(dramatika));   // true
        System.out.println(bojan.vclani(nogomet));     // true
        System.out.println(LOCILO);

        System.out.println(cvetka.vclani(sah));        // true
        System.out.println(cvetka.vclani(dramatika));  // false
        System.out.println(cvetka.vclani(ples));       // true
        System.out.println(LOCILO);

        System.out.println(denis.vclani(sah));         // true
        System.out.println(denis.vclani(dramatika));   // false
        System.out.println(denis.vclani(ples));        // true
        System.out.println(LOCILO);

        System.out.println(eva.vclani(sah));           // true
        System.out.println(eva.vclani(dramatika));     // false
        System.out.println(eva.vclani(ples));          // true
        System.out.println(LOCILO);

        izpisiVse(ucenci, krozki);   // Anja Antolinc -> 3
                                     // Bojan Bevk -> 3
                                     // Cvetka Cirnik -> 2
                                     // Denis Divjak -> 2
                                     // Eva Erlah -> 2
                                     // matematika -> 2
                                     // sah -> 3
                                     // dramatika -> 2
```

```
                                    // ples -> 4
                                    // nogomet -> 1
        System.out.println(LOCILO);

        anja.izclani(matematika);
        bojan.izclani(nogomet);
        cvetka.izclani(ples);
        denis.izclani(sah);
        eva.izclani(nogomet);

        izpisiVse(ucenci, krozki);  // Anja Antolinc -> 2
                                    // Bojan Bevk -> 2
                                    // Cvetka Cirnik -> 1
                                    // Denis Divjak -> 1
                                    // Eva Erlah -> 2
                                    // matematika -> 1
                                    // sah -> 2
                                    // dramatika -> 2
                                    // ples -> 3
                                    // nogomet -> 0
        System.out.println(LOCILO);

        System.out.println(eva.vclani(dramatika));   // false
        System.out.println(eva.vclani(nogomet));     // true
        System.out.println(cvetka.vclani(nogomet));  // true
        System.out.println(bojan.vclani(nogomet));   // false
        System.out.println(anja.vclani(sah));        // true
        System.out.println(LOCILO);

        izpisiVse(ucenci, krozki);  // Anja Antolinc -> 3
                                    // Bojan Bevk -> 2
                                    // Cvetka Cirnik -> 2
                                    // Denis Divjak -> 1
                                    // Eva Erlah -> 3
                                    // matematika -> 1
                                    // sah -> 3
                                    // dramatika -> 2
                                    // ples -> 3
                                    // nogomet -> 2
    }

    private static void izpisiVse(Ucenec[] ucenci, Krozek[] krozki) {
        for (int i = 0;  i < ucenci.length;  i++) {
            System.out.printf("%s -> %d%n",
                              ucenci[i].vrniIP(), ucenci[i].steviloKrozkov());
        }
        for (int i = 0;  i < krozki.length;  i++) {
            System.out.printf("%s -> %d%n",
                              krozki[i].vrniNaziv(), krozki[i].steviloClanov());
        }
    }
```

```
}
```

4 Praštevila

Naloga

Napišite razred `Prastevila`, čigar objekt hrani t.i. *trenutno praštevilo*, ki ga je mogoče nastavljati, vračati in spreminjati. Razred naj vsebuje sledeče javno dostopne elemente:

- `public Prastevila()`

  Ustvari objekt, v katerem je trenutno praštevilo nastavljeno na 2.

- `public void nastaviTrenutno(int prastevilo)`

  Nastavi trenutno praštevilo na `prastevilo`. V vseh testnih primerih je `prastevilo` praštevilo z intervala $[2, 10^9]$.

- `public int vrniTrenutno()`

  Vrne trenutno praštevilo, ki ga hrani objekt `this`.

- `public int naslednje()`

  Trenutno praštevilo nastavi na najmanjše praštevilo, ki je večje od trenutnega, in dobljeno število tudi vrne.

- `public int prejsnje()`

  Če je trenutno praštevilo enako 2, naj metoda zgolj vrne 2, sicer pa naj trenutno praštevilo nastavi na največje praštevilo, ki je manjše od trenutnega, in dobljeno število tudi vrne.

Skupno število klicev metod `naslednje` in `prejsnje` je v vseh testnih primerih omejeno na 1000.

Testni primer 6

Testni razred (in pripadajoči izhod):

```
public class Test06 {

    public static void main(String[] args) {
        Prastevila prastevila = new Prastevila();
        System.out.println(prastevila.vrniTrenutno());  // 2
        System.out.println(prastevila.prejsnje());      // 2
        System.out.println(prastevila.vrniTrenutno());  // 2
        System.out.println(prastevila.naslednje());     // 3
        System.out.println(prastevila.vrniTrenutno());  // 3

        prastevila.nastaviTrenutno(17);
        System.out.println(prastevila.vrniTrenutno());  // 17
        System.out.println(prastevila.prejsnje());      // 13
        System.out.println(prastevila.prejsnje());      // 11
        System.out.println(prastevila.naslednje());     // 13
```

```
        System.out.println(prastevila.vrniTrenutno());  // 13

        prastevila.nastaviTrenutno(47);
        System.out.println(prastevila.vrniTrenutno());  // 47
        System.out.println(prastevila.naslednje());     // 53
        System.out.println(prastevila.naslednje());     // 59
        System.out.println(prastevila.prejsnje());      // 53
        System.out.println(prastevila.vrniTrenutno());  // 53
    }
}
```

5 Poti po grafu (⋆)

Naloga

Napišite program, ki izpiše vse aciklične poti med podanim začetnim in končnim vozliščem v podanem enostavnem usmerjenem grafu brez zank.

Vhod

V prvi vrstici vhoda je zapisano število vozlišč ($n \in [2, 100]$), v drugi pa število povezav ($m \in [1, n(n-1)]$). V vsaki od naslednjih m vrstic sta podani celi števili $u \in [0, n-1]$ in $v \in [0, n-1] \backslash \{u\}$, ločeni s presledkom, ki predstavljata indeksa začetnega in končnega vozlišča povezave. Sledita še dve vrstici, v katerih sta zapisani števili $v_z \in [0, n-1]$ in $v_k \in [0, n-1] \backslash \{v_z\}$, ki predstavljata indeksa začetnega in končnega vozlišča poti.

Izhod

Vsako pot izpišite v svoji vrstici. Pot izpišite tako, da po vrsti navedete indekse vozlišč na njej. Indeksi naj bodo ločeni s presledkom.

Poti naj se izpišejo v leksikografskem vrstnem redu. Pot $u_1 \to u_2 \to \ldots \to u_k$ naj se torej izpiše pred potjo $v_1 \to v_2 \to \ldots \to v_l$ natanko tedaj, ko je izpolnjen eden od sledečih pogojev:

- $k < l$ in $u_i = v_i$ za vse $i \in \{1, \ldots, k\}$;
- obstaja $i \in \{1, \ldots, k\}$, tako da velja $u_i < v_i$ in $u_j = v_j$ za vse $j < i$.

Število poti v nobenem testnem primeru ne bo večje od 100.

Testni primer 7

V tem testnem primeru iščemo poti od vozlišča 3 do vozlišča 1 v grafu na sliki 3.

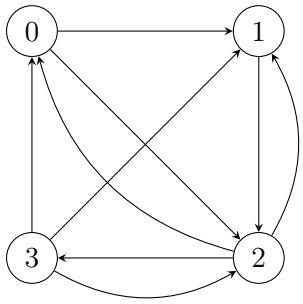
Vhod:

```
4
9
0 1
0 2
```

```
1 2
2 0
2 1
2 3
3 0
3 1
3 2
3
1
```

Izhod:

```
3 0 1
3 0 2 1
3 1
3 2 0 1
3 2 1
```



Slika 3: *Graf v testnem primeru 7.*