Domáca úloha č. 2

(Termín odovzdania úlohy: **do utorka 30. apríla 2024, 9:50**, t. j. do začiatku jedenástych cvičení.)

Vrcholovým pokrytím (angl. *vertex cover*) neorientovaného grafu G rozumieme množinu jeho vrcholov takú, že každá hrana grafu G má aspoň jeden z koncových vrcholov v tejto množine. Každá hrana je teda pokrytá aspoň jedným vrcholom vrcholového pokrytia.
Veľkosťou vrcholového pokrytia rozumieme počet vrcholov, ktoré ho tvoria. *Najmenšie vrcholové pokrytie* neorientovaného grafu G je jeho vrcholové pokrytie s najmenšou možnou veľkosťou spomedzi všetkých vrcholových pokrytí grafu G. Môže teda existovať aj viacero najmenších vrcholových pokrytí jedného grafu.
Pre každý neorientovaný graf G tvorí množina všetkých jeho vrcholov vrcholové pokrytie grafu G; každý neorientovaný graf tak má aspoň jedno vrcholové pokrytie a v dôsledku toho aj aspoň jedno najmenšie vrcholové pokrytie. Prázdna množina vrcholov je vrcholovým pokrytím práve vtedy, keď graf neobsahuje žiadnu hranu. Ak navyše graf G obsahuje slučku vo vrchole v, musí byť vrchol v prvkom každého vrcholového pokrytia.

Priložený ZIP archív obsahuje balík `graphs` a v ňom všetky triedy pre grafy z prednášky, ako aj kostru triedy `VertexCovers`, ktorá má realizovať – pre neorientovaný graf `g` z argumentu konštruktora tejto triedy – hľadanie jeho vrcholových pokrytí s určitými vlastnosťami.

V triede `VertexCovers` implementujte:

- Konštruktor, ktorý dostane ako argument neorientovaný graf `g`, v ktorom sa bude hľadanie vrcholových pokrytí realizovať.
- Metódu

  ```
  public boolean isVertexCover(Set<Integer> vertices),
  ```

  ktorá ako vstup dostane nejakú množinu celých čísel a na výstupe vráti `true` práve vtedy, keď táto množina reprezentuje vrcholové pokrytie grafu `g` z argumentu konštruktora. V prípade, že niektorý z prvkov množiny `vertices` nie je vrcholom grafu `g`, vráti metóda na výstupe booleovskú hodnotu `false`.
- Metódu

  ```
  public int minimumVertexCoverSize(),
  ```

  ktorá vráti veľkosť najmenšieho vrcholového pokrytia grafu `g` z argumentu konštruktora.
- Metódu

  ```
  public Iterator<Set<Integer>> allVertexCoversIterator(),
  ```

  ktorá vráti iterátor postupne prechádzajúci cez všetky vrcholové pokrytia grafu `g` z argumentu konštruktora.
- Metódu

  ```
  public Iterator<Set<Integer>> fixedSizeVertexCoversIterator(int size),
  ```

  ktorá vráti iterátor postupne prechádzajúci cez všetky vrcholové pokrytia veľkosti `size` grafu `g` z argumentu konštruktora. Celočíselná hodnota `size` tu môže byť ľubovoľná – napríklad aj záporná alebo väčšia, než počet vrcholov grafu `g` (v takých prípadoch bude výstupom metódy iterátor, ktorý nevráti žiaden prvok; t. j. napríklad iterátor `Collections.emptyIterator()`).
- Metódu

  ```
  public Iterator<Set<Integer>> minimumVertexCoversIterator(),
  ```

  ktorá vráti iterátor postupne prechádzajúci cez všetky najmenšie vrcholové pokrytia grafu `g` z argumentu konštruktora.

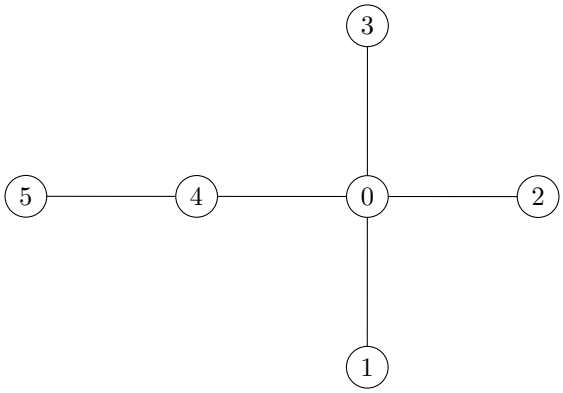
Všetky iterátory vracané na výstupe uvedených metód by mali spĺňať nasledujúce podmienky:

- Každé vrcholové pokrytie na výstupe metódy `next` iterátora by malo byť reprezentované ako *nemodifikovateľná* množina vrcholov, ktoré toto pokrytie tvoria.
- Každý z iterátorov môže cez svoje vrcholové pokrytia prechádzať v ľubovoľnom poradí – každé vrcholové pokrytie príslušného druhu by ale malo byť výstupom metódy `next` práve raz. V prípade, že iterátor už prešiel cez všetky svoje vrcholové pokrytia, by mal mať prípadný ďalší pokus o volanie metódy `next` za následok vyhodenie výnimky typu `NoSuchElementException`.
- Metóda `hasNext` by sa mala správať konzistentne s metódou `next`; ďalšie metódy iterátora nie je potrebné implementovať.
- Iterátory *neimplementujte* prehľadaním všetkých vrcholových pokrytí pomocou prehľadávania s návratom hneď v konštruktore iterátora alebo v metóde, ktorá ho vytvára. Po každom volaní metódy `next` naopak spustite iba malú časť tohto prehľadávania s návratom, ktoré sa preruší po nájdení nasledujúceho vrcholového pokrytia príslušného druhu a obnoví sa po ďalšom volaní metódy `next`.

  Pomocou vašich iterátorov by špeciálne malo byť možné v krátkom čase prejsť cez prvých niekoľko vrcholových niektorých väčších grafov (napr. okolo 100 vrcholov) aj v prípade, že celkový počet všetkých vrcholových pokrytí daného druhu je pre takýto graf príliš veľký (ale existuje dostatočne veľa pokrytí na to, aby hľadanie prvých niekoľkých netrvalo príliš dlho).

Na testovač odovzdávajte súbor `VertexCovers.java` obsahujúci zdrojový kód vašej triedy balíka `graphs`. Hodnotiť sa bude aj programátorský štýl, ako aj dodržiavanie konvencií jazyka Java a elementárnych zásad objektovo orientovaného programovania.

Príklad. Uvažujme neorientovaný graf na nasledujúcom obrázku a predpokladajme, že sme preň vytvorili inštanciu triedy `VertexCovers`.



Metóda `isVertexCover` tejto inštancie potom vráti `true` napríklad pre množiny vrcholov `[0,4]`, `[0,5]`, `[1,2,3,4]` a `[0,1,2,3,4,5]`. Booleovská hodnota `false` je naopak jej výstupom napríklad pre množiny `[0]`, `[4]`, `[0,3]`, alebo `[2,3,4,5]`.
Výstupom metódy `minimumVertexCoverSize` je v tomto prípade číslo 2, pretože najmenšími vrcholovými pokrytiami uvedeného grafu sú množiny `[0,4]` a `[0,5]`.
Metóda `next` iterátora získaného pomocou metódy `minimumVertexCoversIterator` v ľubovoľnom poradí vráti tieto dve najmenšie vrcholové pokrytia uvažovaného grafu; obe sú ale výstupom metódy `next` práve raz a tretie volanie metódy `next` povedie k vyhodeniu výnimky typu `NoSuchElementException`.
Iterátor získaný pomocou metódy `fixedSizeVertexCoversIterator` s argumentom 1 nikdy nevráti žiadne vrcholové pokrytie, pretože vrcholové pokrytie veľkosti 1 v nami uvažovanom grafe neexistuje. Metóda `next` tohto iterátora teda bude od začiatku vyhadzovať výnimku typu `NoSuchElementException` a jeho metóda `hasNext` bude vracať `false`.
V prípade, že je argumentom metódy `fixedSizeVertexCoversIterator` číslo 3, vráti táto metóda na výstupe iterátor, ktorý bude v ľubovoľnom poradí prechádzať cez všetkých sedem vrcholových pokrytí uvažovaného grafu veľkosti 3 – čiže cez pokrytia `[0,1,4]`, `[0,2,4]`, `[0,3,4]`, `[0,4,5]`, `[0,1,5]`, `[0,2,5]` a `[0,3,5]`.
Výstupom metódy `allVertexCoversIterator` je napokon iterátor, ktorý postupne prejde cez všetkých 26 vrcholových pokrytí uvažovaného grafu.