Test č. 3, úloha č. 1

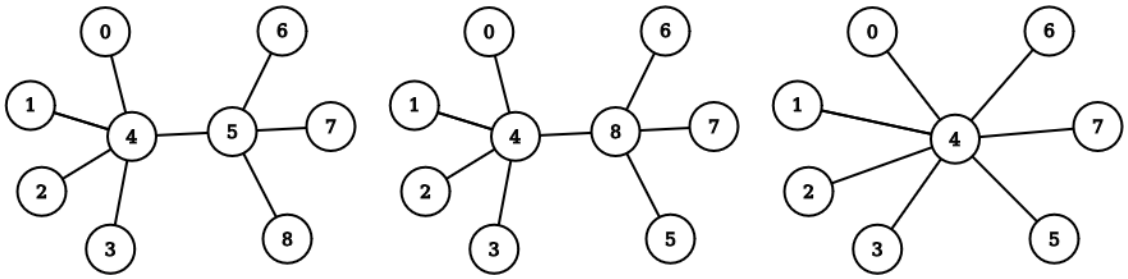
V priloženej kostre nájdete známu implementáciu balíka `graphs` z prednášky. Vašou úlohou bude implementovať funkciu `public static UndirectedGraph contractEdge(UndirectedGraph graph, int from, int to)` v triede `GraphMorph`.
Graf G' vznikne z grafu G kontrakciou hrany $uv \in E(G)$ nasledovne.

- $V(G') = V(G) \setminus \{v\}$,
- $E(G') = (E(G) \setminus \{vx | vx \in E(G)\}) \cup \{ux | vx \in E(G)\}$.

Inými slovami, z grafu G, ktorý obsahuje hranu uv odstránime vrchol v, spolu so všetkými hranami, ktoré sú s nim v G susedné. Následne, všetky vrcholy, ktoré boli susedné s vrcholom v v grafe G budú susedné s vrcholom u v grafe G'. Ak existuje vrchol x, ktorý bol v G susedný aj s vrcholom u aj v, hranu xu znova nepridávame, pretože by išlo o násobnú hranu. Rovnako nepridávame ani slučku (hranu uu) do vrchola u v grafe G'. Pre ilustráciu pozrite obrázok nižšie.

Aby ste vedeli implementovať funkciu `contractEdge`, budete potrebovať implementovať funkciu `public static UndirectedGraph renameVertices(UndirectedGraph graph, List<Integer> permutation)`, ktorá slúži na premenovanie vrcholov grafe. Zoznam `permutation` určuje mená vrcholov v novom grafe. Meno vrchola `i` v novom grafe zodpovedá `permutation.get(i)`. Malo by sa naozaj jednať o permutáciu, teda bijekciu $\{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$. To znamená, že `permuation` by mal byť dĺžky n a obsahovať každé číslo z množiny $\{0, \ldots, n-1\}$ práve raz. Pokiaľ sa o permutáciu nejedná, musí funckia vyhodiť výnimku `IllegalArgumentException` so správou *"Incorrect mapping"*.

Funkcia `contractEdge` dostane v argumente graf a koncové vrcholy hrany, ktorá má byť kontrahovaná. Pokiaľ sa taká hrana v grafe nenachádza, musí funkcia vyhodit výnimku `IllegalArgumentException` so správou *"No such edge"*. Pri kontrahovaní hrany bude **vždy** odstránený vrchol `to` (teda posledný parameter funckie). Predtým ako môže byť vrchol odstránený je potrebné premenovať vrcholy grafu tak, aby odstraňovaný vrchol niesol meno $n-1$ a vrchol $n-1$ bude teda premenovaný na `to`, na čo budete patrične potrebovať funkciu `renameVertices`. Z grafu, v ktorom je odstraňovaný vrchol posledný následne funkcia vyrobí nový graf s kontrahovanou hranou podľa definície.
Na testovač odovzdajte iba súbor `GraphMorph.java`. V priloženej kostre máte aj triedu `Testovac`, ktorá obsahuje funkciu na načítanie grafu. Využite ju na otestovanie svojho riešenia. Môžete využiť príklady nižšie.



G_0 G_1 G_2

Príklady volaní:
Nasledujúci príkaz premenuje v grafe G_0 vrchol 5 na 8 a naopak, výsledkom čoho je graf G_1.

```
UndirectedGraph g_1 = GraphMorph.renameVertices(g_0, List.of(0,1,2,3,4,8,6,7,5));
```

Naledujúci príkaz kontrahuje hranu $(4, 5)$ v grafe G_0, výsledkom čoho je graf G_2. Všimnite si, že v skutočnosti funkcia `contractEdge` najskôr premenuje vrcholy 5 a 8 (graf G_1) a až následne kontrahuje hranu $(4, 8)$, pričom odstráni vrchol 8.

```
UndirectedGraph g_2 = GraphMorph.contractEdge(g_0, 4, 5);
```