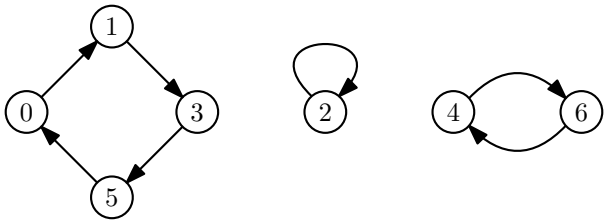
Cvičenia č. 9, úloha č. 3

Budeme hovoriť, že orientovaný graf *reprezentuje permutáciu*, ak je disjunktným zjednotením orientovaných cyklov. To znamená, že množinu jeho vrcholov možno rozložiť na niekoľko disjunktných podmnožín tak, že hrany grafu medzi vrcholmi každej z týchto podmnožín tvoria práve jeden orientovaný cyklus a medzi vrcholmi dvoch rôznych podmnožín nevedú žiadne hrany. Ľahko pritom vidieť, že graf reprezentuje permutáciu práve vtedy, keď má každý jeho vrchol práve jedného následníka a práve jedného predchodcu; to znamená, že z každého aj do každého vrcholu grafu vedie práve jedna orientovaná hrana.[1] To ďalej nastane práve vtedy, keď je každý vrchol grafu súčasťou nejakého orientovaného cyklu a zároveň má práve jedného následníka.
Graf o n vrcholoch $0, 1, \ldots, n-1$ spĺňajúci uvedené ekvivalentné podmienky budeme považovať za reprezentáciu permutácie $\pi\colon \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$ takej, že pre všetky $v \in \{0, 1, \ldots, n-1\}$ je $\pi(v)$ dané ako jediný následník vrcholu v v danom grafe.
Príklad orientovaného grafu reprezentujúceho permutáciu π takú, že $\pi(0) = 1$, $\pi(1) = 3$, $\pi(2) = 2$, $\pi(3) = 5$, $\pi(4) = 6$, $\pi(5) = 0$ a $\pi(6) = 4$:



Priložený ZIP archív obsahuje balík `graphs` a v ňom:

- Triedy pre grafy z prednášky.
- Kostru triedy `Permutation`, ktorú bude vašou úlohou naprogramovať.
- Triedu `PermuteList` realizujúcu vstup a výstup. Táto trieda sa bude spúšťať aj na testovači.

V triede `Permutation` implementujte:

- Konštruktor, ktorý ako jediný parameter vezme orientovaný graf `g` typu `DirectedGraph` a zistí, či ide o graf reprezentujúci permutáciu. Ak nie, vyhodí výnimku typu `IllegalArgumentException`. V opačnom prípade do premenných inštancie triedy uloží informácie o reprezentovanej permutácii tak, aby mohli byť použité metódou `apply` opísanou nižšie.
- Generickú metódu `apply` s jedným typovým parametrom `E` a jedným bežným parametrom `list` typu `List<E>`. Táto metóda aplikuje permutáciu reprezentovanú grafom `g` z argumentu konštruktora na zoznam `list`. Presnejšie: ak je veľkosť zoznamu `list` rôzna od počtu vrcholov grafu `g`, vyhodí sa výnimka typu `IllegalArgumentException`. V opačnom prípade sa vezme permutácia π, ktorá grafu `g` zodpovedá podľa definície z úvodu tohto zadania a zoznam `list` sa transformuje tak, že pre $i = 0, 1, \ldots, n-1$ bude prvok pôvodne na i-tej pozícii zoznamu umiestnený na jeho $\pi(i)$-tu pozíciu.

  Táto metóda iba modifikuje vstupný zoznam `list`. Jej návratový typ má byť `void`.

V rámci triedy `Permutation` môžete implementovať aj ďalšie pomocné metódy. Hodnotiť sa bude aj programátorský štýl a dodržiavanie konvencií jazyka Java.

Metóda `main` triedy `PermuteList` načíta zo vstupu graf, pre ktorý vytvorí inštanciu triedy `Permutation`. Následne načíta zoznam celých čísel alebo reťazcov a pokúsi sa naň aplikovať vytvorenú permutáciu. Formát vstupu a výstupu je ako v príklade nižšie.

Na testovač odovzdávajte iba súbor `Permutation.java` obsahujúci zdrojový kód triedy `Permutation` v balíku `graphs`.

[1] Špeciálnym prípadom takéhoto vrcholu je vrchol so slučkou, do ktorého ani z ktorého nevedie žiadna ďalšia orientovaná hrana.

Príklad vstupu:

```
7 7

0 2
2 3
3 0
1 1
4 5
5 4
6 6

String
7
a b c d e f g
```

Príklad výstupu:

```
[d, b, a, c, f, e, g]
```