Náhradný test, úloha č. 3

Stiahnite si kostru obsahujúcu známu implementáciu balíka `graphs` z prednášky. Vašou úlohou je naprogramovať triedu `GraphInducedComparator` implementujúcu rozhranie `Comparator<Integer>`. Potrebujete implementovať metódu `compare` a nasledujúci konštruktor.

```
GraphInducedComparator(DirectedGraph g, ToIntFunction<Integer> vertexToWeight)
```

Pôjde teda o komparátor prvkov typu `Integer`, ktorý v konštruktore dostane graf `g` a funkciu (objekt implementujúci funkcionálne rozhranie `ToIntFunction<Integer>` z balíka `java.util.function`) ktorá pre *vrchol* grafu vráti jeho *váhu*. Prvky bude porovnávať nasledujúcim spôsobom. Ak jeden alebo oba porovnávané prvky nie sú indexom žiadneho vrchola v grafe, tak ich porovná pomocou štandardného porovnania na celých číslach. Ak sú oba prvky indexmi vrcholov v grafe, porovná ich porovnaním *váh* príslušných vrcholov.

Ďalej naprogramujte statickú vnorenú triedu `ReachableVertexCount` implementujúcu funkcionálne rozhranie `ToIntFunction<Integer>`. Potrebujete implementovať metódu `applyAsInt` a nasledujúci konštruktor.

```
ReachableVertexCount(DirectedGraph g)
```

Pôjde teda o triedu predstavujúcu funkciu, ktorá vrcholom grafu, ktorý dostane v konštruktore, priradí ich váhu. V tomto prípade ako `váhu` vrchola v zoberieme počet vrcholov u takých, že v grafe existuje orientovaná cesta z v do u. Za takúto cestu považujeme aj cestu dĺžky 0 z v do v. Čiže ak z vrchola v neodchádza žiadna hrana, vrátime hodnotu 1. Volanie metódy `applyAsInt` predstavuje volanie funkcie, ktorú táto trieda reprezentuje. Keďže ide o funkciu na celých číslach mala by sa korektne správať (nesmie hádzať žiadne výnimky) aj keď na vstupe nedostane index vrchola grafu ale iné celé číslo. V takomto prípade nechávame dodefinovanie tejto funkcie na vás, teda môžete vrátiť akékoľvek celé číslo.

Pri testovaní bude aj vaša funkcionálna trieda `ReachableVertexCount` použitá v konštruktore vašeho komparátora `GraphInducedComparator`. Môžete prepokladať, že graf v konštruktore oboch tried bude rovnaký, teda funkcia, ktorá priraďuje vrcholom ich váhu, pracuje s rovnakých grafom ako komparátor, ktorý ich má poďla tejto váhy porovnávať. Dodržujte zásady objektovo-orientovaného programovania a konvencie jazyka *java*. Na testovač odovzdávajte iba súbor `GraphInducedComparator.java` obsahujúci vami doplnený kód. Ostatné triedy balíka `graphs` budú k tejto triede na testovači priložené. Priložený archív tiež obsahuje triedu `Testovac`, ktorá obsahuje statickú metódu na načítanie grafu a prázdnu metódou `main`. Môžete ju využiť na otestovanie vášho riešenia na nasledujúcom príklade.

Príklad:

```
ToIntFunction<Integer> weight = new GraphInducedComparator
                                        .ReachableVertexCount(tree);
System.out.println( weight.applyAsInt(0) );     // vypise 13
System.out.println( weight.applyAsInt(7) );     // vypise 1

GraphInducedComparator comp = new GraphInducedComparator(tree, weight);
System.out.println( comp.compare(8, 1) );       // vypise cele cislo <0
System.out.println( comp.compare(0, 4) );       // vypise cele cislo >0
System.out.println( comp.compare(4, 2) );       // vypise 0

List<Integer> list = new ArrayList<>();
for (int i = 0; i < tree.getVertexCount(); ++i) list.add(i);
Collections.sort(list, comp);
System.out.println(list);   // vypise [3, 5, 7, 9, 10, 11, 12, 6, 8, 2, 4, 1, 0]
```

Vstup pre graf `tree`:

```
13 12
0 1
0 2
1 3
1 4
2 5
2 6
4 7
4 8
6 9
6 10
8 11
8 12
```