Druhá domáca úloha

Ospravedlňujeme sa za oneskorené zverejnenie druhej domácej úlohy. Aby zostalo v platnosti, že na riešenie máte 2 týždne, termín odovzdania sa posúva na 13.5. 23:59.

V priloženej kostre nájdete známu implementáciu balíka `graphs` z prednášky spolu s balíkom `gui`, ktorý bude obsahovať všetky triedy týkajúce sa JavaFX časti tohto zadania. Vašou úlohou bude implementovať interaktívnu aplikáciu **Graph GUI** na vizualizáciu neorientovaných grafov. Predtým, než začneme s implementáciou samotnej aplikácie, budeme potrebovať pomocnú triedu `GraphUtil`.

GraphUtil

```
public class GraphUtil {
  public static void renameVertices(UndirectedGraph g, List<Integer> perm) {..}
  public static UndirectedGraph addVertex(UndirectedGraph g) {..}
  public static UndirectedGraph addEdge(UndirectedGraph g, int v1, int v2) {..}
  public static UndirectedGraph removeVertex(UndirectedGraph g, int id) {..}
  public static String encodeGraph6(UndirectedGraph g) {..}
  public static UndirectedGraph decodeGraph6(final String code) {..}
}
```

Všetky funkcie, ktoré majú `UndirectedGraph` ako návratový typ, vytvoria novú inštanciu tejto triedy. To znamená, že ak zadanie hovorí napríklad "funkcia pridá *niečo* grafu", znamená to, že vytvorí novú inštanciu tejto triedy líšiacu sa od grafu v parametri pridaním "*niečoho*".

- `renameVertices` - túto funkciu ste mali za úlohu implementovať už na teste číslo 3. Tí, ktorí ju úspešne vyriešili smú použiť svoje riešenie z testu. Pre úplnosť, opíšeme správanie tejto metódy znova. Funkcia 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"*.
- `addVertex` - pridá do grafu nový vrchol.
- `addEdge` - pridá do grafu neorientovanú hranu z $v1$ do $v2$.
- `removeVertex` - odstráni z grafu vrchol `id` spolu so všetkými hranami, ktoré s ním boli susedné. Všetky vrcholy x také, že $x > id$ budú mať po odstránení číslo $x-1$, tak aby bolo zachované, že graf obsahuje vrcholy 0 až $n-1$. Budete potrebovať využiť funkciu `renameVertices`. Pokiaľ je id odstraňovaného vrchola vačšie rovné ako počet vrcholov v grafe, funkcia vyhodí výnimku `IllegalArgumentException("Non-existe vertex")`.
- `encodeGraph6` - vráti String reprezentujúci graf v Graph6 formáte.
- `decodeGraph6` - vráti graf zodpovedajúci grafu zapísanom v Graph6 formáte.

Túto časť úlohy vám otestuje aj testovač. Všetky funkcie vracajúce `UndirectedGraph` vracajú ľubovoľnú implementáciu tohto rozhrania (je jedno či `AdjacencyMatrixUndirectedGraph` alebo `AdjacencyListsUndirectedGraph`).

Graph6 formát

O Graph6 formáte sa možete dozvedieť všetko potrebné na tomto odkaze. Pre jednoduchosť vysvetlíme princíp aj tu. *Graph6* je formát na zápis grafov, kde jedno slovo (postupnosť tlačiteľných znakov) reprezentuje jeden graf. Toto slovo sa skladá z dvoch podslov $N(G)$ a $E(G)$, kde $N(G)$ určuje počet vrcholov grafu a $E(G)$ určuje

hrany grafu. $N(G)$ aj $E(G)$ sú postupnosti celých čísel, kde každé číslo je v rozsahu od 63 po 126, čo je (nie náhodou) taký rozsah, že všetky ASCII znaky s číslom z tohto rozsahu sú tlačiteľné znaky. Postupnosť 63 109 100 teda vieme reprezentovať ako reťazec znakov "?md". Majme ľubovoľné číslo x, napríklad 12345, ktorého binárny zápis je 000011 000000 111001. Funkcia $R(x)$ jednoducho "naseká" binárnu reprezentáciu na úseky dĺžky 6, kde každý úsek reprezentuje nejaké číslo z rozsahu $0-63$. Po pričítaní 63 ku každému úseku dostávame číslo z rozsahu $63-126$. Poďme si teraz zadefinovať ako vyzerajú postupnosti $N(G)$ a $E(G)$ pre konkrétny graf G.

$$N(G) = \begin{cases} |V(G)| + 63, & \text{ak } 0 \leq |V(G)| \leq 62, \\ 126\ R(x), & \text{ak } 63 \leq x = |V(G)| \leq 258047, \text{ kde } x \text{ je 18 bitový zápis } |V(G)|, \\ 126\ 126\ R(x), & \text{ak } 258048 \leq x = |V(G)| \leq 68719476735, \text{ kde } x \text{ je 36 bitový zápis } |V(G)|. \end{cases}$$

Napríklad:

$$\begin{aligned} N(30) &= 93, \\ N(12345) &= 126\ R(000011\ 000000\ 111001) = 126\ 66\ 63\ 120, \\ N(460175067) &= 126\ 126\ R(000000\ 011011\ 011011\ 011011\ 011011\ 011011) = 126\ 126\ 63\ 90\ 90\ 90\ 90\ 90. \end{aligned}$$

Postačuje implementovať $N(G)$ iba pre prvé dva prípady, teda pre grafy menšie ako 258048 vrcholov. Pre väčší graf vyhodí funkcia `encodeGraph6` výnimku `IllegalArgumentException("Graph too big")`. Naopak, ak Graph6 reprezentácia grafu bude začínať 126 126, vyhodí metóda `decodeGraph6` výnimku `IllegalArgumentException("Incorrect input string")`.
Pre defininíciu $E(G)$ si usporiadajme dvojice vrcholov grafu v poradí $P = (1,0),(2,0),(2,1),(3,0),(3,1),\ldots,(n,n-1)$, kde $n = |V(G)|$, čím dostaneme postupnosť dľžky $n(n-1)/2$. Všimnite si, že takéto usporiadanie zodpovedá čítaniu trojuholníka pod diagonálou matice susednosti po riadkoch. Skonštruujme teraz binárnu postupnosť B nasledovne, kde B_i je je i-ty člen.

$$B_i = \begin{cases} 1, \text{ ak existuje v } G \text{ hrana medzi vrcholmi v } P_i, \\ 0, inak. \end{cases}$$

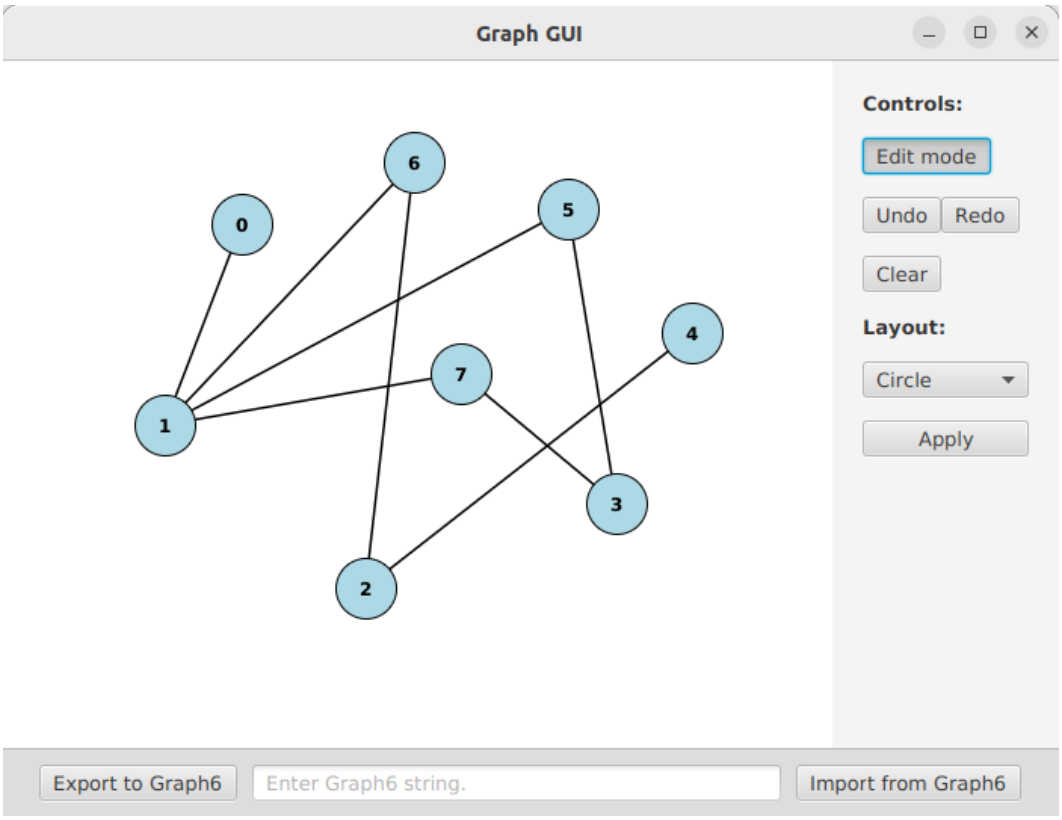
Na záver vytvorme postupnosť B_i^+ tak, že na koniec B_i doplníme toľko 0, aby 6 delilo $|B_i^+|$. Teraz už len stačí zadefinovať $E(G)$ ako $R(B_i^+)$.
Teraz môžeme Graph6 zápis grafu definovať ako $graph6(G) = N(G)E(G)$.
Napríklad, vezmime graf $K_{3,3}$, ktorý má 9 hrán $(0,3),(0,4),(0,5),(1,3),(1,4),(1,5),(2,3),(2,4),(2,5)$. Potom

$$\begin{aligned} N(G) &= 69, \\ E(G) &= R(000111\ 111011\ 100000) = 70\ 122\ 95, \\ graph6(G) &= N(G)\ E(G) = 69\ 70\ 122\ 95 = \text{"EFz\_"}. \end{aligned}$$

Teda práve String "EFz_" je to, čo má funkcia `encodeGraph6` vrátiť pre túto inštanciu grafu $K_{3,3}$.
Ak funkcia `decodeGraph6` zistí, že vstupný string je príliš dlhý alebo príliš krátky, opäť vyhodí výnimku `IllegalArgumentException("Incorrect input string")`. Pozor. Pre inú reprezentáciu toho istého grafu (líšiaciu sa v menách vrcholov) môže Graph6 String vyzerať inak. Stále ale bude reprezentovať ten istý graf.

Vzhľad a design grafickej aplikácie

Okno grafickej aplikácie by sa malo skladať z troch základných oblastí, a to plátna, bočného panelu a spodného panelu. Spodný panel by mal mať šírku od ľavého okraja po pravý a výšku dostatočnú na to, aby sa tam zmestil všetok potrebný obsah. Priestor nad spodným panelom bude tvorený plátnom na ľavej strane a bočným panelom na pravej strane. Šírka bočného panelu by mala byť fixná a dostatočná na zobrazenie celého obsahu tak, aby sa nič neprekrývalo. Zvyšok okna bude vyplnený plátnom, ktorého výška a šírka sa bude dynamicky prispôsobovať veľkosti okna aplikácie tak, aby nijak nezasahovala do spodného ani bočného panela. Na farbách pozadia, textu a tlačidiel nezáleží, ale určite nechcete spôsobiť, aby z vašeho riešenia boleli cvičiaceho oči. Môže to spôsobiť bolesť vašeho bodového ohodnotenia. Príklad vzhľadu a rozloženia si môžete pozrieť na obrázku nižšie.



Plátno

Plátno má slúžiť na kreslenie grafu používateľom. Ako bolo už spomenuté vyššie, jeho rozmery sa majú nastavovať dynamicky podľa šírky okna aplikácie. Správanie plátna možno rozdeliť do dvoch módov.

- **Edit mode** - po kliknutí do plátna na miesto, ktoré sa neprekrýva so žiadnym vrcholom sa na dané miesto pridá nový vrchol s indexom n, kde n je počet vrcholov ktoré už graf obsahuje (číslujeme od 0). Po kliknutí na už existujúci vrchol sa tento vrchol označí (zmení farbu). Následné kliknutie na ľubovoľný iný vrchol spôsobí, že sa medzi tieto dva vrcholy pridá hrana. Ak už medzi nimi hrana bola, nestane sa nič. Každopádne, na konci zostanú všetky vrcholy neoznačené. Ak máme označený vrchol, opätovné klknutie na daný vrchol odznačí daný vrchol.

  Potiahnutie vrchola po plátne (so stlačeným ľavým tlačidlom myši) spôsobí posunutie vrchola a dynamické prekreslovanie všetkých s ním susedných hrán.

- **Delete mode** - po kliknutí na vrchol x sa daný vrchol odstráni spolu so všetkými s ním susednými hranami. Všetkým vrcholom s indexom vačším ako x sa po odstránení index o 1 zníži.

Každá z operácií, ktorá modifikuje štruktúru grafu (pridaj vrchol, pridaj hranu, odstráň vrchol) zodpovedá príslušnej metóde v `GraphUtil`. Aplikácia **Graph GUI** si bude teda v pozadí pamätať inštanciu `UndirectedGraph`-u. Každá z modifikujúcich operácii spôsobí, že si aplikácia bude pamätať novú inštanciu. Triedy v balíku `graphs` (okrem `GraphUtil`) modifikovať nesmiete. Všetky informácie o polohe vrcholov, o tom ktorý je označený a podobne si bude aplikácia pamätať lokálne.

Bočný panel

Na bočnom paneli by sa mali nachádzať prvky ovládania aplikácie (V príklade v sekcií Controls). Mal by obsahovať nasledovné prvky.

- **Prepínač módov**. Tlačidlo (alebo iný ovládač), ktoré nám dovoľuje prepínať sa z Edit módu do Delete módu a naopak. To, o aký konkrétny ovládač sa jedná nie je dôležité, podstatné je, aby sa po stlačení niekde zmenil text (informácia) o tom, v akom móde sa aktuálne nachádzame. V našej ukážke sa po kliknutí na tlačidlo s nápisom *Edit mode* zmení jeho text na *Delete mode*.

- **Undo**/**Redo** tlačidlá. Bola trochu lož, keď sme povedali, že aplikácia si bude pamätať iba aktuálnu inštanciu grafu. Mala by si pamätať históriu úprav (modifikujúce operácie, ale aj posúvanie vrcholov po plátne sa ráta ako úprava) tak, aby sme sa po stlačení **Undo** vrátili na predchádzajúcu verziu, respektíve po stlačení **Redo** na verziu pred stlačením **Undo**. Z technických príčin nie je možné pamätať si nekonečnú históriu, povedzme preto, že aplikácia si bude musieť pamätať iba posledných 25 verzií.
- Tlačidlo **Clear**. Slúži na vymazanie celej plochy.

Pod ovládacími prvkami by sa mali nachádzať prvky na zmenu rozloženia (layout-u) vrcholov na plátne (v našom príklade sekcia Layout). Mali by sa tam nachádzať dva konkrétne prvky.

- **Voľba rozloženia**. Ľubovoľný ovládač dovolujúci výber z viacerých textových možností.
- **Tlačidlo aplikácie rozloženia**. Kliknutie na toto tlačidlo spôsobí, že sa graf na plátne prekreslí podľa zvoleného rozloženia.

Rozloženia

Pre všetky rozloženia platí, že pozície všetkých vrcholov musia byť vrámci akutálnych rozmerov plátna.

- **Random**. Po aplikovaní tohto rozloženia sa vrcholy presunú na náhodné pozície na plátne.
- **Circle**. Vrcholy sa usporiadajú na kružnicu so stredom v strede plátna a maximálnym polomerom takým, aby sa všetky vrcholy zmestili na plátno. Poradie vrcholov na kružnici je dané ich indexom. Vrcholy by sa na kružnici mali nachádzať rovnomerne.
- **Grid**. Vrcholy sa na plátne usporiadajú do mriežky $n \times m$ tak, aby vrcholy v rohu mriežky boli čo najbližšie okrajom plátna. Hodnoty n a m závisia od počtu vrcholov grafu a rozmerov plátna tak, aby $n \times m$ čo najbližšie zodpovedalo pomeru strán plátna, pričom nemusí platiť $nm = |V(G)|$. Posledné pozície na mriežke v poslednom riadku nemusia byť obsadené žiadnym vrcholom, pričom by posledný riadok nemal zostať prázdny (mal by obsahovať aspoň jeden vrchol).
- **BFS (Bonus 1b)**. Vrcholy sa vykreslia na plátno podľa vzdialeností od označeného vrchola. Teraz myslíme vzdialenosť medzi nimi, teda počet hrán na najkratšej ceste medzi nimi. Nech x je označený vrchol a y je vrchol v najväčšej vzdialenosti od x v grafe G. Nech je táto vzdialenosť d. Rozloženie bude pozostávať z $d+1$ riadkov, kde na nultom riadku bude vrchol x, na prvom vrcholy vo vzdialenosti 1 od x a tak ďalej až po riadok d. Vrcholy v jednom riadku by mali byť rozložené po celej šírke riadka rovnomerne (v rovnakých vzdialenostiach). Vrchol x by teda mal byť v strede nultého riadka. Bystrejší z vás už určite rozmýšľajú nad tromi vecami. Nedefinovali sme šírku riadka, nepovedali sme, čo sa stane s vrcholmi, ktoré niesú v rovnakom komponente súvislosti ako x, a čo sa stane ak dáme aplikovať toto rozloženie bez toho, aby sme mali označený nejaký vrchol. Nech X je počet vrcholov v komponente súvislosti, ktorý obsahuje x a Z je počet ostatných vrcholov v grafe. Plátno sa teda rozdelí na dve časti (ľavú a pravú) v pomere X ku Z. Do ľavej časti plátna sa vykreslí komponent obsahujúci X podľa pokynov vyššie. Šírka riadka teda zodpovedá šírke ľavej časti plátna. Na pravú stranu sa vykreslí zvyšok grafu podľa rozloženia **Random**. Pokiaľ nebol označený žiaden vrchol, zoberie sa za x vrchol s indexom 0.

Spodný panel

Na spodnom paneli by sa mali nachádzať dve tlačidlá a jedno textové pole. Textové pole slúži na prácu s Graph6 zápisom grafu. Tlačidlo **Export to Graph6** nastaví text v textovom poli na Graph6 string zodpovedajúci grafu nakreslenom na plátne. Tlačidlo **Import from Graph6** vykreslí na plátno graf reprezentovaný stringom v Graph6 formáte. Rozloženie pre nakreslenie bude **Random**.

Záverečné pokyny a pomôcky

Aplikácia **Graph GUI** je zjednodušenou verziou webovej aplikácie House of Graphs (link). Pohrajte sa s touto webovou aplikáciou, uľahčí vám pochopenie zadania a pomôže vyjasniť nezrozumiteľnosti. Taktiež tam viete otestovať to, či váš program počíta Graph6 stringy správne.
Na testovač odovzdajte `ZIP` balíkov `graphs` a `gui`. Pozor, nezipujte nadadresár obsahujúci tieto dva balíky. Označte oba priečinky (zodpovedajúce balíkom) súčasne a následne ich zazipujte.