Zadání úlohy 1graf - Grafové knihovny a reprodukovatelné prostředí

Cíl úlohy

Hlavní cíl: Osvojit si práci s grafovou knihovnou a vytvořit reprodukovatelné vývojové prostředí pro řešení následujících úloh.

Co se naučíte:

- Vytvořit spustitelný kód s dokumentací a správou závislostí
- Používat grafovou knihovnu pro konstrukci, analýzu a vizualizaci grafů
- Pracovat s atributy vrcholů a hran
- Aplikovat základní grafové algoritmy z knihovny

Zadání

Výstupní produkt

Odevzdejte balíček obsahující:

1. **Dokumentaci** (README.md nebo ekvivalent)
   - Návod k instalaci a spuštění (max 2 kroky)
   - Popis implementovaných funkcí
   - Identifikace autorů a použité knihovny
2. **Konfigurační soubor** (pokud je potřeba)
   - pyproject.toml, requirements.txt, package.json, nebo ekvivalent
   - Specifikace všech závislostí
3. **Spustitelný kód** (Jupyter notebook nebo skripty)
   - Implementace požadovaných funkcí
   - 2 ze 3 předem definovaných grafů
   - Demonstrace všech funkcí z checklistu

Konkrétní požadavky

A. Předem definované grafy (implementujte 2 ze 3):

1. **“Smajlík”**:
   - Kruh pro obličej (6 vrcholů)
   - 2 vrcholy pro oči
   - 3 vrcholy pro ústa
2. **“Domeček”**:
   - 4 vrcholy čtverec (základna)
   - 1 vrchol špička střechy
   - Volitelně: vrchol pro komín
3. **“Hvězda”**:
   - 1 centrální vrchol
   - 6 vrcholů v kruhu kolem něj

- Hrany z centra ke všem okolním vrcholům

Pro každý graf definujte pozice vrcholů (slovník: vrchol → (x,y)) a použijte různé barvy nebo styly pro lepší vizualizaci.

B. Checklist funkcí k implementaci:

- [ ] **create_graph()** - vytvoří prázdný graf a přidá vrcholy/hrany ze seznamů
- [ ] **examine_graph(G)** - vrátí počet vrcholů, hran, seznam stupňů všech vrcholů
- [ ] **visualize_graph(G, positions)** - vykreslí graf s předem definovanými pozicemi vrcholů
- [ ] **get_adjacency_matrix(G)** - vrátí a vizualizuje matici sousednosti
- [ ] **filter_by_edge_attribute(G, attribute, value)** - vrátí podgraf obsahující jen hrany s danou hodnotou atributu
- [ ] **graph_statistics(G)** - vrátí statistiky: počet komponent, min/max/avg stupeň vrcholů, min/max/avg váha hran
- [ ] **find_sources_sinks(G)** - pro orientovaný graf vrátí seznamy source vrcholů (in_degree=0) a sink vrcholů (out_degree=0)

Hodnocení (celkem 2 body)

Základní požadavky (0.6 bodu)

Kritérium	Body	Popis
Reprodukovatelnost	0.3	Kód lze spustit podle návodu (max 2 kroky)
Dokumentace	0.1	Jasný návod, popis funkcí, identifikace autorů
Vytvoření grafů	0.2	2 předem definované grafy s definovanými pozicemi

Implementace funkcí (1.4 bodu)

Funkce	Body	Popis
create_graph()	0.1	Vytvoření grafu ze seznamů vrcholů a hran
examine_graph()	0.1	Získání základních charakteristik
visualize_graph()	0.3	Vizualizace s pozicemi, barvami a styly
get_adjacency_matrix()	0.2	Matice sousednosti + její vizualizace
filter_by_edge_attribute()	0.2	Filtrování podle atributů hran
graph_statistics()	0.3	Komplexní statistiky včetně komponent
find_sources_sinks()	0.2	Práce s orientovaným grafem

Použití AI nástrojů

AI nástroje jsou povoleny a dokonce doporučeny! (ChatGPT, Claude, GitHub Copilot, atd.)

Jak správně používat AI:

- ✅ **Použijte AI jako nástroj** - nechte si vysvětlit koncepty, generovat code snippety, debugovat

- ✅ **Konzultujte s AI** - ptejte se na nejasnosti, alternativní řešení, best practices
- ✅ **Nechte si vysvětlit kód** - MUSÍTE rozumět každému řádku, který odevzdáte

Čeho se vyvarovat:

- ❌ Slepě kopírovat kód bez pochopení
- ❌ Odevzdat něco, co neumíte vysvětlit
- ❌ Spoléhat se pouze na AI bez konzultace dokumentace

Důležité: Při odevzdání musíte být schopni vysvětlit, jak váš kód funguje. AI je skvělý pomocník, ale vy jste zodpovědní za pochopení řešení!

Doporučený postup

Doporučená kombinace nástrojů:

```
Python + (uv nebo Poetry) + NetworkX + JupyterLab
```

Příklad struktury projektu:

```
graf-demo/
├── README.md           # Dokumentace a návod
├── pyproject.toml      # Konfigurace a závislosti
└── graf_demo.ipynb     # Jupyter notebook s implementací
```

Vzorový návod k spuštění:

```
# Krok 1: Instalace
uv sync  # nebo: poetry install

# Krok 2: Spuštění
uv run jupyter lab  # nebo: poetry run jupyter lab
```

Tipy a triky

Vytvoření reprodukovatelného prostředí

S nástrojem uv:

```
uv init graf-demo
cd graf-demo
uv add networkx matplotlib jupyterlab numpy
```

S nástrojem Poetry:

```
poetry new graf-demo
cd graf-demo
poetry add networkx matplotlib jupyterlab numpy
```

Užitečné odkazy

- **NetworkX Getting Started**: https://networkx.org/documentation/stable/tutorial.html
- **NetworkX Examples**: https://networkx.org/documentation/stable/auto_examples/index.html
- **uv dokumentace**: https://docs.astral.sh/uv/
- **Poetry dokumentace**: https://python-poetry.org/docs/
- **JupyterLab**: https://jupyterlab.readthedocs.io/
- **Python Graph Libraries Comparison**: https://www.python-graph-gallery.com/

Alternativní řešení

Můžete použít jiný programovací jazyk a knihovnu, pokud splníte všechny požadavky: - **JavaScript/TypeScript**: cytoscape.js, vis.js, d3.js - **R**: igraph - **Julia**: Graphs.jl, LightGraphs.jl - **C++**: Boost Graph Library - **Java**: JGraphT

FAQ

Q: Musím implementovat všechny funkce přesně podle zadání? A: Názvy mohou být jiné, důležitá je funkcionalita. V jiných jazycích použijte jejich konvence.

Q: Co když knihovna už má funkci, která dělá to, co požadujete? A: Skvělé! Použijte ji a ukažte, že víte, jak s knihovnou pracovat.

Q: Mohu použít více knihoven? A: Ano, ale ujistěte se, že jsou všechny v konfiguračním souboru.

Q: Jak mám odevzdat Jupyter notebook? A: Jako spustitelný kód, ujistěte se, že bloky kódy lze spustit shora dolů.

Poznámka: Cílem není perfektní řešení, ale schopnost rychle vytvořit funkční prototyp s využitím existujících knihoven. Tato dovednost vám usnadní řešení následujících úloh. Nebojte se experimentovat a ptát se!