Vysoké učení technické v Brně
Fakulta informačních technologií

ZADÁNÍ NÁHRADNÍHO PROJEKTU Z PŘEDMĚTŮ IAL

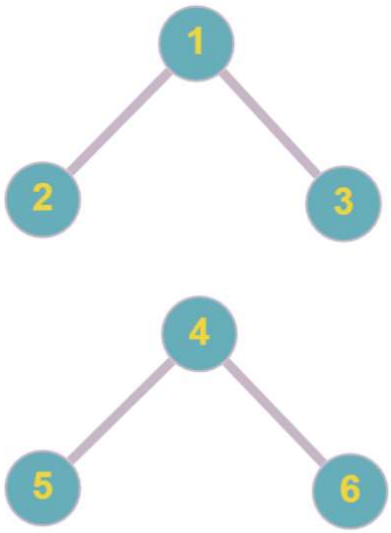
# Zadání č. 3 - Vlastnosti neorientovaných grafů

| | |
|---|---|
| Marek Gergel (vedoucí) | xgergel01 |
| Jindřich Šíma | xsimaj04 |
| Dmytro Sadovskyi | xsadov06 |
| Fišer Tomáš | xfiser16 |

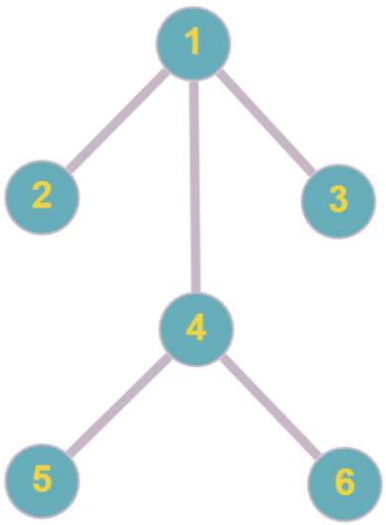
# Obsah

# 1 Popis problematiky

Našim zadáním projektu byla konzolová aplikace, realizovaná pomocí programovacího jazyku C, která na základě vstupních dat uložených v příslušném textovém souboru vyhodnocuje, zda se jedná o data neorientovaného grafu a následně vyhodnocuje jeho základní vlastnosti, kterými jsou počet hran grafu, počet vrcholů, smyček (existuje v grafu cesta, kterou se dostaneme zpět do výchozího vrcholu, hrany se na cestě neopakují), maximální stupeň vrcholu (počet hran končících ve vrcholu), souvislost (mezi každým vrcholem existuje cesta), úplnost (obsahuje všechny možné hrany) a zda je graf lesem nebo stromem. V případě, že se v grafu nenacházejí žádné kružnice, jedná se o les, pokud je navíc souvislí, jedná se o strom. Výsledky analýzy program vypisuje přehledně na výstup do konzole.




*Obrázek 1 Neorientovaný graf - les*




*Obrázek 2 Neorientovaný graf - strom*

# 2 Implementace

Výsledné řešení je reprezentováno několika header soubory ve složce *include* a několika C soubory ve složce *src* na jejich funkcionalitu se v následující kapitole zaměříme.

## 2.1 Soubor main.c

Obsahuje základní funkci main(), která po spuštění programu zavolá funkci pro parsování dat, volání funkce ze souboru parser.c, a následně funkci pro analýzu vlastností grafu.

## 2.2 Soubor graph.c

Zajišťuje inicializaci grafové struktury, správnou alokaci paměti a následně po dokončení práce i správné uvolnění paměťových zdrojů. Dále kontroluje, zda nejsou v zadaných datech chyby v podobě hran s neexistujícími uzly nebo vícenásobné definování jedné hrany. V případě nelezení některého z těchto problému, je definována chybová hláška.

## 2.3 Soubor parser.c

Obstarává parsování vstupných dat, ze kterých je tvořen zadaný grafy. Nejprve parsuje data, pro zadané uzly a následně pro zadané hrany. V případě nalezení nějaké syntaktické chyby v průběhu analýzy dochází k vygenerování a vypsání příslušného chybového hlášení s pozicí chyby na výstup do konzole.

## 2.4 Soubor graph_properties.c

V tomto souboru se nacházejí funkce, které zpracovávají nebo vypisují parametry grafu, které jsme získaly jeho analýzou, patří mezi ně množství hran, uzlů, stupeň uzlu a další, které analyzujeme na základě zadání projektu.

## 2.5 Soubor error.c

Obsahuje chybové funkce, které zajišťují vypsání chybových hlášení na výstup, ukončení provádění programu a zavolání příslušných funkcí pro uklizení paměti.

# 3 Teoretické složitosti

V této kapitole nalezneme časové složitosti jednotlivých funkcí, pro zjišťování jednotlivých vlastností grafu.

- |V| = počet vrcholů
- |E| = počet hran

  - Počet vrcholů: **Θ(1)**
  - Počet hran: **Θ(|V|+|E|)**
  - Počet cyklů: **Θ(|V|+|E|)**
  - Maximální stupeň vrcholu: **Θ(|V|)**
  - Graf je souvislý: **Θ(|V|+|E|)**
  - Graf je úplný: **Θ(|V|)**
  - Graf je strom: **Θ(|V|+|E|)**
  - Graf je les: **Θ(|V|+|E|)**

# 4 Spuštění programu a formát vstupních dat

Postup zkompilování projektu a následné spuštění testovacích dat uložených v podsložce projektu s názvem *testData*.

**Postup:**

1. Stažený zip file, ve kterém je řešení projektu, rozbalíme na našem zařízení
2. Následně v rozbalené složce otevřeme terminál
3. Příkazem *make* do terminálu projekt zkompilujeme
4. Nyní můžeme začít s analýzou vstupních dat ze souboru
5. Formát vstupu: *./graph_properties < cesta_k_souboru_s_daty*
   - ➔ Například: *./graph_properties < testData/test1*
6. Příkazem *make clean* smažeme zkompilovaný projekt

**Formát vstupních dat:**

- **{**vrcholy**} {(**hrana**),(**hrana**),…}**
- Příklad: **{**a,b,c**} {(**a,b**),(**a,c**),(**c,b**)}**

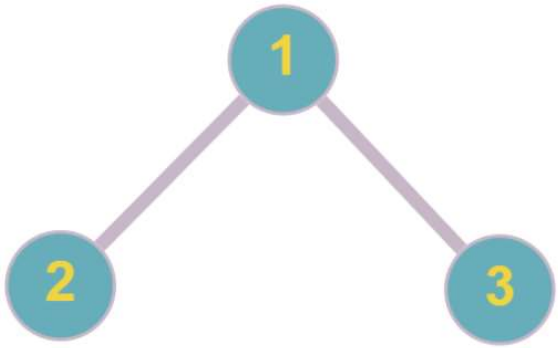
# 5 Testování

V této kapitole si demonstrujeme funkčnost našeho řešení pro různě složité řešení a porovnáme, zda hodnoty vypsané naším programem se shodují s očekávanými výsledky. V podsložce projektu *testData* se nacházejí i další testy na testování jednotlivých vlastností mimo testy demonstrované níže v této kapitole.

## 5.1 1. Experiment

V tomto experimentu si demonstrujeme funkčnost na jednom z nejjednodušších z grafů popisující všechny testované vlastnosti.

### 5.1.1 Grafické zobrazení grafu:




*Obrázek 3 - 1. Experiment*

### 5.1.2 Příkaz pro spuštění 1. Experimentu

Příkaz do konzole: *./graph_properties < testData/test1*

### 5.1.3 Výstup vygenerovaný naším programem:

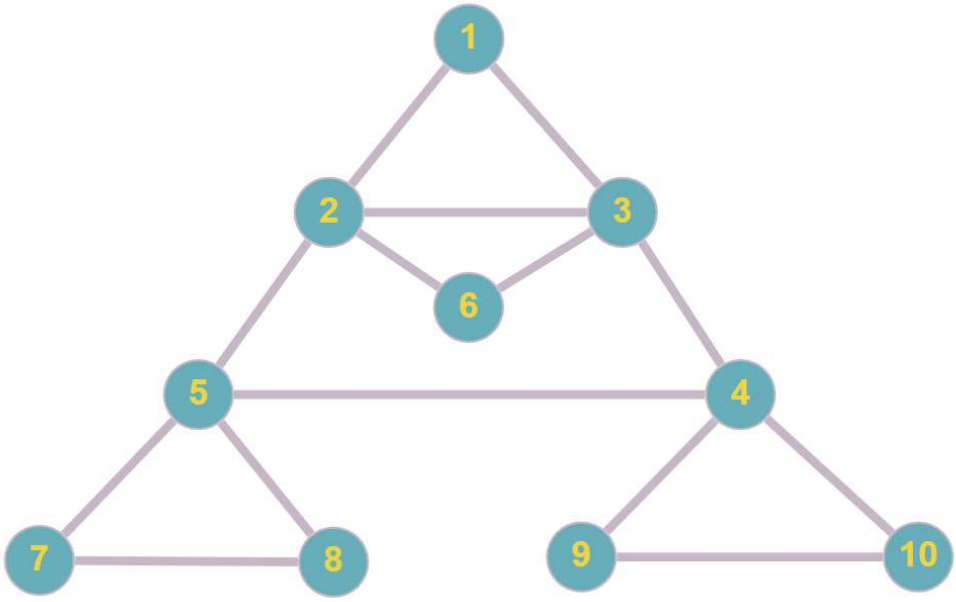
```
==================================
Node count:                      3
Edge count:                      2
Cycle count:                     0
Maximum degree:                  2
Graph is connected:              yes
Graph is complete:               no
Graph is tree:                   yes
Graph is forest                  no
==================================
```

*Obrázek 4 - 1. Experiment - program output*

## 5.2 2. Experiment

V tomto experimentu si funkčnost demonstrujeme na výrazně složitějším grafu. Složitost oproti předchozímu experimentu spočívá ve velkém množství smyček základních, ale také ve velkém množství smyček složených, které je třeba všechny nalézt.

### 5.2.1 Grafické zobrazení grafu:




*Obrázek 5 - 2. Experiment*

### 5.2.2 Příkaz pro spuštění 2. Experimentu

Příkaz do konzole: *./graph_properties < testData/test2*

### 5.2.3 Výstup vygenerovaný naším programem:

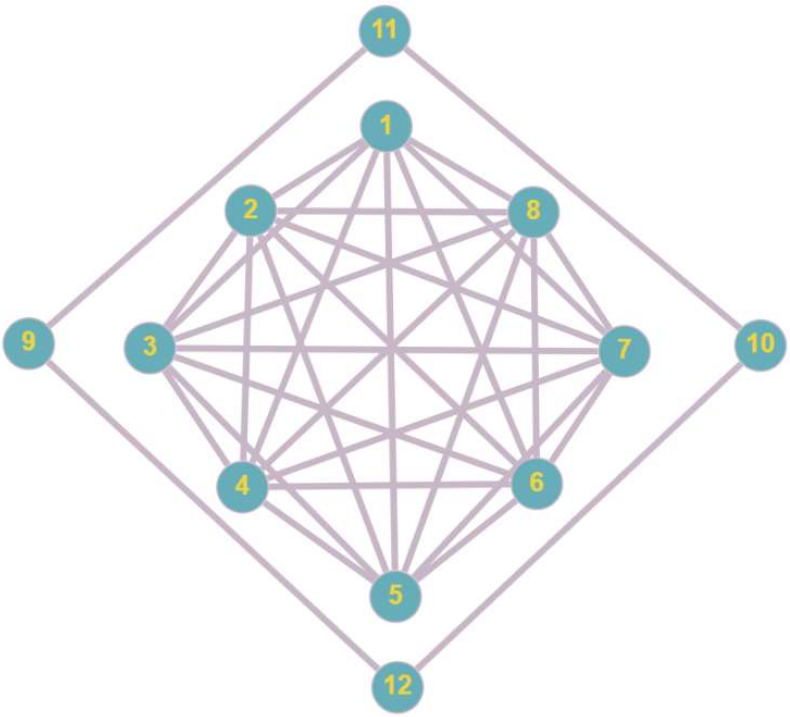
```
==========================================
Node count:                    10
Edge count:                    14
Cycle count:                   8
Maximum degree:                4
Graph is connected:            yes
Graph is complete:             no
Graph is tree:                 no
Graph is forest                no
==========================================
```

*Obrázek 6 - 2. Experiment - program output*

## 5.3 3. Experiment

V posledním experimentu se zaměříme na velmi komplexní řešení, které je zároveň nejsložitější ze všech experimentů, zde uvedených.

### 5.3.1 Grafické zobrazení grafu:




*Obrázek 7 - 3. Experiment*

### 5.3.2 Příkaz pro spuštění 3. Experimentu

Příkaz do konzole: *./graph_properties < testData/test3*

### 5.3.3 Výstup vygenerovaný naším programem:

```
==================================
Node count:                 12
Edge count:                 32
Cycle count:                220
Maximum degree:             7
Graph is connected:         no
Graph is complete:          no
Graph is tree:              no
Graph is forest             no
==================================
```

*Obrázek 8 - 3. Experiment - program output*

## 5.4 Výsledky testování

Výsledky námi provedených testů se shodovali s očekávanými výsledky, čímž jsme ověřili validitu našeho řešení.

# 6 Průběh společné spolupráce

Práce na projektu jsme zahájili hned po získání zadání rozdělením úkolů mezi jednotlivé členy týmu, kteří na nich začali pracovat. V průběhu vypracovávání byla domluva a případné řešení problémů mezi všemi členy v týmu pohotová a věcná. Jako hlavní komunikační kanál jsme si zvolili aplikaci Discord. Pro sdílení a verzování našeho postupu na řešení projektu jsme zvolili aplikaci Github využívající verzovací nástroje Git.

# Seznam obrázků