НАПОМЕНЕ

1. Обавезно прочитати **СВЕ** напомене.
2. Решења писати у оквиру ручно направљеног cargo пројекта.
3. Обавезно уписати име, презиме и број индекса у коментар на почетку датотеке која представља entrypoint програма.
4. Решење које не може да се компајлира носи **0 поена**.
5. Коментарисати оне делове за које сматрате да би коментари додатно појаснили комплексну логику.

Анализа графова за дијагностику кварова на мрежи *FastTransitNetwork (FTN)*

Опис проблема

Студент је ангажован за развој програма који анализира графове великих размера. *FastTransitNetwork* управља великом мрежом чворова и веза (станице, чворови, контролни системи). Током инцидента (прекид везе, квар чвора, деградација сервиса) оператери морају брзо да одговоре на три питања:

- Који чворови су достижни из датог извора и на којој удаљености (у броју корака)?
- Који делови мреже су међусобно повезани ако се занемари смер веза?
- Који чворови су "најважнији" за надзор (приоритизација мониторинга)?

Неопходно је изградити систем за аналитику графова који ради над великим улазним графовима. Потребно је имплементирати и секвенцијалне и паралелне верзије тражених алгоритама и показати тачност и перформансе.

Обим задатка

Потребно је имплементирати:

- Конструкцију графа из фајла (*edge list*).
- Репрезентацију графа погодну за велике графове и паралелну обраду.
- Три функционалности:
  - *BFS* (*Breadth-First Search*): нивое/удаљености од извора у неусмереном смислу потеза (неподнерисани граф).
  - *WCC* (*Weakly Connected Components*): слабо повезане компоненте за усмерени граф.
  - *PageRank* (или скор сличан *PageRank*-у): важност чворова у усмереном графу.

Срећан рад!

Улаз и излаз

Формат улаза

Текстуални фајл. Једна веза по линији. Пример:

Улазна датотека `"ulazniPoda.ci"`

```
// Primer jedne linije:
src dst
```

Где су src и dst ненегативни цели бројеви који денотирају идентификаторе чворова. Граф је подразумевано усмерен. Број чворова се подразумева као max_id + 1 (осим ако се експлицитно не подржава унос броја n).

Формат излаза

- ***BFS*:**
  - **Улаз:** изворни чвор s
  - **Излаз:** низ dist$[v]$:
    - dist$[s] = 0$
    - dist$[v] =$ најкраћи број ивица од s до v
    - dist$[v] = -1$ ако v није достижан из s
- ***WCC*:**
  - **Излаз:** низ comp$[v]$ који додељује ID компоненте за сваки чвор.
  - **Дефиниција:** WCC третира граф као неусмерен (занемарује смер ивица) приликом одређивања повезаности.
- ***PageRank*:**
  - **Излаз:** низ реалних вредности rank$[v]$ (скор важности).

Функционнални захтеви

Имплементирати структуру графа која:

- Омогућава ефикасну итерацију преко суседа.
- Може се безбедно користити у паралелним алгоритмима.

У изввештају обавезно описати:

- Изабрану репрезентацију.
- Евентуалне кораке претпроцесирања.
- Процена меморијске потрошње.

Срећан рад!

Алгоритамски захтеви

- Секвенцијална имплементација: референтна имплементација за проверу тачности.
- Паралелна имплементација: резултат мора бити еквивалентан секвенцијалној.
- Документација:
  - Шта алгоритам рачуна.
  - Очекивана временска сложеност.

Програм треба да буде реализован у програмском језику *Rust*.

Оквирне смернице

- Развити секвенцијалну верзију.
- Паралелизовати програм како би се искористила сва расположива језгра процесора.
- Упоредити перформансе секвенцијалне и паралелне верзије.
- Припремити извештај који објашњава коришћени приступ и резултате мерења.

Технологије и алати

Обавезно је користити следеће технологије и алате:

- **Програмски језик**: *Rust* (мин. верзија 1.75, најсвежија је свакако добродошла).
- **Развојно окружење**: *cargo* (мин. верзија 1.75, најсвежија је свакако добродошла).

Опционално је (али се препоручује) користити следеће технологије и алате:

- ***Rust* библиотеке**: За паралелизацију.
- ***Python***: За визуализацију резултата.

Мерење перформанси

Неопходно је измерити перформансе:

- Секвенцијално vs. паралелно за сваки алгоритам.
- Најмање 4 конфигурације броја нити.
- Најмање 2 величине графа (”мали” и ”велики”).

Минимално известити о:

- Време извршавања.
- Убрзање.

Срећан рад!

Испорука решења

У овом поглављу су описане појединости везане за испоруку решења.

Код (репозиторијум)

Репозиторијум мора да садржи код за:

- конструкцију графа,
- секвенцијалне алгоритме,
- паралелне алгоритме,
- тестове,
- скрипте за мерење перформанси,
- README са *build/run* корацима,

Интерфејс командне линије (*CLI*)

Обезбедити CLI који подржава следеће типове позива (називи, параметри и сличне појединости могу варирати, али функционалност мора постојати).

Улаз / стандардни

```
tool bfs --input edges.txt --source 0 --mode seq --out bfs_seq.txt
tool bfs --input edges.txt --source 0 --mode par --threads 8 --out bfs_par.txt

tool wcc --input edges.txt --mode seq --out wcc_seq.txt
tool wcc --input edges.txt --mode par --threads 8 --out wcc_par.txt

tool pagerank --input edges.txt --mode seq --out pr_seq.txt --alpha 0.85 --iters 50 --eps 1e-10
tool pagerank --input edges.txt --mode par --threads 8 --out pr_par.txt --alpha 0.85 --iters 50 -
```

Тестови

Неопходно је имплементирати тестове:

- јединичне, на малим, ручно конструисаним графовима.
- који пореде секвенцијално и паралелно по правилима тачности.

Извештај и презентација резултата

Неопходно је саставити извештај који у себи садржи следеће целине.

- Кратак опис изабраног алгоритма и мотивације за избор.
- Дизајн репрезентације графа.
- Стратегија верификације.

Срећан рад!

- Графички и/или табеларни приказ резултата, **тумачење**.
- Израчунавање убрзања и дискусија о уским грлима.

Ограничења и правила

- Забрањено коришћење готових библиотека које већ имплементирају *BFS/WCC/PageRank* као ”црну кутију”.

Срећан рад!