Analiza Algorytmów - Projekt

Stawczyk Przemysław 293153

1 Opis Zagadnienia

1.1 Zadane polecenie

Detektyw Karczoch poszukuje nieuczciwych kont na instagramie. Do tego celu analizuje komentarze dotyczące zamieszczanych zdjęć — wie, że takie konta często walczą między sobą, kupując zwolenników i malkontentów. W komentarzach do takich kont, ludzie z jednej grupy piszą tylko do ludzi z drugiej i odwrotnie. Detektyw Karczoch wie, że zwolennik nigdy nie rozmawia z innym zwolennikiem, a malkontent z malkontentem - to dla nich strata czasu.

Zadanie : *Przygotować program, który na podstawie zamieszczonych komentarzy oceni, czy dane konto jest nieuczciwe.*

1.2 Przykładowe dane wejściowe

Uczciwe :
#Jan: Piękne zdjęcie.
#Ola: @Jan Masz rację.
#Ania: @Ola Nie ma racji!
#Jan: @Ania właśnie, że mam!
#Ola: @Ania sama nie masz!

Nieuczciwe :
#Jan: Piękne zdjęcie.
#Ola: @Jan Nieprawda.
#Jan: @Ola czemu tak twierdzisz?
#Ania: @Ola no właśnie?
#Tomek: @Ania twierdzi tak, bo ma rację!!!

1.3 Założenia

- Zakładam, że treść komentarza nie ma znaczenia więc dla opisanego celu liczy się jedynie adresat komentarza.
- Zakładam, że pojedyńczy komentarz może nie mieć adresata lub mieć dokładnie jednego adresata.
- Zakładam, że istnienie pojedyńczych komentarzy od osób które nie adresują do nikogo komentarzy ani nie są adresatami żadnego komentarza nie ma znaczenia dla rozstrzygnięcia uczciwości konta

2 Opis Algorytmu

2.1 Uzywane pojęcia

Przy opisie algorytmu stosuje następujące odwzorowanie:

- Każdy komentujący jest reprezentowany przez wierzchołek grafu.
- Komentarz adresowany do innego użytkownika jest reprezentowany przez krawędź w grafie.
- Wiele komentarzy pomiędzy dwoma użytkownikami nie jest odwzorowane jako wielokrotne krawędzie lub krawędzie ważone.
  Istotny jest fakt komunikacji 2 komentujących, a nie jej objęctość
- Komentarz jest odwzorowany jako krawędz nieskierowana.
  Istotny jest fakt komunikacji 2 komentujących, a nie który z urzytkowników napisał do którego

2.2 Zarys działania

Opis :

Proponowany algorytm opiera się na założeniu, że istnienie 2 grup *[zwolenników i malkontentów]*, które nie piszą komentarzy wewnątrz swoich grup jest równoznaczne z możliwością podzielenia wierzchołków grafu na 2 cześci w których nie ma krawędzi *[dwudzielnosc grafu]*. Pojedyńcze wierzchołki*[uzrytkownicy którzy nie piszą do nikogo, ani nie są adresatami wiadomości]* nie mają znaczenia dla rozstrzygnięcia dwudzielności grafu.

Szkic kroków :

1 Wczytanie listy komentarzy ze standardowego wejścia lub pliku tworząc graf identyfikowany nazwami urzytkowników

2 Usunięcie wierzchołków nieposiadających krawędzi.

3 Przejście po wierzchołkach "kolorując" je na przeciwne kolory zaczynając od dowolnego niepokolorowanego. Następne wierzchołki brane z kolejki.

  I Koloruj wierzchołki sąsiednie na przeciwny kolor. Umieść niepokolorowane wcześniej wierzchołki w kolejce.

  II Gdy wierzchołek po drugiej stronie krawędzi jest w tym samym kolorze:
  STOP –> *Konto jest uczciwe.*

4 Usuń pokolorowane wierzchołki.

5 Jeśli graf niepusty: idź do 3 –> *Przetwarzanie kolejnego spójnego podgrafu*,
jeśli pusty: KONIEC –> *Konto nie jest uczciwe*

3 Złożoność Alorytmu

3.1 Analiza Złożoności Algorytmu

Algorytm iteruje po wszystkich wierzchołkach, a w ramach każdego z wierzchołków po wszystkich jego krawędziach.
Z tego wynika, że złożoność pesymistyczna algorytmu powinna być klasy $\mathbf{O(V+E)}$, gdzie $\mathbf{V}$ to ilość wierzchołków, a $\mathbf{E}$ ilość krawędzi w grafie.

3.2 Metoda pomiarów

Algorytm w przypadku konta uczciwego może zakończyć się wcześniej, więc aby sprawdzać złożoność algorytmu należy mierzyć czasy wykonania analizy nieuczciwych kont. Aby profilować sam algorytm, mierzony powinien być jedynie czas spędzony na obliczeniach więc będzie on mierzony dopiero po wczytaniu całego grafu z pliku do pamięci.

Dla wybranych wielkości grafu mierzony byłby czas dla kilku danych testowych. Analiza byłaby wykonana osobno ze względu na ilość krawędzi *[przy stałej ilości wierzchołków]*, oraz na ilość wierzchołków *[przy stałej ilości krawędzi, jednak większej od liczby wierzchołków]*.

4 Implementacja

4.1 Technologie planowane do wykorzystania

- *C++* jako język do implementacji struktury grafu i algorytmu.
- *Python* lub *C++* jako zewnętrzny program do generowania plików z komentarzami.
- *boost::chrono* w celu pomiaru czasu spędzonego przez algorytm na procesorze.

4.2 Struktury danych

- "Node" - *jako klasa zawierająca nazwe użytkownika, pole określające kolor, identfikowana przez numer pozycji w dynamicznej tablicy*

  W Node : vector<int> - *przechowuje wierzchołki sąsiednie.*

  W Node : enum "Kolor" - *określa czy wierzchołek był odwiedzony oraz kolor odwiedzoneo wierzchołka.*

- map<String, int> - *zapewniająca odwzorowanie nazwy użytkownika na pozycję w tablicy Pomocniczo przy wczytywaniu grafu*
- vector<Node> - *kontener przechowujący wierzchołki*
  Zapewnia dostęp do wierzchołka poprzez numer
- list<int> lub queue<int> - *przechowuje listę wierzchołków do odwiedzenia, umieszczane są w niej kolejni nieodwiedzeni sąsiedzi przetwarzanego wierzchołka*
- "Graph" - *wszystkie struktury danych w ramach makroklasy*

5 Generowanie danych wejściowych

5.1 Parametry:

- liczebność wierzchołków w grafie lub liczebność obu grup z osobna
- liczebność krawędzi w grafie
- uczciwość konta
- nazwa pliku wyjściowego

5.2 Zarys działania:

Przy generowaniu danych w pierwszym kroku tworzone są 2 grupy wierzchołków o podanych licznościach lub wedle losowego podziału jednej liczby na wejściu. Następnie tworzone są

krawędzie *[komentarze]* poprzez losowanie po jednym wierzchołku z każdej grupy. W przypadku zadania parametrem uczciwości konta dodawane są w małej liczbie krawędzie w obrębie grup stworzonych wcześniej. Jako ostatni etap występuje zapis danych do pliku gdzie dla każdej krawędzi losowany jest jeden ze zdefiniowanych w programi komentarzy *[dla uwiarygodnienia danych wyjściowych]*.
Struktury danych analogiczne do wykorzystywanych w algorytmie.