Dokumentacja Implemetacyjna
Podział grafu

Anastasiia Dmytrenko, Arkadiusz Perko

03.04.2025

1 Informacje ogólne

Cel i zakres dokumentacji

Celem tej dokumentacji jest szczegółowe opisanie implementacji programu służącego do podziału grafu. Program został zaprojektowany w języku C i ma na celu efektywny podział grafu na równomiernie wielkościowe części, minimalizując jednocześnie liczbę krawędzi przecinających te części. Podział grafu jest istotny dla zachowania strukturalnej spójności grafu w różnych zastosowaniach, takich jak optymalizacja obliczeń równoległych czy analiza grafów. Dokumentacja ta opisuje szczegóły implementacyjne, strukturę programu, jak również formaty wejściowe i wyjściowe oraz sposób uruchamiania programu.

Parametry uruchomieniowe programu

Program przyjmuje następujące parametry:

- **-i, --input-file** : Ścieżka do pliku z danymi grafu.
- **-o, –output-file** : Ścieżka, w której zapisany zostanie wynik.
- **-r, - -format ascii / binary** : Format pliku wyjściowego (ASCII lub binarny).
- **-p, –parts** : Liczba części, na które graf ma zostać podzielony.
- **-b** : Dopuszczalny margines błędu dla równomierności rozmiarów części (w procentach).
- **-m, –method** (opcjonalnie) : Metoda podziału grafu (domyślnie algorytm Kernighan-Lin, dostępne opcje: ***kl*** (metoda oparta na algorytmie Kernighan-Lin), **m** (metoda oparta na macierzy Laplaciana), **w** (metoda oparta na obliczeniu współczynnika wydajności).
- **Dodatkowe flagi:**
  - **-f, –force** : Zmusza program do podziału nawet w przypadku niemożliwości spełnienia marginesu błędu.
  - **-h, –help** : Wyświetla instrukcję użytkowania programu.

Przykładowe wywołanie programu:

```
./graph_partition -i data.txt -o partition.txt -r ascii -p 4 -b 5 -m kl
```

To wywołanie uruchomi program, który wczyta graf z pliku *data.txt*, podzieli go na *4* części, z marginesem błędu równomierności na poziomie *5%*, używając algorytmu *Kernighan-Lin* do podziału. Wynik zostanie zwrócony w pliku partition.txt w formacie tekstowym.

Środowisko wykonawcze

Program został zaimplementowany w języku C i jest przeznaczony do uruchamiania w systemach operacyjnych opartych na UNIX (np. Linux). Wymaga dostępności kompilatora C (np. GCC) oraz standardowej biblioteki C. Program działa w trybie konsolowym i przyjmuje dane wejściowe w formie pliku tekstowego. Wyniki podziału są zapisywane do pliku tekstowego lub binarnego, w zależności od wybranej opcji.

Wymagania systemowe:

- System operacyjny: Linux, macOS, Windows (z odpowiednim środowiskiem dla C)
- Kompilator C: GCC 4.0 lub nowszy
- Wymagane biblioteki: Standardowa biblioteka C

2 Opis modułów

2.1 Moduł dzielenia grafów algorytmem Kernighan-Lin (klPartition)

Opis modułu: Moduł odpowiedzialny za podział grafu przy użyciu algorytmu Kernighan-Lin. Jest to heurystyczna metoda, która dzieli graf na dwie części, minimalizując liczbę krawędzi między tymi dwiema częściami. Algorytm działa poprzez iteracyjne przemieszczenie wierzchołków pomiędzy dwiema grupami, aby zmniejszyć liczbę krawędzi przecinających te grupy.

Główne zadanie: Algorytm Kernighan-Lin jest najbardziej efektywny w przypadku podziału grafu na dokładnie dwie części. Przy każdym kroku algorytm stara się wymienić wierzchołki między obiema grupami w taki sposób, aby zminimalizować liczbę krawędzi między grupami. Proces powtarza się przez kilka iteracji, aż osiągnięty zostanie optymalny podział.

Jednakże, gdy graf jest dzielony na więcej niż dwie części, algorytm może prowadzić do mniej efektywnych wyników, ponieważ jego iteracyjne podejście jest zoptymalizowane do dzielenia na dwie części.

Interakcje z innymi modułami: Moduł ten współdziała z modułem `input.c`, który dostarcza dane wejściowe, oraz z modułem `output.c`, który zapisuje wynik podziału do pliku. Komunikacja z modułem głównym `main.c` polega na przekazaniu parametrów wyboru metody podziału grafu.

Pliki:

- `klPartition.c` - Implementacja algorytmu Kernighan-Lin.
- `klPartition.h` - Nagłówek zawierający deklaracje funkcji i struktur.

2.2 Moduł dzielenia grafów algorytmem Spectral (spectralPartition)

Opis modułu: Moduł odpowiedzialny za podział grafu przy użyciu metody spektralnej, opartej na analizie macierzy Laplaciana grafu. Metoda ta polega na obliczeniu wartości i wektorów własnych macierzy Laplaciana, które następnie wykorzystywane są do podziału grafu na kilka części.

Główne zadanie: Aby zastosować algorytm spektralny, najpierw obliczamy macierz Laplaciana grafu, która jest podstawową strukturą wykorzystywaną do podziału grafu. Następnie obliczamy wartości i wektory własne tej macierzy. Wektory własne odpowiadają różnym "częściom" grafu, które są następnie wykorzystywane do podziału wierzchołków grafu. Dzięki tej metodzie, algorytm może dzielić graf na więcej niż dwie części, z minimalizacją liczby przecinających się krawędzi.

Metoda spektralna jest uznawana za bardziej skuteczną w przypadku podziału grafów na więcej niż dwie części, ponieważ wykorzystuje pełną analizę strukturalną grafu, a nie tylko lokalne informacje, jak w przypadku algorytmu Kernighan-Lin.

Interakcje z innymi modułami: Podobnie jak w przypadku algorytmu Kernighan-Lin, moduł spektralny współpracuje z modułem `input.c`, który wczytuje dane, oraz z modułem `output.c`, który zapisuje wyniki. Wybór metody (KL lub spektralna) przekazywany jest przez moduł główny `main.c`.

Pliki:

- `spectralPartition.c` - Implementacja algorytmu spektralnego.
- `spectralPartition.h` - Nagłówek zawierający deklaracje funkcji i struktur.

2.3 Moduł wczytywania danych (input)

Opis modułu: Moduł odpowiedzialny za wczytywanie danych wejściowych z pliku tekstowego, który zawiera opis grafu w formie listy sąsiedztwa. Moduł przetwarza dane wejściowe i konwertuje je na odpowiednią strukturę danych, którą następnie wykorzystują inne moduły.

Główne zadanie: Moduł wczytuje dane grafu z pliku tekstowego i przetwarza je na strukturę danych reprezentującą graf. Dodatkowo, moduł obsługuje parametry uruchomieniowe, takie jak ścieżka do pliku z danymi, liczba części, margines błędu i metoda podziału grafu, dostosowując program do wybranych przez użytkownika ustawień.

Interakcje z innymi modułami: Moduł wczytywania danych dostarcza dane wejściowe do modułów dzielenia grafu, przekazując im graf w odpowiednim formacie. Interaguje także z modułem `output.c`, aby przygotować plik wyjściowy, w którym zapisane zostaną wyniki podziału.

Pliki:

- `input.c` - Implementacja wczytywania danych.
- `input.h` - Nagłówek zawierający deklaracje funkcji i struktur.

2.4 Moduł zapisywania danych (output)

Opis modułu: Moduł odpowiedzialny za zapis wyników działania programu do pliku wyjściowego. W zależności od ustawionych parametrów uruchomieniowych, wyniki mogą być zapisane w formacie tekstowym (ASCII) lub binarnym.

Główne zadanie: Moduł zapisuje wyniki podziału grafu do pliku, który może zawierać informacje o wierzchołkach przypisanych do poszczególnych części, liczbie krawędzi przecinających części oraz innych statystykach, zależnych od wybranej metody podziału. Format zapisu (ASCII lub binarny) jest określany przez użytkownika podczas uruchamiania programu.

Interakcje z innymi modułami: Moduł ten odbiera wyniki podziału z modułów dzielenia grafu (KL lub spektralnego) i zapisuje je do pliku. Współpracuje również z modułem głównym `main.c`, który określa ścieżkę i format pliku wyjściowego.

Pliki:

- `output.c` - Implementacja zapisywania wyników.
- `output.h` - Nagłówek zawierający deklaracje funkcji i struktur.

2.5 Moduł główny (main)

Opis modułu: Moduł główny programu, który zarządza uruchamianiem programu, obsługą argumentów wywołania i wyborem odpowiednich metod podziału grafu. Moduł ten jest odpowiedzialny za organizację działania programu, wywoływanie odpowiednich modułów, takich jak `klPartition` lub `SpectralPartition`, oraz obsługę plików wejściowych i wyjściowych.

Główne zadanie: Moduł główny przetwarza argumenty wywołania, ustala, która metoda podziału grafu zostanie użyta, oraz zarządza przepływem danych pomiędzy modułami. Po otrzymaniu danych wejściowych z `input.c`, wywołuje odpowiedni algorytm podziału grafu, a po zakończeniu zapisywania wyników do pliku przez `output.c` program kończy działanie.

Pliki:

- `main.c` - Plik zawierający kod uruchamiający cały program.
- `main.h` - Nagłówek zawierający deklaracje funkcji modułu głównego.

3 Opis struktur

3.1 Struktura `Vertex`

Prototyp struktury:

```
typedef struct vertex {
    int edge_num;
    int *conn;
    int group;
    int fixed;
    int D;
} Vertex;
```

Opis strukrury:

Struktura `Vertex` jest kluczową częścią reprezentacji grafu, w którym przechowywane są informacje o każdym wierzchołku. Każdy wierzchołek zawiera dane dotyczące liczby krawędzi wychodzących z niego, połączeń z innymi wierzchołkami, przynależności do grupy w procesie podziału oraz dodatkowe informacje pomocnicze, jak poprawki w algorytmie Kernighan-Lin (KL).

Argumenty struktury:

- **edge_num**:
  - Typ: `int`
  - Opis: Liczba krawędzi wychodzących z danego wierzchołka. Określa, ile krawędzi łączy dany wierzchołek z innymi wierzchołkami w grafie.
- **conn**:
  - Typ: `int*` (wskaźnik do tablicy)
  - Opis: Tablica przechowująca numery wierzchołków, z którymi dany wierzchołek jest bezpośrednio połączony. Każdy element tej tablicy wskazuje na inny wierzchołek, z którym obecny wierzchołek jest połączony krawędzią.
- **group**:
  - Typ: `int`
  - Opis: Wartość tej zmiennej określa, do której grupy należy dany wierzchołek, gdy graf jest dzielony na mniejsze części.
- **fixed**:
  - Typ: `int`
  - Opis: Określa, czy wierzchołek był poprawiany w trakcie działania algorytmu Kernighan-Lin. Wartość `1` oznacza, że wierzchołek był już przetworzony i nie jest brany pod uwagę przy dalszym podziale, podczas gdy `0` oznacza, że wierzchołek jest nadal aktywny i może być przenoszony między grupami w dalszej iteracji algorytmu.
- **D**:

– Typ: `int`
– Opis: Przechowuje różnicę między liczbą krawędzi przecinających grupy a liczbą krawędzi wewnątrz grupy. Jest to zmienna pomocnicza w algorytmie Kernighan-Lin, używana do oceny, czy zamiana wierzchołków między grupami poprawia podział grafu.

4 Specyfikacja wejścia

Program przyjmuje dane wejściowe w postaci pliku tekstowego o rozszerzeniu `.csrrg`, który opisuje graf w formie listy sąsiedztwa. Format pliku wejściowego musi być zgodny z poniższym opisem:

- **Pierwsza linia:** Maksymalna możliwa liczba węzłów w wierszu (w grafie nie musi znajdować się wiersz o takiej liczbie węzłów).
- **Druga linia:** Indeksy węzłów w poszczególnych wierszach - liczba wszystkich indeksów odpowiada liczbie węzłów grafu.
- **Trzecia linia:** Wskaźniki na pierwsze indeksy węzłów w liście wierszy z punktu 2.
- **Czwarta linia:** Lista sąsiedztwa węzłów grafu.
- **Piąta linia (i każda kolejna w przypadku grafów niespójnych):** Wskaźniki na pierwsze węzły w poszczególnych grupach z linii 4.

Przykładowa zawartość pliku wejściowego:

```
1 4
2 0; 2; 3; 1; 1; 2
3 0; 0; 3; 4; 6
4 0; 1; 1; 2; 3; 2; 3; 5; 3; 4
5 0; 2; 5; 8; 10
```

5 Specyfikacja wyjścia

Po wykonaniu procesu podziału grafu, program generuje plik wyjściowy, który zawiera wynik podziału. Format pliku wyjściowego zależy od wybranego formatu (ASCII lub binarny), jak określono przez użytkownika podczas uruchamiania programu.

5.1 Format pliku: .bin

5.1.1 Opis formatu pliku

Plik binarny wyjściowy zawiera wyniki podziału grafu na różne podgrafy. Jego celem jest zapisanie w sposób efektywny informacji o przypisaniu wierzchołków do podgrafów, oraz o strukturze tych podgrafów. Plik jest zoptymalizowany pod kątem minimalizacji rozmiaru, przechowując takie niezbędbe dane, jak współrzędne wierzchołków, numer podgrafu(grupy), liczba połączeń z innymi wierzchołkami, oraz lista numerów tych wierzchołków.

Plik jest zapisany w formacie binarnym, co pozwala na oszczędność pamięci oraz szybszy dostęp do danych w porównaniu do formatu tekstowego.

5.1.2 Zawartość pliku

Plik binarny zawiera następujące sekcje:

- **Endianness** (uint8): 1 bajt – sposób zapisania bitów w pliku:
  - `0x01` – Little Endian
  - `0x00` – Big Endian
- **File ID** (uint32): 4 bajty – identyfikator pliku.
- **Checksum** (uint32): 4 bajty – suma kontrolna obliczona na podstawie danych pliku (SHA-256).
- **Współrzędne wierzchołka** (uint16): 2 bajty na współrzędną, w tym przypadku 2 współrzędne (x, y).
- **Numer podgrafu** (uint16): 2 bajty – numer podgrafu, do którego należy wierzchołek.
- **Liczba połączeń** (uint16): 2 bajty – liczba połączeń z innymi wierzchołkami.
- **Połączenia** (uint16): 2 bajty na każdy numer wierzchołka, z którym dany wierzchołek jest połączony.

Uwaga: numery wierzchołków w Połączeniach nie są dublowane!

Wszystkie dane są zapisane w porządku binarnym, gdzie każda wartość ma stałą długość. Struktura pliku jest zoptymalizowana pod kątem minimalnego rozmiaru oraz efektywnego przechowywania danych.

5.1.3 Struktura pliku

Plik binarny składa się z następujących pól:

Nazwa pola	Wielkość w bitach	Opis
Endianness	1	Sposób zapisania(1 - little endian, 0 - big endian)
File Id	32	FileID
Checksum	32	Suma kontrolna
X	16	Współrzędna X wierzchołka
Y	16	Współrzędna Y wierzchołka
Group Number	16	Numer podgrafu wierzchołka
Conn Number	16	Liczba połączeń wierzchołka
Connections	16 x n	Połączenia wierzchołka (n to Adjacency Number)

5.1.4 Przykład formatu zapisu

Załóżmy, że mamy graf z 3 wierzchołkami, podzielony na 2 podgrafy:

- Wierzchołek 1: współrzędne (2, 3), należy do podgrafu 1, ma 2 połączenia z wierzchołkami 2 i 3.

- Wierzchołek 2: współrzędne (4, 5), należy do podgrafu 1, ma 1 połączenie z wierzchołkiem 3.
- Wierzchołek 3: współrzędne (6, 7), należy do podgrafu 2, ma 1 połączenie z wierzchołkiem 1.

Przykładowy zapis w formacie binarnym:

Nazwa pola	Dane	Opis
Endianness	0x01	Sposób zapisania bitowy(1 - little endian, 0 - big endian)
File Id	0x54554c12	FileID
Checksum	0xc40bb90d	Suma kontrolna
X	00 02	Współrzędna X wierzchołka 1
Y	00 03	Współrzędna Y wierzchołka 1
Group Number	00 01	Numer podgrafu wierzchołka 1
Conn Number	00 02	Liczba połączeń wierzchołka 1
Connections	00 02 00 03	Lista połączeń wierzchołka 1

5.2 Format pliku: .txt

5.2.1 Opis formatu pliku

Plik tekstowy zawiera dane o wierzchołkach grafu. Każdy wierzchołek wraz z informacjami zapisywany jest na oddzielnej linii. Format ten zawiera jedynie wartości liczbowe int oddzielone średnikami, co czyni go bardziej kompaktowym.

Plik zawiera następujące informacje o grafie i dla każdego wierzchołka:

- **Liczba wierzchołków**: Całkowita liczba wierzchołków w grafie.
- **Liczba podgrafów**: Całkowita liczba podgrafów w grafie.
- **Współrzędne wierzchołka**: Współrzędne (X, Y) wierzchołka.
- **Numer podgrafu**: Numer podgrafu, do którego należy wierzchołek.
- **Liczba połączeń**: Liczba połączeń wierzchołka z innymi wierzchołkami.
- **Połączenia**: Numery wierzchołków, z którymi dany wierzchołek jest połączony.

Uwaga: numery wierzchołków w Połączeniach są dublowane!

Format pliku tekstowego:

```
1 linia: [Liczba_wierzchołków]
2 linia: [Liczba_podgrafów]
3 i każda kolejna linia: [Współrzędna_X]; [Współrzędna_Y];[Numer_podgrafu];
[Liczba_połączeń]; [Połączenia: numer_wierzchołka_1 numer_wierzchołka_2 ...
numer_wierzchołka_n]
```

5.2.2 Struktura pliku

Plik tekstowy ma następującą strukturę:

```
[Liczba wierzcholkow]
[Liczba podgrafow]
[Wspolrzedna X wierzcholka 1];[Wspolrzedna Y wierzcholka 1];[Numer
   podgrafu];[Liczba polaczen wierzcholka 1];[Polaczenia]
[Wspolrzedna X wierzcholka 2];[Wspolrzedna Y wierzcholka 2];[Numer
   podgrafu];[Liczba polaczen wirzcholka 2];[Polaczenia]
...
```

5.2.3 Przykład formatu zapisu

Załóżmy, że mamy graf z 3 wierzchołkami, podzielony na 2 podgrafy:

- **Wierzchołek 1**: współrzędne (2, 3), należy do podgrafu 1, ma 2 połączenia z wierzchołkami 2 i 3.
- **Wierzchołek 2**: współrzędne (4, 5), należy do podgrafu 1, ma 1 połączenie z wierzchołkiem 3.
- **Wierzchołek 3**: współrzędne (6, 7), należy do podgrafu 2, ma 1 połączenie z wierzchołkiem 1.

Przykładowy zapis w formacie tekstowym:

```
3
2
2;3;1;2;2;3
4;5;1;1;3
6;7;2;1;1
```