

Specyfikacja implementacyjna programu dzielącego graf

Adam Domański, Oliwier Osiński

28.04.2025

Struktura plików

Opisywany program składa się z następujących katalogów i plików:

- Makefile - plik wykonywalny kompilujący i uruchamiający różne testowe scenariusze programu;
- bin - katalog zawierający skompilowany program o nazwie "graf";
- include - katalog zawierający pliki nagłówkowe;
- src - katalog zawierający pliki źródłowe c;
- tests - katalog zawierający testowe pliki wejściowe opisujące grafy

Argumenty wywołania

Do poprawnego uruchomienia programu niezbędne jest podanie jednego obowiązkowego argumentu. Opcjonalnie można dodać dwa dodatkowe argumenty zmieniające parametry programu.

1. **plik.in** - ścieżka do pliku, w którym zapisany jest graf przeznaczony do podziału.

W przypadku, gdy podany plik nie istnieje, program zwróci błąd o treści: Błąd: Nie udało się otworzyć pliku wejściowego o podanej ścieżce. Przerywam działanie. i zakończy działanie.

Natomiast, gdy dane przedstawiające graf są niepoprawne, program zwróci błąd o treści: "Błąd: Dane w pliku przedstawiające graf są niepoprawne. Przerywam działanie." i zakończy działanie.

2. **N** (opcjonalny) - dodatnia liczba całkowita podzielenia grafu na podgrafy, której wartość domyślna wynosi 1.

W przypadku, gdy nie poda się wartości liczbowej, program zwróci błąd o treści: "Błąd: Liczba podzielen grafu została niepoprawnie zdefiniowana. Przerwywam działanie." i zakończy działanie.

Natomiast, gdy podana liczba, będzie niedodatnie lub zmiennoprzecinkowa, to program zwróci błąd o treści: "Błąd: Liczba podzielen grafu musi być większa bądź równa 1. Przerwywam działanie." i zakończy działanie.

3. **M** (opcjonalny) - nie ujemna liczba całkowita nie przekraczająca wartości 100, liczba przedstawia graniczną wartość procentową, pod którą musi się zmieścić różnica wierzchołków podzielonych podgrafów, jej domyślna wartość wynosi **10**.

W przypadku, gdy nie poda się wartości liczbowej, program zwróci błąd o treści: "Błąd: Liczba marginesu różnicy procentowej została niepoprawnie zdefiniowana. Przerwywam działanie." i zakończy działanie.

Natomiast, gdy podana liczba, będzie ujemna lub zmiennoprzecinkowa, to program zwróci błąd o treści: "Błąd: Liczba marginesu różnicy procentowej między wierzchołkami powstałych grafów musi znajdować się w przedziale [0-100]. Przerwywam działanie." i zakończy działanie.

Przykłady użycia argumentów podczas wywoływania programu znajdują się w sekcji **Uruchomienie programu**.

Flagi

Program przyjmuje następujące flagi:

- **-h** - flaga wyświetlająca informacje o argumentach programu oraz dostępne flagi wraz z ich opisami.
- **-o plik.out** - flaga przyjmująca jako argument ścieżkę do pliku, do którego ma zostać zapisany wynik końcowy programu. Domyślna wartość argumentu flagi to **"wynik.txt"**.
- **-b** - flaga zmienia tryb wyświetlania wyniku z domyślnie tekstowego na binarny.
- **-t** - flaga sprawiająca, że wynik końcowy programu zostanie wypisany w terminalu. Można łączyć z flagą -b, wtedy w terminalu zostanie wyświetlony wynik binarny.

Przykłady użycia flag podczas wywoływania programu znajdują się w sekcji **Uruchomienie programu**.

Uruchomienie programu

Do kompilacji programu należy użyć komendy **make** w terminalu lub od razu wybrać wcześniej przygotowany scenariusz testowy, wpisując **make {nazwa_testu}** o których więcej w sekcji **Testy**.

Aby uruchomić program należy w terminalu wywołać plik wykonywalny znajdujący się w katalogu bin: **/bin/graf**, a następnie podać odpowiednie argumenty, o których mowa była w sekcjach **Argumenty wywołania** oraz **Flagi**.

Przykład wywołania programu wczytującego graf z pliku `/tests/graf.csrrg`, wypisującego wynik w trybie tekstowym w terminalu oraz do pliku `podzial.txt`:
./bin/graf /tests/graf.csrrg 2 20 -o podzial.txt -t

Przykład wywołania programu wczytującego graf z pliku `/tests/graf2.csrrg`, wypisującego wynik w trybie binarnym w terminalu:
./bin/graf /tests/graf2.csrrg -t -b

Format wyjściowy

Program zawsze zapisuje końcowy wynik w pliku, który został podany w odpowiedniej fladze podczas wywoływania (domyślnie "wynik.txt").

Na wynik końcowy w trybie tekstowym (domyślny) składa się:

- Liczba udanych podziałów grafu w pierwszej linijce.
- Następnie graf w takim samym formacie co w pliku wejściowym.

W przypadku trybu binarnego, podawany jest jedynie sam graf. Sposób zapisania grafu w postaci binarnej przedstawia sekcja **Plik binarny**.

Plik binarny

W trybie binarnym każda liczba reprezentowana jest binarnie na 32 bitach. Znak `\n` jest reprezentowany przez ciąg 32 jedynek, natomiast znak `;` przez ciąg 31 jedynek i zera.

Przykładowo:

- Liczba 5 jest reprezentowana przez 00000000000000000000000000000101
- Znak `\n` przez 11111111111111111111111111111111
- Znak `;` przez 11111111111111111111111111111110

Pliki źródłowe

W katalogu src znajdują się następujące pliki:

- **main.c** - plik, w którym obsługiwane są argumenty i flagi wywołujące program oraz inicjalizuje podział grafu.
- **graf.c** - plik zawierający funkcje do obsługi struktury grafu oraz funkcje odpowiedzialne za faktyczny podział grafu.
- **file_graph.c** - plik zawierający funkcje odpowiedzialne za odczyt grafu z pliku, jak i jego zapis do pliku.
- **getopt.c** - plik zawierający funkcje do poprawnego i bezbłędnego odczytu argumentów oraz flag wywoływanego programu.

Struktury

Struktura grafu:

```
typedef struct {  
    int n;  
    int start;  
    int parent;  
    Node ** adj;  
} Graph;
```

gdzie:

- n - liczba wierzchołków grafu
- start oraz parent - zmienne potrzebne do podziału grafu
- adj - "lista" wierzchołków grafu

Struktura wierzchołka:

```
typedef struct n{
    int vertex;
    struct n *next;
} Node;
```

gdzie:

- vertex - etykieta wierzchołka
- next - wskaźnik do następnego wierzchołka

Funkcja podziału grafu

Użyty w naszym programie algorytm do podziału grafu jest uproszczoną wersją algorytmu Kerninghana-Lina.

W pierwszym kroku wierzchołki są przypisywane w miarę po równo do dwóch części (0 i 1). Następnie za pomocą pętli, algorytm przechodzi kolejno wszystkie wierzchołki i sprawdza czy przeniesienie danego wierzchołka do drugiej części zmniejsza liczbę przecięć. Warto zaznaczyć, że wierzchołek jest przenoszony tylko wtedy gdy nie zaburzy równowagi dwóch części. Takim sposobem udaje nam się uzyskać dwa grafy z grafu pierwotnego. Ze względu, że nowo powstałe grafy mają tyle samo wierzchołków co pierwotny graf, to nasz algorytm szuka tej granicy między tymi grafami. Następnie zmniejsza grafy do ich faktycznych rozmiarów, czyli pozbywa się wierzchołków, które wykorzystuje drugi graf. Kolejny algorytm sprawdza czy zgadza się różnica wierzchołków z podanym marginesem, jeśli się zgadza, nowe grafy są dodawane do tablicy grafów. W przeciwnym razie, brany jest kolejny graf z tablicy grafów. Pod koniec wszystkie grafy z tablicy grafów są scalane w jeden duży graf z pierwotnymi etykietami wierzchołków, co powoduje powstanie grafu z taką samą liczbą wierzchołków, co graf z pliku wejściowego.

Link do repozytorium:

https://github.com/t0q1/JIMP2_graph_C/tree/main