

# Specyfikacja implementacyjna programu dzielącego graf

Adam Domański, Oliwier Osiński

30.04.2025

## 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 podzielenie 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 flagie podczas wywoływania (domyślnie "wynik.txt").

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

- Liczba udanych podziałów grafu w pierwszej linii.
- 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

Przykład: graf w formie tekstowej: 5\n0;2;4;1;3;0;1;2;3\n0;3;5;9\n0;1;4;7;5;5;8;6;8;7\n0;2;5;7  
zapisa w postaci binarnej:

Zapis grafu w formacie binarnym nie zawiera spacji, tylko stały ciąg zer i jedynek. Powyższy przykład zawiera spacje dla łatwiejszej wizualizacji.

## 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 reprezentowane w tablicy intów o długości ilości wierzchołków w grafie, etykieta węzła odpowiada indeksowi w tej tablicy. Do tablicy przypisywane są po równo wartości (0 i 1), wartości te będą reprezentować na koniec dwa grafy, jeden z wierzchołkami o wartościach 0 i drugi z wierzchołkami o wartościach 1. Następnie przy pomocy pętli, algorytm przechodzi kolejno wszystkie wierzchołki w tablicy i sprawdza czy przeniesienie danego wierzchołka (zmiana wartość na przeciwną) do drugiej części zmniejsza liczbę przecięć. Warto zaznaczyć, że wierzchołek jest przenoszony tylko wtedy gdy nie zaburzy równowagi dwóch części. Następnie z powstałej tablicy składającej się z 0 i 1, są tworzone dwa grafy. W przypadku gdy powstałe grafy spełniają warunek różnicy liczby wierzchołków, to pierwotny dzielony graf w tablicy grafów jest podmieniany na pierwszy nowo powstały graf, a drugi powstały graf jest dodawany na koniec tablicy grafów. W przypadku gdy nie spełniają tego warunku, indeks wskazujący na obecny graf w tablicy grafów jest zwiększany o jeden i proces podziału zaczyna się od początku dla kolejnego grafu w kolejce. Po zakończeniu wszystkich podziałów grafów, wszystkie grafy z tablicy grafów zostają scalone w jeden duży graf, który ma tyle samo wierzchołków co graf wczytywany z pliku wejściowego. Ze względu na parametry start i parent w strukturze grafu, każdy graf zna pierwotne wartości etykiet z wczytywanego grafu. Następnie funkcja zwróci liczbę udanych podziałów grafów i cały proces dobiegnie do końca.