# Cel projektu

Program `przetnij-graf` służy do podziału zadanego, nieskierowanego grafu prostego na określoną liczbę części (podgrafów) z zachowaniem balansu wielkości w granicach zadanej tolerancji. Umożliwia to dalszą analizę lub przetwarzanie rozłącznych fragmentów grafu.

# Dane wejściowe

Program przyjmuje w pierwszym argumencie ścieżkę do pliku z reprezentacją grafu w postaci tekstowej:

1. **Pierwsza linia**: dwie liczby całkowite (oddzielone spacją):
   - `V` – liczba wierzchołków,
   - `E` – liczba krawędzi.
2. **Następne `E` linii**: każda linia to para `src dest` ($0 \quad$ `src`,`dest` $<$ `V`), oznaczająca nieukierunkowaną krawędź.

**Przykład**:

```
6 7
0 1
0 2
1 2
2 3
3 4
4 5
3 5
```

# Wywołanie programu

`Usage: przetnij-graf <input file> <N> <M> [-o <output file>] [-t] [-b]`

- `<input file>` Plik tekstowy z grafem.
- `<N>` Liczba **podgrafów** do utworzenia.
- `<M>` Margines nierównowagi (float). Każdy podgraf będzie miał liczbę wierzchołków mieszczącą się w przedziale $rednia–M, rednia + M$, gdzie średnia = V / N.

Opcjonalne:

- `-o <output file>` Ścieżka pliku wyjściowego (domyślnie `plik.out`).
- `-t` Po wygenerowaniu pliku — wydrukuje reprezentację tekstową grafu na terminal.
- `-b` Zapisze graf w formacie binarnym zamiast tekstowego.

# Format wyjściowy

## 1. Tekstowy

Funkcja `graphToTextFile` (oparta na `graphToString`) generuje tekst:

```
V' E'
src1 dest1
src2 dest2
...
srcE' destE'
splitCount
```

- `V'`, `E'` – liczba wierzchołków i krawędzi w wynikowym grafie.
- Następnie `E'` linii z parami wierzchołków.
- Ostatnia linia: `splitCount` – wartość `<N>` przekazana w argumencie (funkcja w kodzie ustawia `balancedGraph->splitCount = N;`).

Po zapisie pojawi się na wyjściu:

```
Graph written to text file: <output file>
```

## 2. Binarny

Funkcja `graphToBinaryFile` zapisuje (little-endian) tylko:

1. `int32` – liczba wierzchołków (`numVertices`)
2. `int32` – liczba krawędzi (`numEdges`)
3. Dla każdej krawędzi:
   - `int32` – `src`
   - `int32` – `dest`

   **Uwaga:** w formacie binarnym **nie** jest zapisywana wartość `splitCount`. Po zapisie pojawi się komunikat:

```
Graph written to binary file: <output file>
```

# Schemat działania (`splitGraph`)

1. **Inicjalizacja** Kopiowanie oryginalnego grafu (`copyGraph`), odczyt liczby wierzchołków `totalV`.
2. **Obliczenie docelowych rozmiarów**

   ```
   baseSize = totalV / N;
   remainder = totalV % N;
   for (i=0; i<N; i++)
     desiredSizes[i] = baseSize + (i < remainder ? 1 : 0);
   ```

– pierwsze `remainder` grup ma o jeden wierzchołek więcej.

3. **Wielośródłowe BFS**
   - Wybór `N` wierzchołków-ziaren o największym stopniu.
   - Rozszerzanie kolejnych grup za pomocą BFS, aż każda osiągnie swój `desiredSize`.
4. **Dopasowanie balansujące** Jeżeli któraś grupa odbiega od `desiredSizes[i]` o więcej niż `M`, wykonywane są heurystyczne przeniesienia wierzchołków między grupami w celu zredukowania odchyleń (pętle korekcyjne).
5. **Rekonstrukcja krawędzi** Dla każdej grupy zbierane są krawędzie, których oba końce należą do tej samej części. Aktualizowane są pola `numVertices`, `numEdges`.
6. **Zapis wyniku** Funkcja zwraca wskaźnik `Graph *` z polami:
   - `numVertices`, `numEdges`,
   - `edges[]` – przefiltrowana lista krawędzi,
   - `splitCount` = `N`.

# Przykłady użycia

```
# Podział na 4 podgrafy (N=4) z marginesem 0.10, tekstowo:
./przetnij-graf graph.txt 4 0.10 -o podzielony.txt

# Taki sam, plus wydruk w terminalu:
./przetnij-graf graph.txt 4 0.10 -o podzielony.txt -t

# Zapis w formacie binarnym:
./przetnij-graf graph.txt 4 0.10 -o podzielony.bin -b
```

# Komunikaty błędów

- **Brak wymaganych argumentów** Wypisuje:

  `Usage: przetnij-graf <input file> <N> <M> [-o <output file>] [-t] [-b]`

- **Nieznany przełącznik**

  `Unknown argument: <arg>`

- **Błąd wczytania grafu** Gdy `graphFromTextFile` zwraca `NULL`:

  `Failed to load graph from file: <input file>`

- **Błąd podziału** Gdy `splitGraph` zwróci `NULL` (np. nieudane spełnienie ograniczenia marginesu):

  `Failed to split graph into <N> subgraphs with margin <M>`

- **Błędy zapisu wyjścia** Brak dodatkowych komunikatów (funkcja `graphToBinaryFile`/`graphToTextFile` pomija błędy `fopen` lub `fwrite`).

## Uwagi końcowe

- Graf wejściowy musi być prosty (bez pętli i powtórzeń krawędzi) oraz nieskierowany.
- Dla dużych wartości `N` i małych `M` algorytm może nie znaleźć podziału; wtedy nastąpi błąd podziału.
- Warto testować wartości argumentów wejściowych na mniejszych grafach przed przetwarzaniem bardzo dużych.