Specyfikacja funkcjonalna programu służącego do analizy grafów.

Filip Sosnowski, Krzysztof Tadeusiak

17.03.2022

Cel projektu

Program służy do analizy grafu wygenerowanego na podstawie podanych przez użytkownika argumentów. Umożliwia znalezienie najkrótszej drogi pomiędzy wybraną parą węzłów, a także oferuje sprawdzenie spójności grafu. Program działa w trybie wsadowym. Wygenerowane dane są zapisywane do pliku tekstowego. W naszym projekcie wykorzystane jest działanie algorytmu Breadth-first Search (BFS) oraz algorytmu Dijkstry.

Teoria

Graf - zbiór wierzchołków, które mogą być połączone krawędziami, w taki sposób, ze każda krawędź kończy się i zaczyna w pewnym wierzchołku. Wierzchołki grafu są numerowane. Krawędzie obrazują relacje pomiędzy obiektami.

Graf wagowy - krawędzie mogą mieć przypisaną wagę, tzn. przypisaną wartość liczbową. Określa ona w naszym projekcie długość drogi - im mniejsza waga, tym krótsza droga pomiędzy dwoma wierzchołkami.

Spójność grafu - graf jest spójny, gdy każda para wierzchołków jest połączona ścieżką.

Algorytm BFS - to algorytm, który przeszukuje grafy wszerz. Służyć może do sprawdzenia spójności grafu lub do znalezienia najkrótszej drogi pomiędzy dwoma wybranymi wierzchołkami grafu.

Algorytm Djikstry - algorytm, który służy do znalezienia na grafie najkrótszej drogi między określonym węzłem a każdym innym. Znajduje również zastosowanie, gdy należy znaleźć najkrótszą ścieżkę od określonego wierzchołka startowego do wybranego węzła końcowego. Algorytm Dijkstry na wejściu wymaga grafu skierowanego lub nieskierowanego o dodatnich wagach krawędzi.

Dane wejściowe

Program wymaga danych, które opisują parametry tworzonego grafu. Każdy z grafów musi być opisany następującymi parametrami:

- ilość kolumn: na jej podstawie program może utworzyć siatkę o danej ilości wierzchołków znajdujących sie kolejno po sobie wzdłuż osi X. Dopuszczalne wartości: x > 0, x $\leq 10^8$, x$\times y \leq 10^8$ oraz $x \in C$;
- ilość wierszy: na jej podstawie program może utworzyć siatkę o danej ilości wierzchołków znajdujących sie kolejno po sobie wzdłuż osi Y. Dopuszczalne wartości: y > 0, y $\leq 10^8$, x$\times y \leq 10^8$ oraz $y \in C$;
- minimalna wartość wagi krawędzi: determinuje jaka minimalna waga może pojawić się między wierzchołkami w grafie. Dopuszczalne wartości: min $\geq 0, min \leq 1$ oraz min$\in R$;
- maksymalna wartość wagi krawędzi: determinuje jaka maksymalna waga może pojawić się między wierzchołkami w grafie. Dopuszczalne wartości: max $\geq 0, max \leq 1$ oraz max$\in R$;
- punkt startowy: determinuje, z którego wierzchołka program ma zacząć znajdować najkrótszą drogę do punktu końcowego. Dopuszczalne wartości: ps $\geq 1, ps \leq$ x$\times y$ oraz $ps \in C$;
- punkt końcowy: determinuje, do którego wierzchołka program ma wyznaczyć najkrótszą drugę z punktu startowego. Dopuszczalne wartości: pk $\geq 1, pk \leq$ x$\times y$ oraz $pk \in C$;
- liczba podziału grafu: determinuje, ile grafów otrzymamy z podstawowo wygenerowanego grafu. Dopuszczalne wartości: n $\geq 1, n \leq \frac{x \times y}{4}$ oraz $n \in C$;

Dane wyjściowe

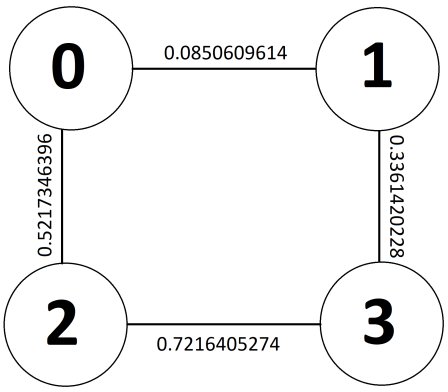
Program ten generuje graf o określonych wymiarach oraz wartościach wag krawędzi oraz zapisuje go do pliku o ustalonym formacie.

Przykładowy plik z grafem:

```
2 2
        2: 0.5217346396 1: 0.0850609614
        3: 0.3361420228 0: 0.0850609614
        0: 0.5217346396 3: 0.7216405274
        2: 0.7216405274 1: 0.3361420228
```

W pierwszej linijce określona jest liczba kolumn oraz wierszy. Kolejne wiersze opisują wartości wag krawędzi w rozważanym grafie.

Poniżej załączona jest wizualizacja graficzna wygenerowanego grafu.



Program określi spójność grafu za pomocą algorytmu BFS. Wskaże także najkrótszą możliwą drogę między wybranymi węzłami, korzystając z algorytmu Dijkstry. Obie te informacje zostaną zakomunikowane w strumieniu wyjścia.

Argumenty wywołania programu

Komenda wykorzystana do uruchomienia programu to

./graf [argumenty_wywołania]

Nasz program przyjmuje poniżej podane argumenty wywołania:

- `-x (liczba)` określa liczbę kolumn w grafie;
  Jest to flaga obowiązkowa.
- `-y (liczba)` określa liczbę wierszy w grafie;
  Jest to flaga obowiązkowa.
- `-n (liczba)` określa liczbę grafów otrzymanych z pierwotnie wygenerowanego grafu;
  domyślnie `n = 1`;
- `-min (liczba)` określa minimalną wartość wagi krawędzi między węzłami;
  domyślnie `min = 0`;
- `-max (liczba)` określa maksymalną wartość wagi krawędzi między węzłami;
  domyślnie `max = 1`;
- `-ps (numer_węzła)` określa punkt startowy;
  domyślnie `ps = 1`;
- `-pk (numer_węzła)` określa punkt końcowy;
  domyślnie `pk =` $x \times y$;

- `-out (nazwa_pliku)` określa nazwę pliku, do którego zostaną zapisane dane;
  domyślnie `output = mygraph`;

Projekt ten może działać także w drugim trybie. Program potrafi przeczytać graf z pliku o ustalonym formacie. Służy do tego flaga:

- `-in (nazwa_pliku)` określa nazwę pliku, z którego zostaną odczytane dane;

W tym przypadku dopuszczalne są jedynie flagi `-ps (numer_węzła)` oraz `-pk (numer_węzła)` określające punkt startowy oraz końcowy, które posłużą do znalezienia najkrótszej drogi w odczytanym z pliku grafie.

Przykładowe wywołania programu:

- `./graf -x 1300 -y 780 -min 0.2 -max 0.6 -ps 144 - pk 676 -output plik.txt`

Powyższe wywołanie programu utworzy graf o 1300 kolumnach i 780 wierszach. Minimalna wartość wagi krawędzi wynosi 0.2, a maksymalna 0.6. Wybranym punktem startowym jest węzeł 144, a końcowym 676. Graf ten zostanie zapisany do pliku "plik.txt". Nie określono wartości flagi -n, dlatego wygeneruje jeden graf.

- `./graf -in mygraph -ps 17 -pk 7`

Powyższe wywołanie programu odczyta graf z pliku "mygraph". Ustalonym punktem startowym jest węzeł 17, zaś końcowym 7.

Komunikaty błędów

Dane numeryczne akceptowalne są jedynie w systemie dziesiętnym rzeczywistym. Funkcję separatora dziesiętnego będzie pełnić kropka („.”). Program nie obsługuje podanych przez użytkownika liczb ujemnych. Wyświetla on wtedy odpowiedni komunikat błędu oraz prosi użytkownika o sprawdzenie danych i ich ponowne wprowadzenie. Program pobiera od użytkownika wartość 0 tylko i wyłącznie dla danych "minimalnej wartości wag krawędzi" oraz "maksymalnej wartości wag krawędzi". Dla innych danych wyświetla komunikat błędu i prosi o ponowne ich wprowadzenie: `"Niepoprawnie wprowadzone dane. Prosze spróbować ponownie."`

Wartości liczby kolumn i wierszy, punktu startowego, punktu końcowego oraz liczby podziału grafu należy podać jako liczby całkowite. W przeciwnym wypadku program zaokrągli je do najbliższej mniejszej liczby całkowitej będacej wiekszą niż 0 (w przypadku podania liczby od 0 do 1 (bez 0) program domyślnie ustawia 1). Przy podaniu minimalnej wartości wag krawędzi większej niż wartość maksymalna, program wyświetli błąd oraz poprosi użytkownika o ponowne wprowadzenie danych:

`"Minimalna wartość wagi krawędzi jest większa od podanej maksymalnej wartości. Prosze wprowadzić dane ponownie."`

Ustalenie punktu początkowego odpowiadającego punktowi końcowemu, skutkować będzie określeniem wartości najkrótszej drogi jako 0. Przy stworzeniu grafu i niesprecyzowaniu początkowego oraz końcowego węzła, program obierze punkt początkowy jako węzeł 1, a punkt końcowy jako węzeł $(x \times y)$.

Program umożliwia stworzenie grafu o wymiarach 1×1. W takiej sytuacji początek i koniec grafu będą znajdowały sie na tym samym miejscu o wartości 1. Wartość najkrótszej drogi między początkiem a końcem wyniesie 0. Program wypisze błąd po otrzymaniu argumentu n większego niż $\frac{x \times y}{4}$: `"Niepoprawnie wprowadzona liczba podziału grafu (n)."`

Przy losowym wyborze wierzchołka rozpoczynającego dzielenie możliwe jest uzyskanie maksymalnie podzielonego grafu w mniej niż $\frac{x \times y}{4}$ iteracjach. Program w tym wypadku wypisze w ilu iteracjach graf został maksymalnie podzielony.

Wszystkie inne argumenty podane przez użytkownika, które przekraczają zdefiniowane wcześniej zakresy, będą skutkować wypisaniem błędu: `"Wprowadzone dane przekraczają określone zakresy. Prosze spróbować ponownie."`