Sprawozdanie końcowe z programu na projekt CGrafy

Szymon Posiadała i Jordan Parviainen

23.04.2022r.

1 Osiągnięcie celu projektu

Cel projektu będący wytworzeniem oprogramowania operującego na grafach uznaje się za osiągnięty. Program tworzony w ramach projektu znajduje najkrótszą możliwą ścieżkę pomiędzy dwoma wybranymi wierzchołkami oraz sprawdza, czy graf jest spójny. Potrafi on generować grafy o zadanej liczbie kolumn i wierszy w 3 trybach. Program dodatkowo wyposażony jest w możliwość zapisu wygenerowanego grafu do pliku oraz odczytu grafu z takiego pliku. W programie wykorzystywane są dwa algorytmy:

- algorytm Dijkstry – algorytm dzięki, któremu wyszukiwana jest najkrótsza ścieżka,
- algorytm BFS – algorytm umożliwiający sprawdzenie czy graf jest spójny.

2 Co zostało zaimplementowane

- main.c - główny moduł sterujący pracą programu i przetwarzający argumenty wejścia
- graph_generator.c
  Generator grafu oferuje stworzenie grafu w 3 trybach:
  1. tryb pierwszy, gdzie każdy wierzchołek grafu ma połączenie z każdym sąsiadującym wierzchołkiem
  2. tryb drugi, gdzie graf generowany jest losowo na zadanym obszarze(siatce) i graf w obszarze wygenerowanym jest spójny
  3. tryb trzeci, gdzie graf generowany jest losowo na zadanym obszarze(siatce) i graf w obszarze wygenerowanym jest niekoniecznie spójny
- file_handler.c
  1. parseGraphFromFile(): funkcja ta odczytuje graf z pliku o formacie opisanym w Specyfikacji Funkcjonalnej zapisuje go do struktury danych w programie
  2. saveGraphToFile(): funkcja ta zapisuje graf ze struktury danych do pliku w formacie opisanym w Specyfikacji Funkcjonalnej
- graph_cohesion.c
  1. isGraphCohesive(): funkcja wykorzystująca algorytm BFS sprawdzająca czy graf jest spójny. Dodatkowo zwraca tablicę przeszukanych wierzchołków.
- path_finder.c
  findShortestPath(): funkcja wykorzystująca algorytm Dijkstry znajdująca najkrótszą ścieżkę pomiędzy danymi wierzchołkami, a następnie wypisuje ją w jednym z dwóch trybów:
  1. tryb pierwszy, gdzie wypisywane są jedynie kolejne wierzchołki oddzielone znakami "=>"
  2. tryb drugi, gdzie wypisywane są również wagi między wierzchołkami. Poszczególne wierzchołki oddzielane są przez: "=(" + waga_krawędzi_między_wierzchołkami + ")>"

3 Różnice względem planowanej wersji

Wywołania poszczególnych funkcji różnią się szczegółami względem Specyfikacji Implementacyjnej.

4 Testy

Zrobione zostało kilka testów do pojedynczych modułów. Testy są uruchamiane przez program Make i są w większości automatyczne - porównują wyjście testu do wyjścia wzorcowego.

5 Co teraz zrobilibyśmy inaczej

1. Zastosowalibyśmy bardziej przejrzystą strukturę plików w projekcie - aktualnie wszystko jest w jednym katalogu src i się miesza.
2. Użylibyśmy specjalnie zmodyfikowanej tablicy zamiast samodzielnie napisanej kolejki priorytetowej - kolejka powodowała wiele problemów podczas sortowania tak że obecnie i tak musimy zamienić ją na tablicę.