Dokumentacja implementacyjna

Mateusz Charczuk

March 2022

1 Cel projektu

Celem projektu jest napisanie programu który wygeneruje graf o podanej liczbie kolumn i wierszy, posiadajacego losowe wartości na krawedzi. Dodatkowo program sprawdza czy graf jest spójny oraz zapisuje go do pliku o podanym formacie.

2 Zastosowane algorytmy

1. Algorytm Dijkstry - Algorytm służacy do znajdowania najkrótszej ścieżki miedzy wybranym wezłem a wszystkimi innymi, w grafie w którym ścieżki maja wartości nie ujemne. Przy odpowiedniej modyfikacji tego algorytmu oraz przy użyciu kolejki priorytetowej można sprawić, by wyliczał jedynie najkrótsza ścieżke miedzy dwoma punktami.
   Działanie algorytmu Dijkstry:
   1. Dla każdego wezła należy w zapisać w strukturze jego poprzednika
   2. Dodać do kolejki wage połaczenia.
   3. Z kolejki wybrać wezeł o najmniejszej wadze połaczenia.
   4. Nastepnie z pozostałych wezłów wybrać ten o najmniejszej wadze połaczenia. Kroki 3 i 4 powtarzać, do momentu aż kolejka bedzie pusta.

2. BFS - Algorytm służacy do przeszukiwaniu grafu w postaci drzewa. Przeszukanie polega na wybraniu wezła a nastepnie odwiedzeniu wszystkich sasiadujacych z nim wierzchołków.Do zastosowania tego algorytmu potrzebna jest implementacja kolejki priorytetowej.

   działanie algorytmu:
   1. Sprawdzanie czy wezeł jest tym który szukamy, jeżeli tak to zwrócenie go i wyjście z funkcji.
   2. Jeżeli sasiadujacy wezeł nie jest odwiedzony to oznacz go jako odwiedzony.
   Kroki 1 i 2 powtarzaj, do momentu gdy kolejka bedzie pusta.

3 Struktura danych

1. Dane z pliku / wygenerowane za pomoca parametrów wywołania sa poczatkowo przechowywane w tablicy struktur o liczbie elementów odpowiadajacej iloczynowi kolumn i wierszy grafu. Struktura ta zawiera 4 elementowa tablice intów w której przechowywane sa indeksy wierzchołków połaczonych z wierzchołkiem któremu odpowiada indeks w tablicy struktur. Kolejnym elementem który przechowuje struktura jest 4 elementowa tablica intów zawierajaca wagi połaczeń miedzy indeksem tablicy struktur i wartościa tablicy 4 elementowej wierzchołków (w ramach tej samej struktury) o tych samych indeksach.
   Dokładny wyglad definicji struktury:

   typedef struct graph
   int* n; // sasiedzi
   double* w; // wagi połaczeń
   graph;

2. Kolejka – liniowa struktura danych, w której nowe dane dopisywane sa na końcu kolejki, a z poczatku kolejki pobierane sa dane do dalszego przetwarzania.

3. Kolejki priorytetowe - Kolejka – liniowa struktura danych, w której nowe dane dopisywane sa w sposób uporzadkowany, w ten sam sposób sa potem pobierane do dalszego przetwarzania.

4. Opis funkcji kolejki oraz kolejki priorytetowej
   initQ - alokuje pamieć na kolejke i zwraca wskaźnik do struktury danych kolejka
   freeQ - zwalnia pamieć na kolejke
   Qfull - sprawdza czy kolejka jest pełna
   QEmpty - sprawdza czy kolejka jest pusta
   popQ - wyciaga ostatni element z kolejki (zwykła kolejka)
   popminQ - wyciaga najmniejszy element z kolejki (kolejka priorytetowa)
   addQ - dodaje element do kolejki
   countQ - liczy elementy kolejki

4 Opis modułów

1. main.c - Łaczy wszystkie moduły, przyjmuje parametry użytkownika przez opt i sprawdza ich poprawność za pomoca funkcji z pliku errorshandling.c, wywołuje funkcje: graph_gen(), graph_from_file(), bfs(), dijstra().

2. dijkstra.c - Odpowiada za realizacje algorytmu Dijkstry w programie

3. bfs.c - Odpowiada za sprawdzenie spójności grafu. Funkcja odpowiedzialna za obsługe algorytmu bfs przyjmuje adres pierwszego elementu tablicy struktur graph. Zwraca wartość int: 1 gdy graf jest spójny, 0 gdy graf jest nie spójny.

4. graphgen.c - zawiera funkcje odpowiadajace za generacje grafu z parametrów lini poleceń oraz wpisuje je do tablicy struktur graph. Funkcje które zawiera:

   graph_gen - generuje graf na podstawie parametrów wywołania.
   Przyjmuje ilość wierzchołków(r)(int), ilość krawedzi(c)(int), maksymalna wartość wagi(s)(double), minimalna wartość wagi(e)(double). Zwraca adres pamieci na wygenerowany graf(graf*). Jest wywoływana w mainie.

   graph_to_file - wypisuje do pliku wyjściowego podanego przez użytkownika graf w takim samym formacie jak plik wejściowy.
   Przyjmuje nazwe pliku(filename)(char*), adres pamieci struktury graf (g)(graph*), ilość wszystkich wierzchołków grafu (rc)(int), ilość wierszy grafu (r)(int) oraz ilość kolumn grafu (c)(int). Funkcja nic nie zwraca. Jest wywoływana w funkcji graph_gen

   free_graphgen - zwalnia pamieć zajmowana przez tymczasowe tablice(search graph) oraz (searched) potrzebne do jego generacji.
   Przyjmuje adres pamieci pierwszej tablicy tymczasowej(search_graf)(int **) oraz adres pamieci drugiej tablicy tymczasowej(searched)(int*) a także liczbe wierszy grafu(r)(int). Funkcja nic nie zwraca. Jest wywoływana w funkcji graph_gen

   neighbors_gen - odpowiada za sprawdzanie czy wierzchołek o indeksie 'x' i 'y' jest sasiadem wierzchołka o indeksie 'i' i 'j'. Jeżeli jest to zwraca wierzchołek o indeksie 'x' i 'y' (int), jeżeli nie to zwraca wartość -1(int).
   Przyjmuje adres pamieci pierwszej tablicy tymczasowej(search_graf)(int **) oraz adres pamieci drugiej tablicy tymczasowej(searched)(int*), indeks i(x)(int), indeks j(x)(int), indeks wiersza grafu (r)(int) oraz indeks kolumny grafu (c)(int). Jest wywoływana w funkcji graph_gen

   random_double - generuje losowa liczbe z zakresu od s(int) do e(int) z wyłaczeniem 0. Przyjmuje poczatek zakresu generacji(s)(double), oraz koniec zakresu generacji(e)(double). Zwraca wartość wygenerowanej liczby(double). Jest wywoływana w funkcji graph_gen.

   search_graph_gen - generuje tablice dwuwymiarowa imitujaca wyglad grafu który użytkownik chce wygenerować by umożliwić faktyczna generacje w

funkcji graph_gen.
Przyjmuje ilość wierszy grafu (r)(int) oraz ilość kolumn grafu (c)(int). Zwraca wygenerowana tablice dwuwymiarowa(int**). Jest wywoływana w funkcji graph_gen
searched_cleaner - zastepuje wszystkie pola tablicy buff zerami. Przyjmuje adres pamieci tablicy buff(buff)(int*)
Funkcja nic nie zwraca. Jest wywoływana w funkcji graph_gen.

graph_memory_allocator - alokuje pamieć dla tablicy struktur graph. Przyjmuje ilość wszystkich wierzchołków grafu. Zwraca adres pamieci na zaalokowana tablice struktur graph(graph*).

5. graph.c - zawiera funkcje odpowiedzialne za generacje grafu z pliku podanego przez użytkownika oraz funkcje wypisujaca wynik działania programu do pliku. A także funkcje zwalniajaca oraz alokujaca pamieć na graf

   funkcje: graph_from_file - odpowiada za wygenerowanie tablicy struktur graph na podstawie pliku wejściowego podanego przez użytkownika. Przyjmuje nazwe pliku wejściowego(fname)(char*) oraz adres pamieci ilości wszystkich wierzchołków grafu(rowcol)(int*). Zwraca adres pamieci wygenerowanej tablicy struktur graph (graph*). Funkcja jest wywoływana w mainie.

   graph_free - zwalnia pamieć zajmowana przez tablice struktur graph. Przyjmuje adres pamieci tablicy struktur graph(g)(graph*) oraz ilość wszystkich wierzchołków grafu(rowcol)(int). Funkcja nic nie zwraca. Jest wywoływana w mainie, funkcji graph_from_file oraz w funkcji EOFfaliture. Funkcja jest wywoływana w funkcji graph_from_file.

   graph_filler - zapełnia odpowiednie indeksy tablicy struktur graph wartościami -1(int).
   Przyjmuje adres pamieci tablicy struktur graph(g)(graph*), indeks (j)(int) oraz indeks (k)(int). Funkcja jest wywoływana w funkcji graph_from_file.

   buff_cleaner - wypełnia wszystkie pola tablicy buff(char*) wartościa ".
   Przyjmuje adres pamieci tablicy (buff)(char*) oraz indeks do którego ma zapełniać ta tablice (i)(int). Funkcja jest wywoływana w funkcji graph_from_file.

6. errorshandling.c - Zawiera funkcje obsługujace błedy formatu danych w pliku oraz danych podanych przez użytkownika w linii wywołania programu. Zawiera funkcje:

is_good_i - sprawdza czy podana tablica buf jest liczba typu int (czy nie zwiera w sobie liter oraz inych znaków oprócz liczb od 0 do 9) a także czy jest to liczba ujemna. Jeżeli nie jest to liczba to zmienia wartość err na 1 oraz zwraca -1(int), gdy liczba jest ujemna to zmienia wartość err na 2, oraz zwraca -1(int). Jeżeli wszystko jest dobrze to funkcja zwraca tablice buf przekonwertowana na int funkcja atoi.
Przyjmuje adres pamieci tablicy buff(char*) oraz adres pamieci zmiennej (err)(int*). Funkcja jest wywoływana w mainie oraz funkcji graph_from_file.

is_good_i - sprawdza czy podana tablica buf jest liczba typu double (czy nie zwiera w sobie liter oraz inych znaków oprócz liczb od 0 do 9 oraz znaku .) a także czy jest to liczba ujemna. Jeżeli nie jest to liczba to zmienia wartość err na 1 oraz zwraca -1(double), gdy liczba jest ujemna to zmienia wartość err na 2, oraz zwraca -1(double). Jeżeli wszystko jest dobrze to funkcja zwraca tablice buf przekonwertowana na double funkcja atof.
Przyjmuje adres pamieci tablicy buff(char*) oraz adres pamieci zmiennej (err)(int*). Funkcja jest wywoływana w mainie oraz funkcji graph_from_file.

is_file_good - sprawdza czy podana przez użytkownika nazwa pliku jest poprawna (program potrafi ja wczytać).
Przyjmuje nazwe pliku(inpf)(char*). Zwraca 1 gdy nazwa pliku jest niepoprawna, 0 gdy nazwa pliku jest poprawna. Funkcja jest wywoływana w mainie.

check_error_file_format - sprawdza errorflagi i na podstawie ich wartości wyświetla odpowiednie komunikaty błedów szerzej opisane w punkcie Sytuacje wyjatkowe. w przypadku nei wykrycia żadnego błedu (wartości wrorflagów od 1 do 3 sa zerami) nie modyfikuje wartości errorflag4, w przeciwnym wypadku ustawia wartość errorflaga4 na 1.
Przyjmuje wartości errorflagów oraz adres pamieci errorflaga4, (error_flag_1)(int), (error_flag_3)(int), (error_flag_4)(int), (error_flag_4)(int*). Funkcja nic nie zwraca. Funkcja jest wywoływna w funkcji graph_from_file

EOFfaliture - sprawdza czy podany do funkcji char nie jest znakiem specjalnym EOF. W przypadku gdy nie jest, nie dzieje sie nic. W przeciwnym przypadku funkcja wyświetla komunikat błedu, zwalnia pamieć oraz kończy działanie programu z błedem.
Przyjmuje ilość wszystkich wierzchołków grafu(rc)(int), adres pamieci tablicy struktur graph(g)(graph*), adres pamieci tablicy (tmp)(char*), uchwyt na plik który trzeba zamknać (in)(FILE *).Funkcja nic nie zwraca. Jest wywoływana w funkcji graph_from_file.

Makefile - umożliwia szybka kompilacje programu jedna komenda "make all".

5 Sytuacje wyjatkowe

- Błedy w pliku wejściowym:

  1. Źle skonstruowana pierwsza linijka pliku komunikat:
     Format pliku jest niepoprawny(1 linijka)
  2. Enter postawiony w złym miejscu pliku uniemożliwiajacy jego poprawne odczytanie komunikat:
     Format pliku jest niepoprawny(entery)
  3. znak : postawiony w złym miejscu pliku uniemożliwiajacy jego poprawne odczytanie komunikat:
     Format pliku jest niepoprawny(brakuje : przed waga)
  4. Wartości które powinny być liczbami w odpowiednich miejscach w pliku nimi nie sa komunikat:
     Format pliku jest niepoprawny: wartości podane przez użytkownika nie sa liczbami
  5. Liczby podane w pliku sa ujemne komunikat:
     Format pliku jest niepoprawny: liczby podane przez użytkownika sa ujemne
  6. Plik kończy sie nie w tym miejscu co trzeba lub dotychczasowo wczytane wartości nie sa wystarczajace by zapisać graf o podanych rozmiarach komunikat:
     Format pliku jest niepoprawny(EOF)

- Błedy w parametrze wywołania:

  1. Gdy użytkownik nie poda żadnych argumentów wywołania pojawi sie komunikat:
     Nie podano żadnych parametrów wywołania
  2. Gdy użytkownik poda nazwe pliku który nie istnieje badź nie da sie go otworzyć pojawi sie komunikat:
     KRYTYCZNY BŁAD: Podany plik wejściowy nie istnieje
  3. Gdy użytkownik jako argument który powinien być liczba poda nie liczbe wyświetli sie komunikat:
     KRYTYCZNY BŁAD: Wartość podana przez użytkownika nie jest liczba
  4. Gdy użytkownik jako argument który powinien być liczba dodatnia poda nie liczbe dodatnia wyświetli sie komunikat:
     KRYTYCZNY BŁAD: Wartość podana przez użytkownika jest ujemna
  5. gdy użytkownik poda argument świadczacy o checi wygenerowania grafu przez program z argumentów linii polecenia a nie poda wystarczajacej ich ilości pojawi sie komunikat:
     Nie podano wystarczajacej ilości argumentów aby wygenerować graf