laboratorium: zadanie 11 dodatkowe *termin: 11–14 czerwca 2018 r.*

JĘZYK PROGRAMOWANIA C++

GRAFY

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Graf to podstawowa struktura danych służąca do modelowania i badania skomplikowanych relacji między jakimiś obiektami. W uproszczeniu graf to zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.

Wierzchołki grafu reprezentujące jakieś obiekty są numerowane (od 0 do $n-1$), natomiast krawędzie obrazują relacje między takimi obiektami (liczba krawędzi m w grafie prostym bez cykli i krawędzi wielokrotnych może się wahać od 0 do $\frac{n(n-1)}{2}$). Wierzchołki należące do krawędzi nazywane są jej końcami. Krawędzie mogą mieć wyznaczony kierunek; graf zawierający takie krawędzie nazywany jest *grafem skierowanym* lub *digrafem*. Krawędź grafu może posiadać wagę, to znaczy przypisaną liczbę, która określa na przykład odległość (koszt przejścia) między wierzchołkami; graf zawierający takie krawędzie nazywamy *grafem ważonym*.

Zadanie.

Zdefiniuj klasę abstrakcyjną `Graf` reprezentującą graf prosty z dodatnimi wagami na krawędziach (domyślna waga krawędzi to 1.0) . Następnie zdefiniuj dwie imlementacje grafu: `GrafGesty` w postaci macierzy sąsiedztwa oraz `GrafRzadki` w postaci list sąsiadów. Grafy te mają mieć zaimplementowaną semantykę kopiowania i przenoszenia.

W grafie zbiór wierzchołków $V = \{0, 1, \ldots, n-1\}$ ma być określony na etapie konstrukcji i nie będzie się zmieniać do końca egzystencji obiektu. Krawędzie w takim grafie można jednak dynamicznie dokładać, usuwać lub zmieniać im wagę. Zaprogramuj także zaprzyjaźnione operatory do odczytania grafu ze stumienia wejściowego i zapisania grafu do strumienia wyjściowego. Uzupełnij funkcjonalność grafu o iteratory umożliwiające przeglądanie wierzchołków i krawędzi.

Następnie stwórz bibliotekę dynamiczną zawierającą wszystkie definicje grafów.

Na koniec napisz program, który przetestuje obie implementacje grafu, implementując *algorytm Dijkstry*, znajdujący najkrótszą ścieżkę pomiędzy zadaną parą wierzchołków. Do projektu z programem testującym dołącz stworzoną wcześniej bibliotekę dynamiczną z grafami.

Uwaga.

Implementując graf wykorzystaj kolekcje z biblioteki standardowej.

Elementy w programie, na które należy zwrócić szczególną uwagę.

- Definicja abstrakcyjnej klasy dla grafu.
- Definicje klas implementujących graf rzadki i gęsty.
- Użycie kolekcji standardowych w implementacjach grafów.
- Kopiowanie i przenoszenie grafów.
- Implementacja algorytmu Dijkstry.
- Utworzenie biblioteki dynamicznej.
- Podział programu na pliki nagłówkowe i źródłowe.