

## Zadanie projektowe nr 2

### **Badanie efektywności algorytmów grafowych w zależności od rozmiaru instancji oraz sposobu reprezentacji grafu w pamięci komputera.**

Należy zaimplementować oraz dokonać pomiaru czasu działania wybranych algorytmów grafowych rozwiązujących pewne problemy. Szczegółowe informacje znajdują się w punkcie **Zadania i oceny**.

Algorytmy te należy zaimplementować dla obu poniższych reprezentacji grafu:

- ❖ reprezentacja macierzowa (macierz incydencji),
- ❖ reprezentacja listowa (lista sąsiedztwa).

#### **1. Założenia**

- ❖ wszystkie struktury danych powinny być alokowane dynamicznie; w przypadku tablic powinny zajmować jak najmniej miejsca – powinny być relokowane przy dodawaniu/usuwaniu kluczy,
- ❖ koszt krawędzi jest liczbą całkowitą,
- ❖ po zaimplementowaniu każdego z algorytmów dla obu reprezentacji należy dokonać pomiaru czasu działania algorytmów w zależności od rozmiaru grafu oraz jego gęstości (liczba krawędzi w stosunku do liczby wierzchołków). Badania należy wykonać dla 5 różnych (reprezentatywnych) liczb wierzchołków  $V$  oraz następujących gęstości grafu: 25%, 50%, 75% oraz 99%. Przy czym w programie musi istnieć możliwość ustawienia gęstości w zakresie 1 do 100% (co 1). Dla każdego zestawu: *reprezentacja, liczba wierzchołków i gęstość*, należy wygenerować po 100 losowych instancji, zaś w sprawozdaniu umieścić wyniki uśrednione,

dotatkową funkcją programu musi być możliwość sprawdzenia poprawności zaimplementowanych operacji i zbudowanej struktury (szerzej w **Sprawdzenie poprawności zbudowanych struktur/operacji**),

- ❖ dokładnego pomiaru czasu w systemie Windows dokonujemy z wykorzystaniem funkcji *QueryPerformanceCounter* lub `std::chrono::high_resolution_clock`. Opis na stronie:

<http://cpp0x.pl/forum/temat/?id=21331>,

- ❖ dopuszczalnymi językami programowania są języki kompilowane do kodu natywnego (np. C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Python),
- ❖ używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- ❖ z gotowych bibliotek np. STL, Boost lub innych można korzystać jedynie w uzasadnionych przypadkach; uzasadnienie musi zostać zaakceptowane przez prowadzącego,

- ❖ implementacja projektu powinna być wykonana w formie jednego programu,
- ❖ kod źródłowy powinien być komentowany.

## 2. Sprawdzenie poprawności

Sprawdzenie poprawności stworzonej struktury/operacji obejmuje:

- ❖ wczytanie struktury grafu z pliku tekstowego. Plik zawiera opis poszczególnych krawędzi według wzoru: początek krawędzi, koniec krawędzi oraz waga/przepustowość/koszt. Struktura pliku jest następująca:
  - w pierwszej linii zapisana jest liczba krawędzi oraz liczba wierzchołków (rozdzielone spacją),
  - wierzchołki numerowane są w sposób ciągły od zera,
  - w kolejnych liniach znajduje się opis krawędzi (każda krawędź w osobnej linii) wg podanego wzoru,
  - dla problemu MST pojedynczą krawędź traktuje się jako nieskierowaną, natomiast dla algorytmów najkrótszej drogi i maksymalnego przepływu jako skierowaną,
- ❖ losowe wygenerowanie grafu (jako dane podaje się liczbę wierzchołków oraz gęstość w %). Graf z pliku i wygenerowany losowo zajmują tę samą zmienną (czyli ostatnia operacja generowania lub wczytywania z pliku nadpisuje poprzednią),
- ❖ możliwość wyświetlenia na ekranie wczytanego lub wygenerowanego grafu w formie reprezentacji listowej i macierzowej,
- ❖ uruchomienie algorytmu dla obu reprezentacji i wyświetlenie wyników na ekranie. Dla problemu najkrótszej drogi w grafie i maksymalnego przepływu musi być możliwość podania wierzchołka początkowego i końcowego.

## 3. Menu programu

Wybór problemu (algorytmu)

Wybór reprezentacji

Wczytaj (z pliku)

Wygeneruj graf (losowo o zadanej gęstości)

Wyświetl (w określonej formie)

Wcięcia oznaczają poziomy menu.

## 4. Sprawozdanie

Sprawozdanie powinno zawierać:

- ❖ krótki wstęp zawierający oszacowanie złożoności obliczeniowej poszczególnych problemów na podstawie literatury,

- ❖ plan eksperymentu czyli założenia co do wielkości struktur, sposobu generowania ich elementów, sposobu pomiaru czasu, itp.
- ❖ opis metody losowego generowania grafu. Sposób powinien zapewnić spójność grafu,
- ❖ wyniki – należy przedstawić w tabelach oraz w formie wykresów (oś X : liczba wierzchołków, oś Y : czas działania) dla każdego problemu osobno (oddzielnie MST, najkrótsza droga w grafie, maksymalny przepływ). Dla każdego problemu należy przygotować następujące wykresy:
  - wykresy typ1 (osobne wykresy dla każdej reprezentacji grafu) – w formie linii (połączonych punktów), których parametrem jest gęstość grafu i typ algorytmu (Kruskal/Prim lub Dijkstra/Bellman) – czyli  $4 \times 2 = 8$  linii na rysunek,
  - wykresy typ2 (osobne wykresy dla każdej gęstości grafu) – w formie linii których parametrem jest typ algorytmu i typ reprezentacji grafu (czyli 4 linie na każdy rysunek).
  - dla wszystkich wykresów: czas wykonania algorytmu (oś Y) w funkcji liczby wierzchołków (oś X).
- ❖ wnioski dotyczące efektywności poszczególnych struktur. Wskazać (jeśli są) przyczyny rozbieżności pomiędzy złożonościami teoretycznymi a uzyskanymi eksperymentalnie,
- ❖ załączony kod źródłowy w formie elektronicznej (cały projekt wraz z wersją skompilowaną programu) oraz wydrukowane sprawozdanie.

## 6. Zadania i oceny

Ćwiczenie polega na wykonaniu trzech zadań. Zadanie to implementacja i badanie jednego algorytmu. Obramowania określają algorytmy do wyboru w ramach zadania. Minimalne wymagania do uzyskania określonych ocen określone są w tabeli umieszczonej poniżej. Każde zadanie wykonane ponad minimum daje (nie gwarantuje) szanse na podwyższenie oceny.

3,0	3,5	4,0	4,5	5,0	algorytm	
D					Dijkstra	
BF	BF				Bellman-Ford	
P	P	P			Prim	<i>ls</i> - lista sąsiedztwa
K	K	K			Kruskal	<i>kb</i> - kopiec binarny
	<i>Dkb</i>				Dijkstra: <i>kb</i>	<i>kF</i> - kopiec Fibonacciego
		<i>DkF</i>	<i>DkF</i>		Dijkstra: <i>kF</i>	
		<i>PkF+ls</i>	<i>PkF+ls</i>	<i>PkF+ls</i>	Prim: <i>kF +ls</i>	
			B	B	Boruvka	
			A*	A*	A*	
				FF	Ford-Fulkerson	
				EK	Edmonds-Karp	

Przykład. Ocena 4,5; zadania do wykonania: *DkF*, *PkF+ls* albo B, A\*

## 7. Termin złożenia projektu

Zadanie projektowe nr 1 należy złożyć do dnia **09 maja 2017r.** Każdy rozpoczęty tydzień spóźnienia oznacza obniżenie oceny uzyskanej z zadania projektowego o jeden stopień. Należy również pamiętać, że ocena końcowa z projektu (jako formy zajęć) liczona będzie jako średnia z sumy ocen z poszczególnych zadań projektowych z uwzględnieniem spóźnień. Oznacza to, że aby uzyskać ocenę dostateczną z projektu (formy zajęć) suma z trzech zadań projektowych z uwzględnieniem ewentualnych spóźnień, musi wynieść co najmniej 9,0. W przeciwnym przypadku projekt (forma zajęć) zostanie oceniony na ocenę niedostateczną.

## 8. Forma złożenia projektu

Patrz [zjo.iiar.pwr.wroc.pl/sdizo.html](http://zjo.iiar.pwr.wroc.pl/sdizo.html) **składanie projektów** (pdf)

## 9. Źródła

- [1] T. H. Cormen (i inni) , Wprowadzenie do algorytmów, WNT, Warszawa, 1997
- [2] A. Drozdek, C++ Algorytmy i struktury danych, Helion, Gliwice, 2001
- [3] [http://eduinf.waw.pl/inf/alg/001\\_search/index.php](http://eduinf.waw.pl/inf/alg/001_search/index.php)

**Plagiaty będą surowo karane !!!**