Algorytmy optymalizacji dyskretnej

LABORATORIUM 1

Podstawowe algorytmy grafowe – przypomnienie

Termin wysyłania (MS Teams): **21 października 2021 godz. 15:14**

Zadanie 0.

Zapoznaj się z artykułem [Joh02] na temat eksperymentalnego badania algorytmów (tekst artykułu dostępny jest m.in. na stronie https://dimacs.rutgers.edu/archive/Challenges/TSP/papers/experguide.pdf). Podczas testowania algorytmów i przeprowadzania eksperymentalnej analizy ich własności stosuj się do omówionych tam dobrych praktyk i staraj się unikać wspomnianych pułapek.

W kontekście zagadnień, które będą omawiane na kursie (oraz przyszłych zadań na laboratorium), warto przeczytać także rozdział 18 (*Computational Testing of Algorithms*) z podręcznika [AMO93].

W zadaniach 1–4 wybierz odpowiednią reprezentację grafów (pozwalającą na uzyskanie możliwie efektywnych implementacji) i przetestuj swoje implementacje dla grafów o liczbie wierzchołków rzędu 10, 100, 1000 i 10000 (w trakcie oddawania listy należy m.in. zaprezentować otrzymane rezultaty wcześniej przeprowadzonych testów). Definicja grafu $G = (V, E)$ powinna być przekazywane na standardowym wejściu w następujący sposób (kolejno wczytywane są):

- jednoliterowa flaga mówiąca, czy graf jest skierowany (`D`) czy nieskierowany (`U`),
- liczba wierzchołków $n = |V|$ (przyjmujemy, że wierzchołki są etykietowane kolejnymi liczbami naturalnymi ze zbioru $\{1, \ldots, n\}$),
- liczba krawędzi $m = |E|$,
- kolejno m definicji krawędzi postaci `u v`.

Zadanie 1. [2 pkt]

Zaimplementuj algorytmy przeszukiwania grafów wgłąb i wszerz (DFS i BFS; patrz np. rozdział 3.4 w [AMO93], rozdziały 3.2, 3.3 i 4.2 w [DPV06] lub rozdziały 22.2 i 22.3 w [CLRS09]). Algorytm na wyjściu powinien wypisywać kolejność, w której wierzchołki były odwiedzane oraz – po podaniu odpowiedniego parametru wywołania – zwracać drzewo przeszukiwania (DFS/BFS tree). Program powinien obsługiwać przypadki grafów skierowanych i nieskierowanych.

Zadanie 2. [1 pkt]

Zaimplementuj algorytm sortowania topologicznego dla grafów skierowanych z wykrywaniem istnienia skierowanego cyklu (patrz np. rozdział 3.4 w [AMO93], rozdział 3.3.2 w [DPV06] lub rozdział 22.4 w [CLRS09]). Algorytm na wyjściu powinien zwracać listę wierzchołków w porządku topologicznym.

Zadanie 3. [1 pkt]

Zaimplementuj algorytm, który dla podanego na wejściu grafu skierowanego $G = (V, E)$ zwróci jego rozkład na silnie spójne składowe (patrz np. rozdział 3.4 w [DPV06] lub 22.5 w [CLRS09]). Algorytm powinien działać w czasie $O(|V| + |E|)$.

Zadanie 4. [1 pkt]

Zaimplementuj efektywny algorytm, który dla podanego na wejściu grafu $G = (V, E)$ (skierowanego lub nieskierowanego, niekoniecznie spójnego) zwraca informację, czy G jest grafem dwudzielnym i jeśli tak, to zwraca rozbicie V na dwa podzbiory V_0 i V_1 takie, że jeśli $(u, v) \in E$, to $u \in V_i$ i $v \in V_{1-i}$, $i \in \{0, 1\}$. Jaką złożoność ma zaimplementowany algorytm?

Literatura

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., USA, 1993.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[DPV06] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Vazirani. *Algorithms*. McGraw-Hill, Inc., USA, 1st edition, 2006.

[Joh02] David S. Johnson. A Theoretician's Guide to the Experimental Analysis of Algorithms. In D.S. Johnson M.H. Goldwasser and C.C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, January 2002.