Algorytmy i struktury danych

Laboratorium - lista 5

Termin wysłania: 2025-06-10

Zadanie 1. [20 p.]

1. Zaimplementuj program, który dla danego n generuje n-wierzchołkowy graf **pełny** o losowych wagach krawędzi. Wagi krawędzi mają być losowane z przedziału $(0, 1)$ według rozkładu jednostajnego.
2. Zaimplementuj algorytmy Prima i Kruskala do wyznaczania minimalnego drzewa rozpinającego (MST) dla grafów wygenerowanych przez program opisany w pierwszym punkcie.
3. Wyznacz eksperymentalnie średnie koszty czasowe obu algorytmów na losowych grafach wygenerowanych przez program z punktu pierwszego. Testy wykonaj dla $n = [nMin, nMax]$ z krokiem $step$ powtarzając eksperyment dla każdej wielkości grafu rep razy. Dobierz wartości $nMin$, $nMax$, $step$ i rep na tyle duże/małe aby można było zauważyć występujące wzorce w otrzymanych wynikach, ale również aby eksperymenty wykonały się w 'sensownym' czasie (przed terminem oddania zadania). Zwizualizuj otrzymane dla obu algorytmów wyniki na wspólnym wykresie .

Zadanie 2. [20 p.]

Zaimplementuj algorytm, który dla danego *drzewa* (tj. grafu spójnego bez cykli) z ustalonym wierzchołkiem v jako *korzeniem,* generującym pewną informację, wyznaczy dla każdego wierzchołka kolejność, w jakiej ma on informować swoje dzieci tak, żeby liczba *rund* potrzebna do dotarcia informacji do wszystkich wierzchołków grafu była jak najmniejsza. W jednej rundzie wszystkie wierzchołki posiadające informację mogą ją przekazać jednemu ze swoich dzieci w drzewie. (Dokładny opis: Lista 6 na ćwiczenia Zadanie 5.)

Wykonaj eksperymentalnie 'average case analysis' (średnia, max, min) liczby rund potrzebnych do rozesłania wiadomości po całym grafie z losowego wierzchołka startowego. Wyniki przedstaw na wykresach zależności od liczby wierzchołków drzewa.

Jako dane wejściowe do eksperymentów wykorzystuj drzewa MST generowane przez programy z poprzedniego zadania.

Zadanie 2. [20 p.]

Zaimplementuj kopiec dwumianowy:
Dla $n = 500,$ wykonaj następujący eksperyment:

1. Utwórz dwa puste kopce H_1 i H_2 (operacje `Make-Heap`).
2. Do każdego z tych dwóch kopców wstaw losowy ciąg elementów długości n operacjami `Heap-Insert`.

3. Scal H_1 i H_2 w jeden kopiec H operacją `Heap-Union`.
4. Na kopcu H wykonaj $2n$ operacji `Extract-Min`. (Sprawdzaj, czy ciąg usuwanych elementów jest posortowany i czy dokładnie po ostatniej operacji kopiec staje się pusty.)

Dla każdej wykonanej operacji policz liczbę wykonanych porównań między kluczami. Następnie wykonaj "historyczny" wykres przedstawiający liczbę porównań c_i wykonanych w i-tej wykonanej operacji.
Wykonaj po 5 eksperymentów z wykresami, aby sprawdzić czy występują duże różnice dla różnych wylosowanych ciągów wejściowych.
Wykonaj także eksperymenty, dla $n \in \{100, 200, \cdots, 10000\}$, oraz wykres, na którym przedstawiona jest zależność między n a **łączną** liczbą porównań we wszystkich operacjach eksperymentu przy danej wartości n podzieloną przez n (średni koszt operacji).

Literatura

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