Wiktor Gonczaronek 127191

Poznań 22. 05. 2017 r.

Algorytmy i struktury danych

Badanie algorytmów znajdowania cyklu Eulera i Hamiltona

1 Cel badań

Celem badań było przebadanie algorytmów znajdowania cyklu Eulera i cyklu Hamiltona.

2 Środowisko testowe

Badania były wykonywane na komputerze z procesorem i5 2,7 GHz, 8GB RAM i systemem operacyjnym OS X 10.12 z kernelem Darwin Kernel Version 16.4.0. Na czas trwania testów wyłączono dostęp do sieci. Algorytmy implementowano w języku Python w wersji 3.5.2.

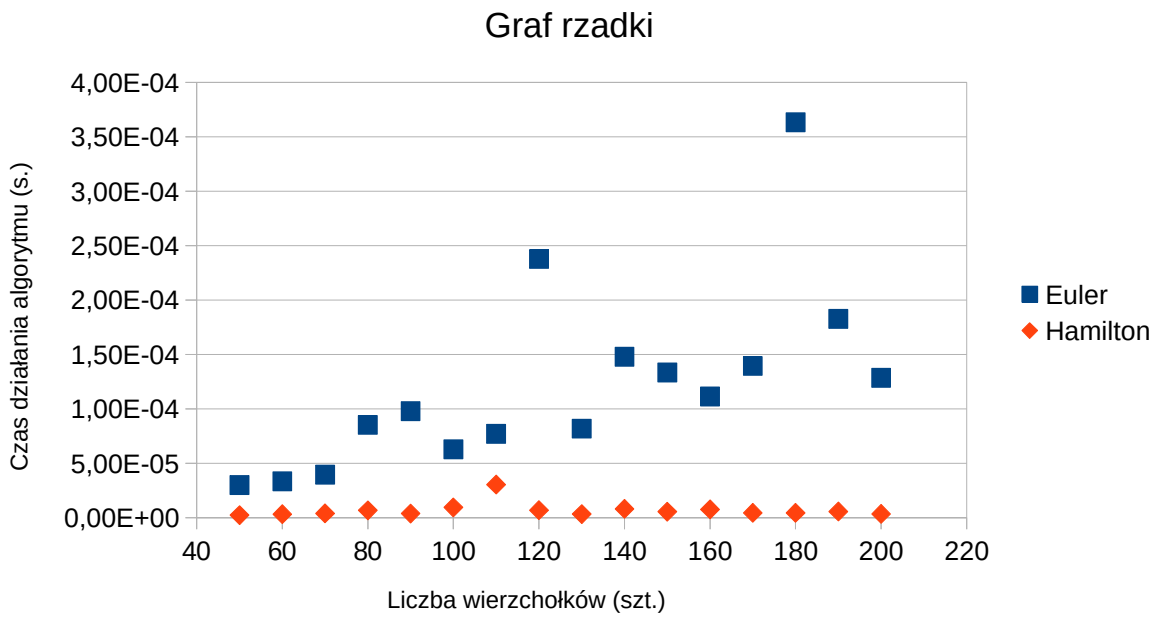
3 Metodologia

Utworzono spójne grafy o małym i dużym nasyceniu wierzchołkami. Ich rozmiary wynosiły od 50 do 200 z krokiem co 10 oraz od 10 do 28 dla znajdowania wszystkich cykli Eulera w grafie. Każdy algorytm przebadano na danym grafie 5-krotnie, po czym wyciągnięto średnią z pomiarów.

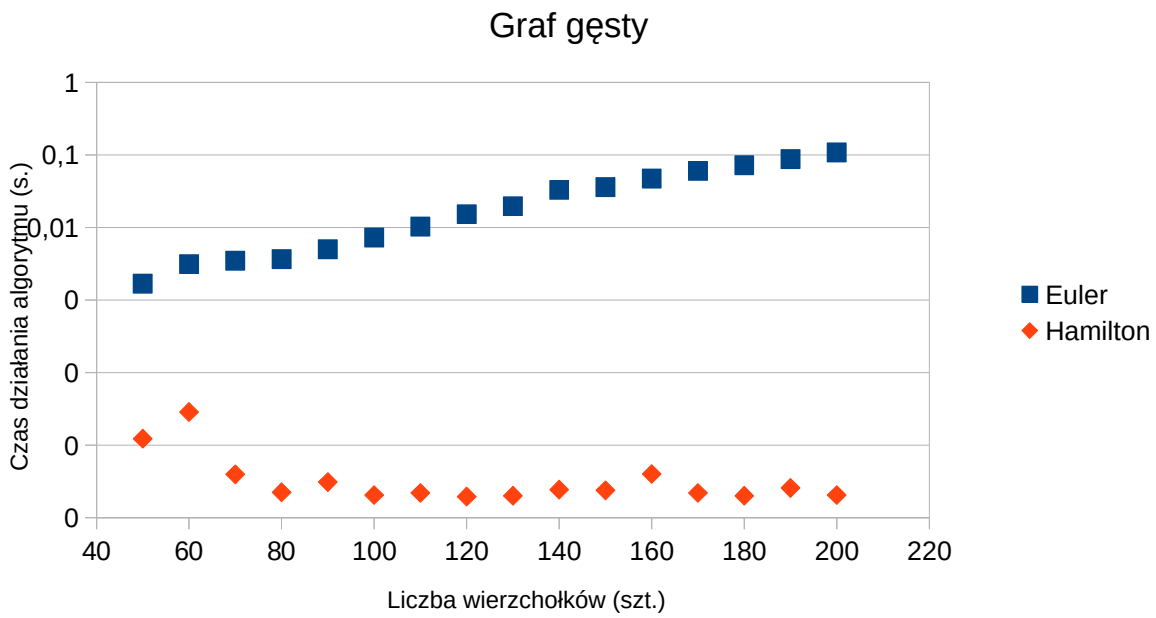
4 Badania

Przebadanie algorytmów przysporzyło kilku problemów. Python ma wbudowane ograniczenie maksymalnej rekurencji do 1000 zagłębień, konieczne było zwiększenie tego limitu do 50 000. Przy dużych rozmiarach danych działanie było przerywane przez system operacyjny ze względu na naruszenie ochrony pamięci. Wspomniane wyżej rozmiary danych były maksymalnymi, dla których udało się przeprowadzić badania.

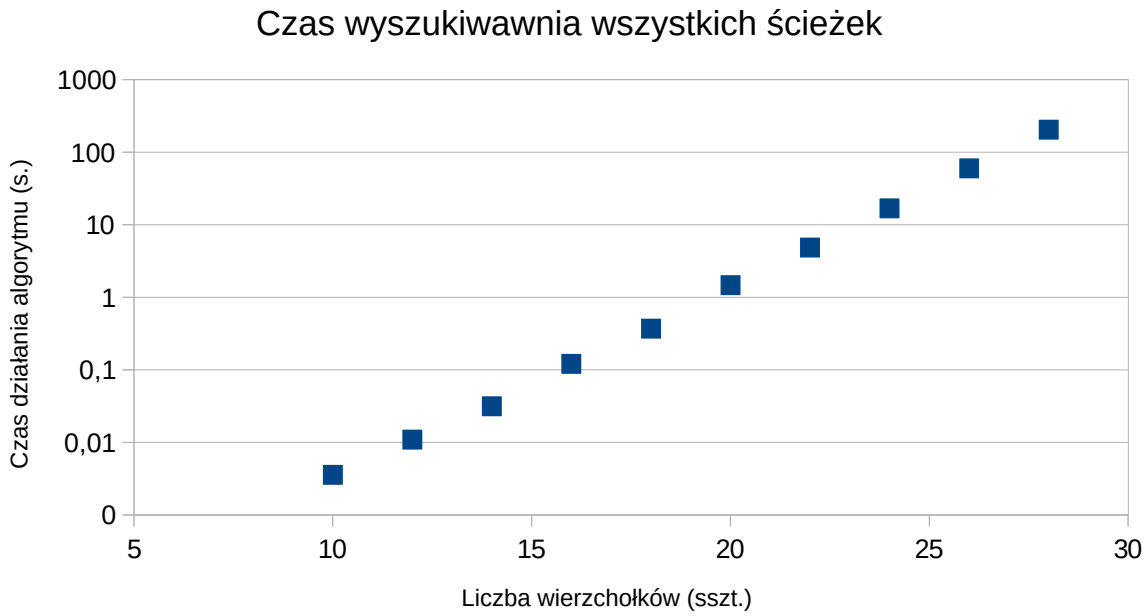
4.1 Porównanie algorytmów na rzadkich grafach.



4.2 Graf gęsty



5 Badanie czasu działania algorytmu znajdującego wszystkie ścieżki Hamiltona w grafie



6 Analiza wyników i wnioski.

Problem znajdowania cykli Hamiltona i Eulera należy do problemów NP-zupełnych, co oznacza, że nie są znane algorytmy znajdujące owe cykle w czasie wielomianowym. O ile faktycznie działanie algorytmów dla grafów gęstych trwało dłużej, o tyle kształt uzyskanych krzywych sugeruje, że badanie nie było prowadzone w poprawny sposób. Poprawność działania obu algorytmów została zbadana podczas zajęć, należy szukać przyczyn takiego stanu rzeczy gdzie indziej. Można zauważyć, że w przypadku utworzenia struktury danych takiej, że nie jest wymagane częste

wycofywanie się, algorytmy będzie działały w czasie zbliżonym do liniowego (bo wystarczy wejść do pierwszego napotkanego sąsiada, co w praktyce będzie oznaczało, że algorytm będzie działał podobnie jak w przypadku przeglądania listy). Konieczne jest zatem przeanalizowanie algorytmu tworzącego grafy pod kątem tego problemu. Kolejnym problemem jest zastosowanie rekurencji, które dla dużych danych wywoływało błąd SEGFAULT. Naruszenie ochrony pamięci nie występowałoby w przypadku przerobienia algorytmów na wersje iteracyjne. Można byłoby zbadać większy zakres wartości i zastosować większy krok uzyskując bardziej wiarygodne wyniki.