Optymalizacja Algorytmu Brandesa
Programowanie Współbieżne

Szymon Pajzert

9 stycznia 2017

Cel

Dokonanie optymalizacji współbieżnej implementacji algorytmu Brandesa, przedstawionego dokładniej tutaj

Dane testowe

Do testowania wydajności użyte zostały dane wiki-vote-sort.txt powstałe z posortowania po pierwszej i drugiej kolumnie danych wiki-vote.txt.

Wierzchołki	7115
Krawędzie	103689
Wierzchołki w największym WCC	7066 (0.993)
Krawędzie w największym WCC	103663 (1.000)
Wierzchołki w największym SCC	1300 (0.183)
Krawędzie w największym	39456 (0.381)
Średni clustering coefficient	0.1409
Liczba trójkątów	608389
Średnica	7

Dokonane optymalizacje

a. Nieużywanie mutexów - są drogie oraz istnieją inne metody synchronizacji - w tym wypadku atomics. Do incrementowania ich wartości użyłem `compare_exchange_weak` - ale tylko jeśli wynik w `delta` jest dodatni.

b. Zamiana numerów wierzchołków - pozwala na przetrzymywanie grafu w wektorze, dając mu stały lookup, przy stosunkowo znikomym narzucie wczytywania

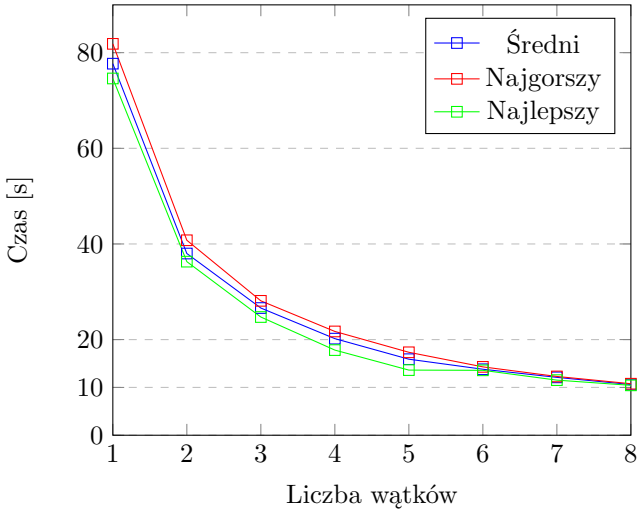
c. Używanie `unordered_map` - w testach wydajnościowych hashmapy dawały 3 razy większą wydajność w porównaniu z mapami opartymi o drzewa.

d. Użycie flagi `gcc -O3`

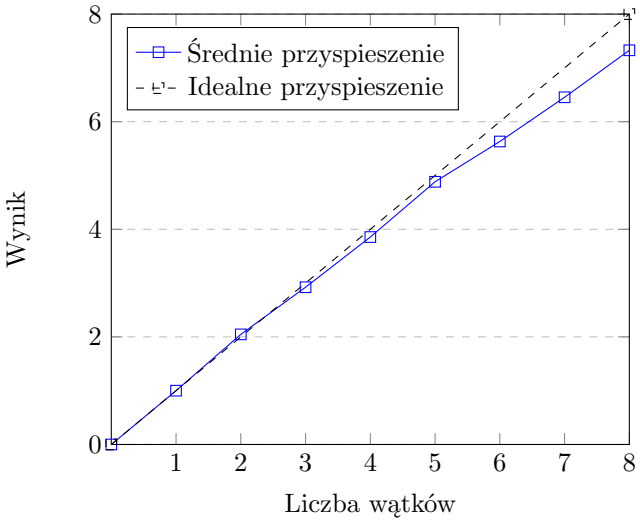
Benchmark

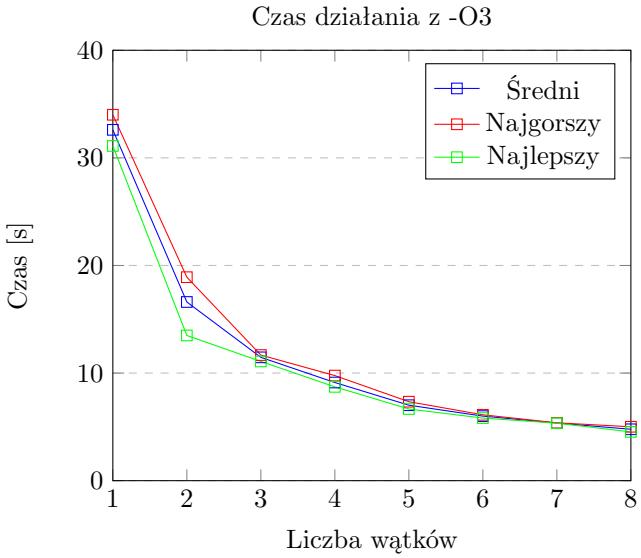
Testy zostały wykonane na serwerze students.mimuw.edu.pl.

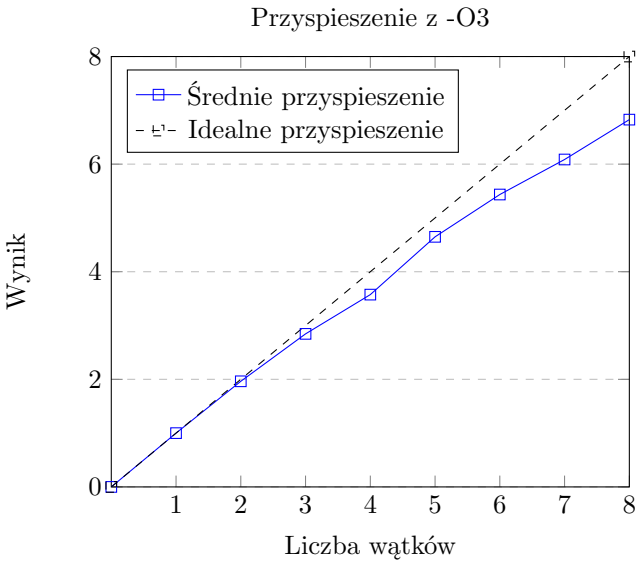
Czas działania bez -O3



Przyspieszenie bez -O3




Czas działania z -O3
Czas [s]
Liczba wątków
Średni
Najgorszy
Najlepszy
0
10
20
30
40
1
2
3
4
5
6
7
8


Przyspieszenie z -O3
Wynik
Liczba wątków
Średnie przyspieszenie
Idealne przyspieszenie
0
2
4
6
8
1
2
3
4
5
6
7
8