Politechnika Wrocławska

Projektowanie Algorytmów i Metod Sztucznej Inteligencji

Imię i nazwisko prowadzącego	dr inż. Łukasz Jeleń
Termin zajęć	środa, $11^{15} - 13^{00}$
Data oddania sprawozdania	7 maja 2019

Temat projektu

Grafy

Imię i Nazwisko	**Indeks**
Anna Bielecka	241159

1 Wstęp

Tematem projektu było zbadanie efektywności algorytmu Dijkstry w zależności od sposobu reprezentacji grafu, ilości wierzchołków oraz jego gęstości. Test efektywności algorytmu należało przeprowadzić mierząc czas jego wykonania dla grafów z 5 różnymi ilościami wierzchołków (10, 50, 100, 500, 1000) oraz dla następujących gęstości grafu: 25%, 50%, 75% i 100%. Dla każdego zestawu parametrów: reprezentacja grafu, liczba wierzchołków i gęstość należało wygenerować po 100 losowych instancji, natomiast w sprawozdaniu umieścić wyniki uśrednione.

Do pomiaru czasu wykonania algorytmu w milisekundach wykorzystano funkcję:

```
LARGE_INTEGER getTimer()
{
  LARGE_INTEGER time;
  DWORD_PTR oldmask = SetThreadAffinityMask(GetCurrentThread(), 0);
  QueryPerformanceCounter(&time);
  SetThreadAffinityMask(GetCurrentThread(), oldmask);
  return time;
}

QueryPerformanceFrequency(&freq);
start = getTimer();
Dijkstra(graphAdjList, naglowek->wierzcholekStartowy, false);
stop = getTimer();
czas = 1000 * (stop.QuadPart - start.QuadPart) / (double)freq.QuadPart;
```

2 Opis algorytmu Dijkstry

Algorytm Dijkstry służy do rozwiązywania problemu najkrótszych ścieżek z jednym źródłem w ważonym grafie $G = (V, E)$, w przypadku gdy wagi wszystkich krawędzi są nieujemne.

W przedstawionej implementacji wykorzystywana jest kolejka priorytetowa oparta na kopcu (typu min). Algorytm polega na wielokrotnym powtarzaniu następujących operacji: pobraniu z kolejki wierzchołka u o najmniejszym dystansie od wierzchołka startowego i wykonaniu relaksacji krawędzi wychodzących z wierzchołka u. Kolejka priorytetowa zawiera wierzchołki nie należące do bieżącego MST, pobranie wierzchołka z kolejki powoduje dodanie go do niego.

Złożoność obliczeniowa algorytmu gdy graf jest zaimplementowany za pomocą listy sąsiedztwa wynosi $O((V+E)logV)$, natomiast w przypadku grafu zaimplementowanego za pomocą macierzy sąsiedztwa wynosi $O((V^2 + E)logV)$.

3 Wyniki pomiarów

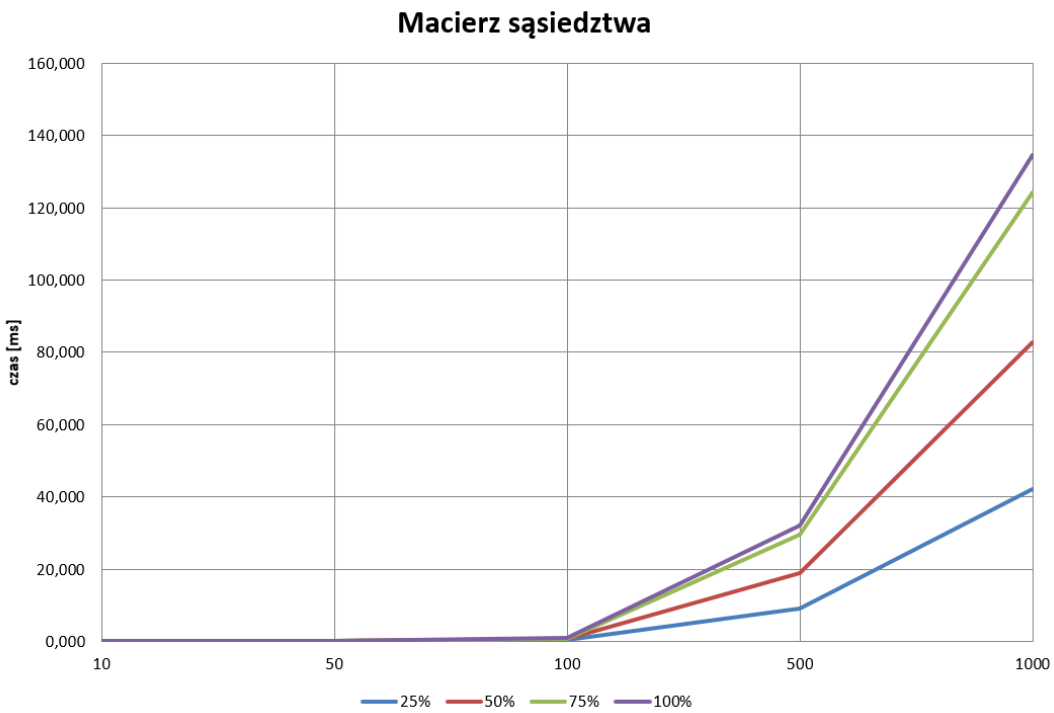
Tablica 1: Wyniki pomiaru czasu wykonania algorytmu Dijstry przy użyciu listy sąsiedztwa.

Gęstość \ Liczebność	Czasy otrzymane dla listy sąsiedztwa [ms]				
	10	50	100	500	1000
25%	0,017	0,117	0,361	14,372	62,351
50%	0,017	0,170	0,637	29,329	126,964
75%	0,021	0,231	0,890	45,539	193,248
100%	0,023	0,286	1,070	48,927	206,987

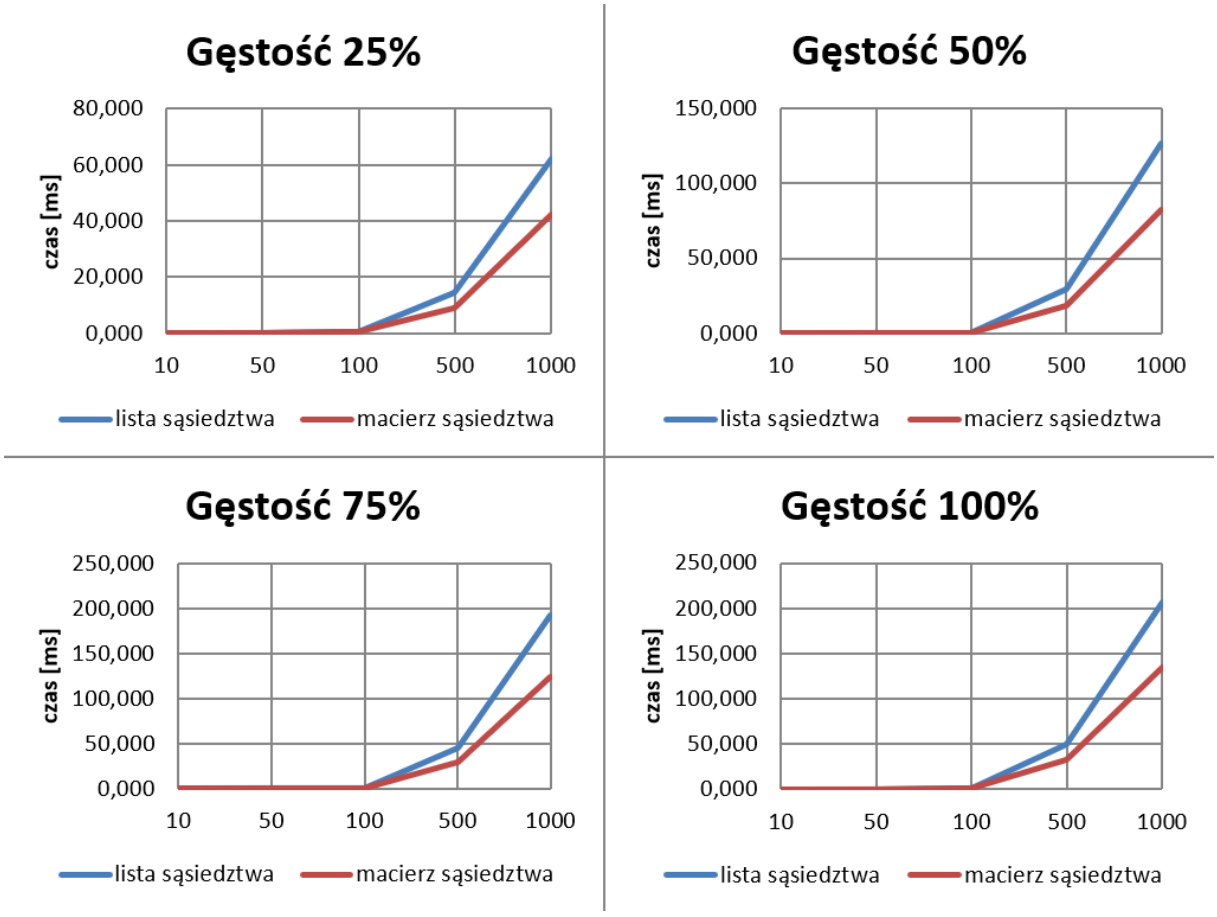
Tablica 2: Wyniki pomiaru czasu wykonania algorytmu Dijstry przy użyciu macierzy sąsiedztwa.

Gęstość \ Liczebność	Czasy otrzymane dla macierzy sąsiedztwa [ms]				
	10	50	100	500	1000
25%	0,018	0,115	0,349	9,159	42,226
50%	0,020	0,163	0,583	18,753	82,683
75%	0,024	0,208	0,763	29,535	124,346
100%	0,022	0,254	0,926	32,058	134,490

Rysunek 1: Wykres zależności czasu od ilości danych dla różnych gęstości grafu.



Rysunek 2: Wykres zależności czasu od ilości danych dla różnych gęstości grafu.



Rysunek 3: Porównanie wykresów zależności czasu od ilości danych dla różnych reprezentacji grafu.

4 Wnioski

Na podstawie powyższych wykresów i otrzymanych wyników można zauważyć, że zarówno zwiększenie ilości wierzchołków i zwiększenie gęstości grafu wydłuża czas działania algorytmu dla obydwu reprezentacji grafu (macierz i lista sąsiedztwa).

Mimo, iż złożoność obliczeniowa algorytmu Dijstry dla grafu zaimplementowanego za pomocą listy sąsiedztwa jest mniejsza od złożoności obliczeniowej macierzy sąsiedztwa czasy wykonania dla listy były dłuższe. Wynika to prawdopodobnie z faktu, że czas przechodzenia po elementach tabeli jest krótszy od czasu przechodzenia po elementach listy. Różnica czasu wykonania wzrasta wraz ze wzrostem gęstości, czyli gdy ilość elementów na listach incydencji zbliża się do ilości wierzchołków.

Literatura

1. Michael T. Goodrich, Roberto Tamassia , David M. Mount, *Data Structures and Algorithms in C++*, Second Edition, str. 593-694

2. Clifford A. Shaffer Department of Computer Science Virginia Tech Blacksburg, VA 24061, *A Practical Introduction to Data Structures and Algorithm Analysis*, Third Edition (C++ Version), str 401-435

3. Cormen T., Leiserson C.E., Rivest R.L., Stein C., *Wprowadzenie do algorytmów*, WNT, str. 671-673

4. Drozdek A., C++. *Algorytmy i struktury danych, Helion*, str. 81-93

5. Wróblewski P., *Algorytmy, struktury danych i techniki programowania*, Wydanie VI, Helion, str. 287-288

6. http://bcs.wiley.com/he-bcs/Books?action=resourcebcsId=6384itemId=0470383275resourceId=24107 Data Structures and Algorithms in C++, 2nd Edition, Source Code, 1 maja - 5 maja 2019

7. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/ Dijkstra's shortest path algorithm, 1 maja - 5 maja 2019

8. https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-greedy-algo-8/ Dijkstra's Algorithm for Adjacency List Representation, 1 maja - 5 maja 2019

9. https://www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/ Printing Paths in Dijkstra's Shortest Path Algorithm, 1 maja - 5 maja 2019