Laboratoria Algorytmów i Struktur Danych

Projekt 3 - sprawozdanie

Sortownia topologiczne Grafy

Prowadzący: **Dominik Piotr Witczak**

Przygotowali: **Jan Czyżewski 160377, Albert Łapa 160381**

Grupa: **LAB 13**

Kierunek: **Informatyka, zajęcia pt. 8:00-9:30**

POLITECHNIKA POZNAŃSKA

1 | Generowanie grafu

Program pozwala na wybranie reprezentacji grafu wraz z opcją jego generowanie lub wpisania danych z ręki.

```
fred@Fredputer:~/aisd$ /bin/python3 /home/fred/aisd/AiSD/main.py -generate
Enter how you want to view the graph (matrix, list, table): matrix
Enter the number of nodes: 7
Enter the saturation (in percentage): 100
[[0 1 1 1 1 1 1]
 [0 0 1 1 1 1 1]
 [0 0 0 1 1 1 1]
 [0 0 0 0 1 1 1]
 [0 0 0 0 0 1 1]
 [0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0]]
```

Rysunek 1: Generacja automatyczna

```
fred@Fredputer:~/aisd$ /bin/python3 /home/fred/aisd/AiSD/main.py -user-provided
Enter how you want to view the graph (matrix, list, table): matrix
Enter the number of nodes> 4
1> 2
2>
3> 2
4> 2 3
[[0. 1. 0. 0.]
 [0. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 1. 1. 0.]]
```

Rysunek 2: Generacja z danymi wprowadzonymi przez użytkownika

Program wyposażono w przejrzyste menu **help**.

```
--- Help ---
Commands:
Help - display this message
Quit - exit the program
Find - find an edge in the graph
BFS - perform a breadth-first search
DFS - perform a depth-first search
kahn - perform a topological sort using Kahn's algorithm
tarjan - perform a topological sort using Tarjan's algorithm
Tikz - save the graph to a LaTeX file
```

Rysunek 3: Menu programu

2 | Prezentacja grafów w terminalu

Dane zostały wygenerowane automatycznie dla 7 węzłów oraz 100 procentach saturacji.

```
[[0 1 1 1 1 1 1]
 [0 0 1 1 1 1 1]
 [0 0 0 1 1 1 1]
 [0 0 0 0 1 1 1]
 [0 0 0 0 0 1 1]
 [0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0]]
```

Matryca

Source Node	Target Node
1	2
1	3
1	4
1	5
1	6
1	7
2	3
2	4
2	5
2	6
2	7
3	4
3	5
3	6
3	7
4	5
4	6
4	7
5	6
5	7
6	7

Tablica

```
Node 1 has edges to: 2 3 4 5 6 7
Node 2 has edges to: 3 4 5 6 7
Node 3 has edges to: 4 5 6 7
Node 4 has edges to: 5 6 7
Node 5 has edges to: 6 7
Node 6 has edges to: 7
Node 7 has edges to:
```

Lista

3 | Funkcje programu

```
Enter a command: find
from> 2
to> 4
True: edge (2,4) exists in the graph :)
Enter a command: bfs
Enter the start node for given search method: 2
2 3 4 5 6 7 1
Enter a command: dfs
Enter the start node for given search method: 2
2 3 4 5 6 7 1
Enter a command: kahn
Topological order: 1 2 3 4 5 6 7
Enter a command: tarjan
Topological order: 1 2 3 4 5 6 7
Enter a command:
```

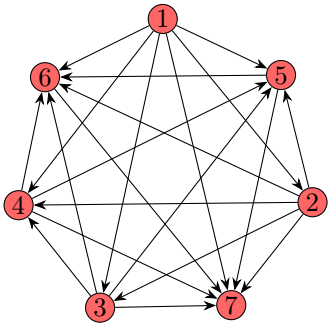
Rysunek 5: Użyte funkcje

4 | Wizualizacje grafów

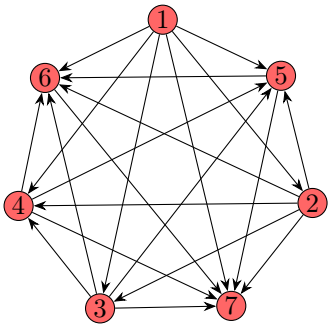
Ekport grafów jest możliwy do tikzpicture za pomocą komendy wraz z własną nazwą.
Wygenerowany graf skaluje się w zależności od ilości węzłów.

```
Enter a command: tikz
Enter the filename for the LaTeX file: sussybaka
```

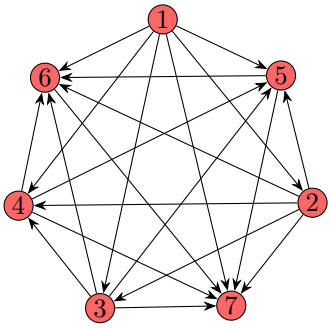
Rysunek 6: Działanie komendy tikz



Rysunek 7: Wizualizacja Matrycy



Rysunek 8: Wizualizacja Listy

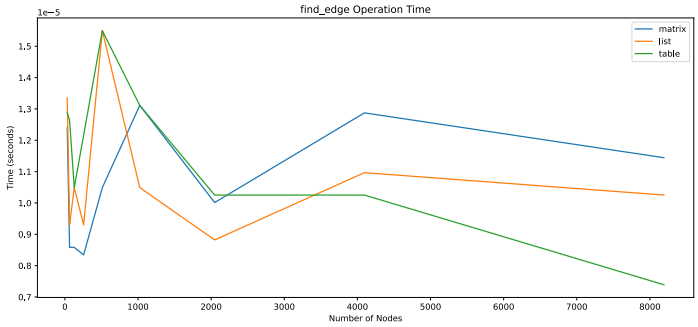


Rysunek 9: Wizualizacja Tablicy

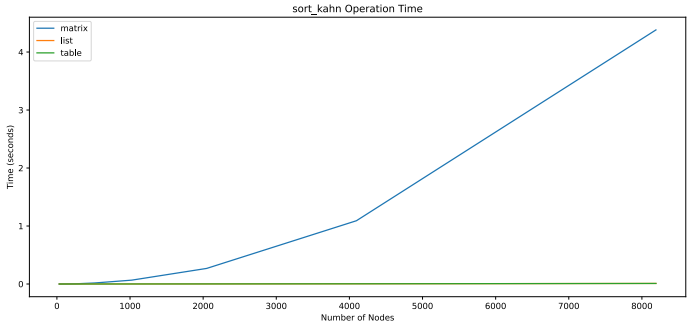
5 | Wykresy

Efektywność algorytmów sprawdzono przy pomocy dostarczonych plików typu benchmark. Komputery testującę wyposażone były w jednostki obliczeniowe Apple M1 oraz Intel Core i5 8500 wraz z 16 GB pamięci operacyjnej każda. Jednostka od Apple pracowała pod kontrolą systemu operacyjnego MacOS Sonoma 14.4.1, natomiast Intel pod kontrolą systemu operacyjnego Windows 11 Pro 23H2 w środowisku wirtualnym WSL Ubuntu. Testy wykonano na algorytmach napisanych w języku Python w wersji 3.12.2.

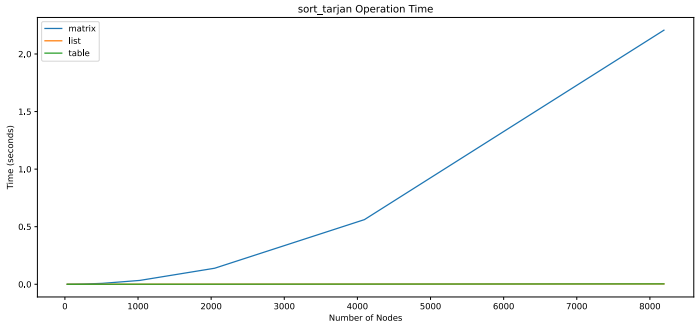
5.1 | Windows(WSL)



Szukanie krawędzi

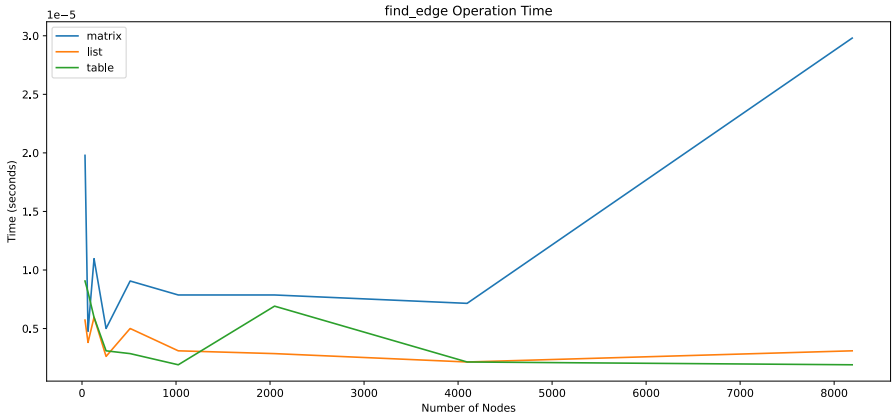


Algorytm Kahna

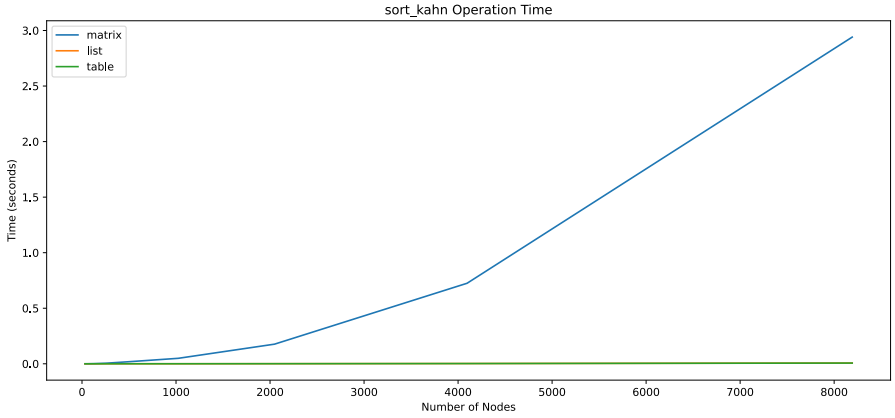


Algorytm Tarjana

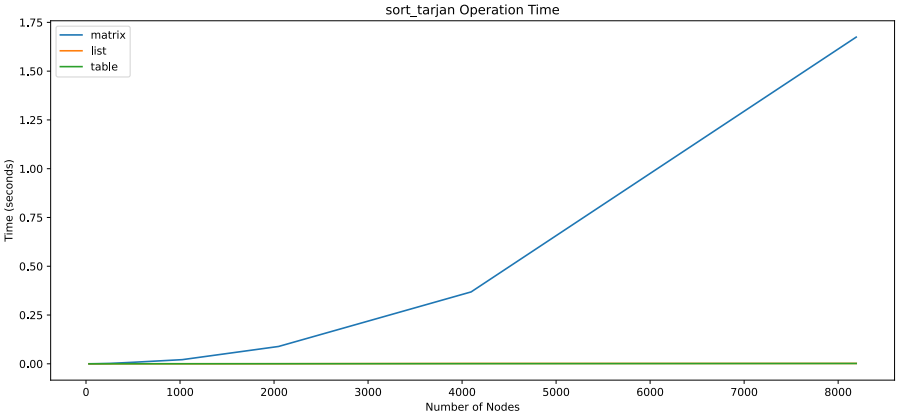
5.2 | Apple



Szukanie krawędzi



Algorytm Kahna



Algorytm Tarjana

6 | Podsumowanie

Powyższe wykresy pokazują różnicę w zachownianiu omawianych algorytmów podczas używania różnych typów danych. Obrazują również, kiedy lepiej zastosować odpowiedni algorytm.