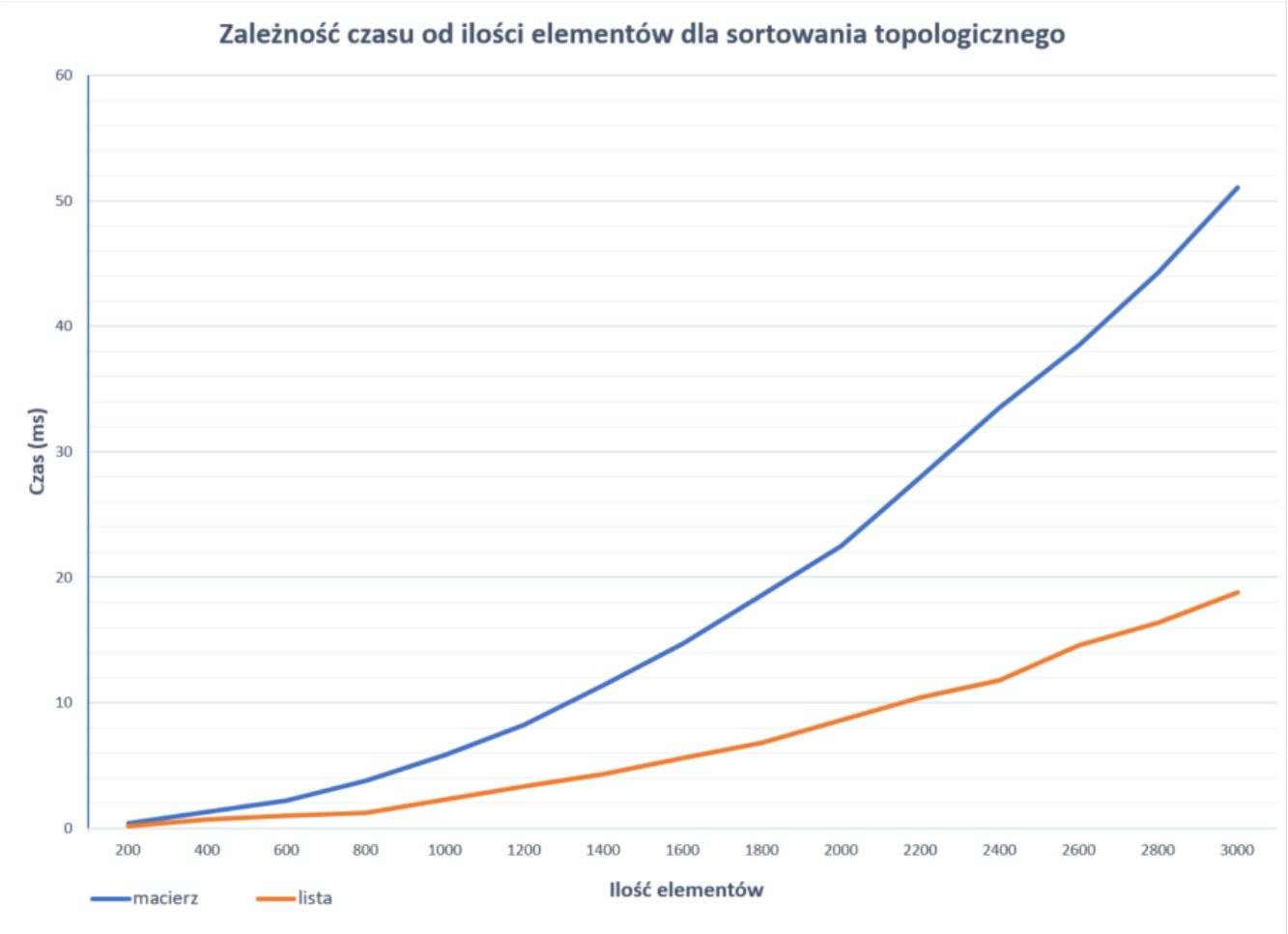
Zadanie 3. – Algorytmy grafowe

Grupa I1, Jan Techner (132332), Sebastian Maciejewski (132275)

Wszystkie wykresy są zrealizowane w podobny sposób: na osi oY oznaczono czas działania algorytmów w milisekundach, zaś na osi oX przedstawiono ilość danych (ilość węzłów grafu).

Pomiary wykonano dla przynajmniej 10 próbek danych (czas ich działania jest średnią czasów dla poszczególnych próbek) w przypadku sortowania topologicznego, zaś dla znajdowania minimalnego drzewa rozpinającego (MST) wykonano nawet do 100 pomiarów dla danej ilości elementów.

Zacznijmy od rozważenia problemu sortowania topologicznego skierowanego grafu acyklicznego (DAG). Na potrzeby pomiarów wygenerowane zostały losowe grafy DAG o nasyceniu krawędziami 60%, przedstawione w postaci macierzy sąsiedztwa i listy incydencji, następnie poddano je sortowaniu topologicznemu wykorzystując przeszukiwanie w głąb. Zależność zmierzonych czasów sortowania od ilości węzłów grafu dla danego sposobu jego reprezentacji przedstawia wykres:



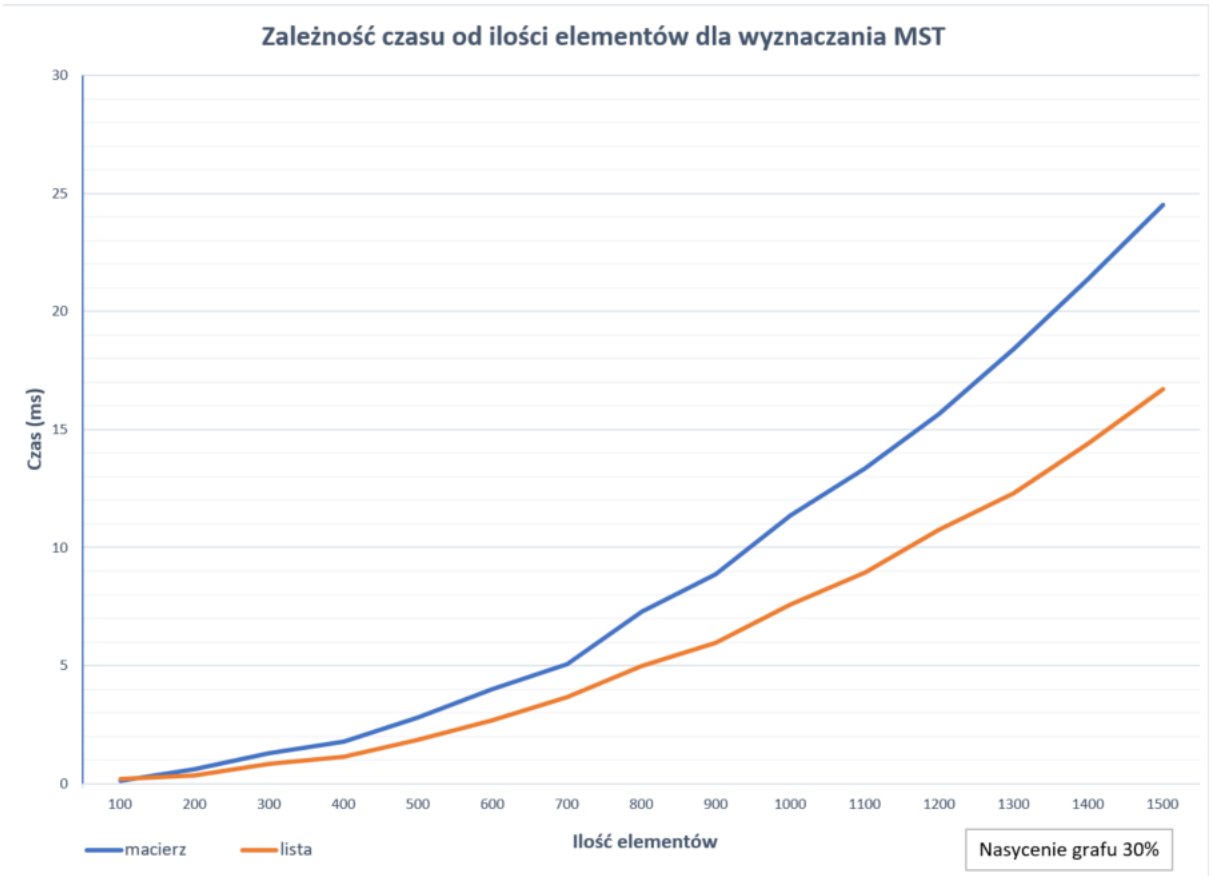
Jak widać z wykresu, problem sortowania topologicznego zdecydowanie faworyzuje reprezentację grafu przy pomocy listy, zatem już na sam początek nasuwa się nam wniosek, że odpowiednim doborem reprezentacji grafu dla tego problemu jest właśnie przedstawienie go jako listy incydencji. Tak znaczna różnica na niekorzyść macierzy sąsiedztwa między zmierzonymi czasami (dla 3000 elementów czas działania dla listy stanowi zaledwie 37% wartości zmierzonej dla macierzy) wynika z ustalonej z góry wielkości macierzy sąsiedztwa. Algorytm, w przypadku reprezentacji

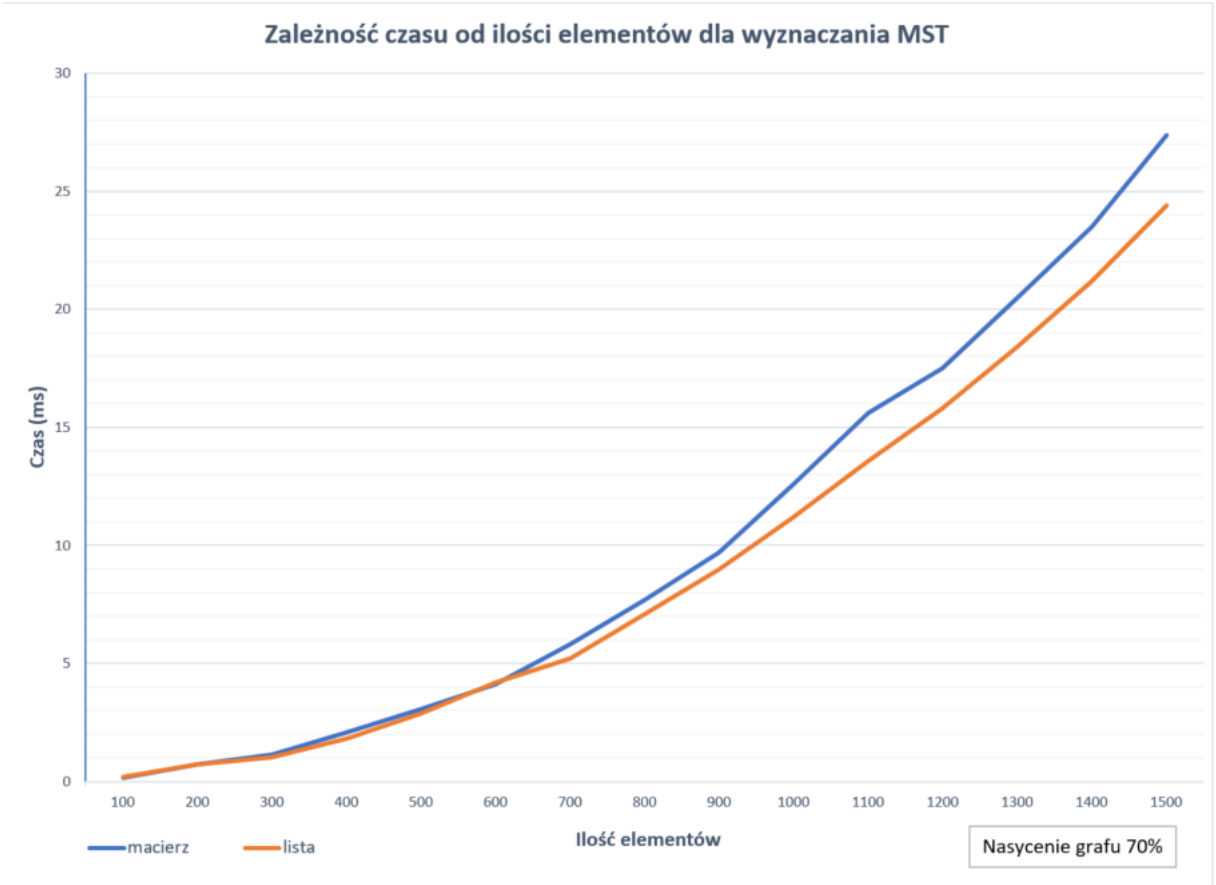
macierzą, za każdym razem pracuje na tabeli n-elementowej, a następnie na kolejnej n-elementowej tablicy, co wymaga, nawet mimo użycia rekurencyjnego algorytmu przeszukiwania w głąb, przeszukania znacznie większej ilości danych (duża ilość porównań) niż w przypadku listy incydencji, która różni się od macierzy tym, że jej wielkość zależy od ilości połączeń (nasycenia grafu) – lista składa się z n-elementowej tabeli wskaźników na głowy list, które mają najwyżej n elementów, jednak w przypadku nasycenia 60% średnia wielkość listy wynosi właśnie 0,3*n, co skutkuje znacznie mniejszą ilością niezbędnych porównań.

Mimo, iż na ogół listy działają wolniej niż tablice (wymagają przejścia przez kolejne elementy), to jednak w tym przypadku okazują się lepszym wyborem, ponieważ ilość porównań koniecznych przy sortowaniu topologicznym jest bardzo duża. Możemy sobie wyobrazić, że dla grafu o nasyceniu krawędziami 100% to właśnie macierz sąsiedztwa będzie lepszym wyborem reprezentacji, ponieważ jej zaletą jest szybszy niż w przypadku listy (gdy długość każdej z list jest maksymalna) czas dostępu do danego elementu (dla macierzy jest on stały).

Możemy zatem wysnuć wnioski dotyczące złożoności obliczeniowej tego algorytmu sortowania w zależności od sposobu reprezentacji grafu: zarówno dla listy jak i dla macierzy złożoność tego algorytmu to $O(n^2)$, jednak znaczna różnica czasów działania między nimi wynika głównie z wielkości tych struktur – dla nasycenia krawędziami mniejszego niż 100% zawsze lista będzie zawierała mniej elementów niż macierz, co prowadzi do szybszego działania algorytmu dla listy, mimo, że teoretyczna złożoność obliczeniowa jest dokładnie taka sama dla obydwóch struktur.

Przejdźmy teraz do problemu wyznaczania minimalnego drzewa rozpinającego dla grafu nieskierowanego z wagami na krawędziach, przedstawionego w postaci listy incydencji i macierzy sąsiedztwa. W tym celu dokonaliśmy implementacji algorytmu Prima. Pomiary wykonano dla dwóch rodzajów grafów – o nasyceniu 30% oraz 70% - aby ukazać zmianę w różnicy między czasem działania algorytmu na grafie reprezentowanym listą, a czasem działania dla grafu reprezentowanego macierzą. Wyniki przedstawione zostały na poniższych wykresach:


Zależność czasu od ilości elementów dla wyznaczania MST
Czas (ms)
0
5
10
15
20
25
30
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
Ilość elementów
macierz
lista
Nasycenie grafu 30%


Zależność czasu od ilości elementów dla wyznaczania MST
Czas (ms)
0
5
10
15
20
25
30
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
Ilość elementów
macierz
lista
Nasycenie grafu 70%

Już na pierwszy rzut oka widać, że dla algorytmu znajdowania MST różnica między czasami działania dla poszczególnych struktur danych jest mniejsza niż w przypadku algorytmu sortowania topologicznego. Od razu widać także, że czas działania algorytmu dla listy incydencji jest w obu przypadkach szybszy niż dla macierzy sąsiedztwa. Narzuca się zatem wniosek, że lepszym wyborem jest użycie do rozwiązania tego problemu listy incydencji.

Takie tendencje na wykresach są spowodowane dokładnie tą samą różnicą między listami a macierzami, która była przyczyną znacznych różnic w pomiarach z poprzedniego zadania – różnicą w rozmiarach listy incydencji i macierzy sąsiedztwa. Należy w tym miejscu zauważyć, że dla nasycenia grafu 70% czasy działania algorytmu dla listy i macierzy niewiele już się różnią, znacznie większą dysproporcję można zauważyć na wykresie dla grafu nasyconego tylko w 30%. Jest to spowodowane właśnie ową różnicą w rozmiarach – dla 70% nasycenia lista incydencji jest już niewiele mniejsza niż macierz sąsiedztwa, a zgodnie z tym, co napisaliśmy w opisie poprzedniego wykresu, wiemy, że czas dostępu do elementów listy jest większy, co sprawia, że różnica między krzywymi na wykresie jeszcze bardziej się zmniejsza.

Co do złożoności obliczeniowej tego algorytmu, sytuacja jest podobna do tej z poprzedniego algorytmu – zarówno dla listy incydencji jak i dla macierzy sąsiedztwa złożoność obliczeniową można opisać jako O(n*log(n)). Różnica w czasie działania algorytmu dla tych dwóch struktur wynika tylko z faktu, iż lista incydencji jest w obydwu przypadkach (nasycenie 30% i 70%) mniejsza od macierzy sąsiedztwa – zupełnie tak samo jak było dla algorytmu sortowania topologicznego.

Wielomianowy czas działania opisywanych algorytmów dowodzi ich przynależności do klasy złożoności P – problemów łatwo rozwiązywalnych na deterministycznej maszynie Turinga.