AiSD — zadanie trzecie

Gliwiński Jarosław Marek
Kruczyński Konrad Marek
Grupa dziekańska I5

5 czerwca 2008

1 Wstęp

Celem postawionym przez zadanie trzecie było tzw. sortowanie topologiczne. Jest to typ sortowania przydatny w zagadnieniach szeregowania zadań. Daną wejściową dla algorytmu sortowania topologicznego jest skierowany, spójny (w sensie spójności nieskierowanej) graf acykliczny. Każdy wierzchołek grafu symbolizuje pewne zadania do wykonania, a powiązania pomiędzy wierzchołkami tj. krawędzie odzwierciedlają zależności pomiędzy zadaniami w następujący sposób: jeżeli zadanie A musi być wykonane przed B, to istnieje krawędź od A do B. W wyniku wykonania algortymu sortowania topologicznego uzyskać powinniśmy zalecaną kolejność wykonania zadań, spełniającą zależności podane w grafie.

W dalszej części protokołu ilość wierzchołków grafu oznaczać będziemy symbolem n, natomiast ilość jego krawędzi – m, która w przypadku grafów z zadania będzie proporcjonalna do n^2, o czym warto pamiętać.

Oczywiście graf w pamięci komputera może być przechowywany na wiele różnych sposobów. Reprezentacje te różnią się efektywnością wykonywania różnych operacji i tak też jest w przypadku omawianego algorytmu. W naszym programie zastosowaliśmy cztery różne struktury:

- macierz sąsiedztwa;
- lista następników;
- lista krawędzi;
- macierz grafu.

2 Pomiary

2.1 Przygotowanie struktury

Pierwszym etapem działania programu jest przygotowanie grafu (w formie macierzy sąsiedztwa), który spełnia warunki zadania, tj. spójnego i acyklicznego, o żądanym wypełnieniu, które w naszym zadaniu wynosi 50%. Wykonywane jest to w następujący

sposób. Na początku w sposób losowy łączymy wierzchołki w grafie (przy czym tworzymy górną trójkątną macierz sąsiedztwa, co zapewnia acykliczność) aż do uzyskania wymaganego wypełnienia. Wówczas powstały graf jest w zdecydowanej większości wypadków spójny (wszystkie grafy wygenerowane podczas pomiarów okazały się spójną), jednak na wszelki wypadek po jego utworzeniu przechodzi on próbę spójności. Wykonywana jest ona w następujący sposób: badamy, czy z wybranego wierzchołka można (niezależnie od kierunku krawędzi) dotrzeć do wszystkich pozostałych. Jeżeli powstanie graf niespójny, jest on usuwany z pamięci i procedura rozpoczyna się od nowa.

W dalszej kolejności program przekształca macierz sąsiedztwa w pozostałe wymagane struktury i na każdej z nich wykonuje odpowiedni algorytm sortowania topologicznego (taki sam z punktu widzenia abstrahującego od struktury grafu w pamięci) i mierzy czas jego wykonania. Wypracowana metoda pomiaru czasu jest identyczna ze stosowaną w zadaniach pierwszym i drugim.

Zmierzone wyniki przedstawione są na kilku wykresach. Na osi odciętych zaznaczone są rozmiary grafu (ilość wierzchołków), natomiast na osi rzędnych średni czas (z trzydziestu prób) wykonywania algorytmu w centysekundach. Wielkości odchyleń mówią o tym, jak duży wpływ na wydajność algorytmu miała konkretna postać grafu, dlatego może być dosyć duża, natomiast odchylenie odchylenia powinno być niewielkie, jeżeli próba miała dostatecznie duży rozmiar. Wspomniane wielkości zebrane zostały w tabeli pod wykresem zbiorczym.

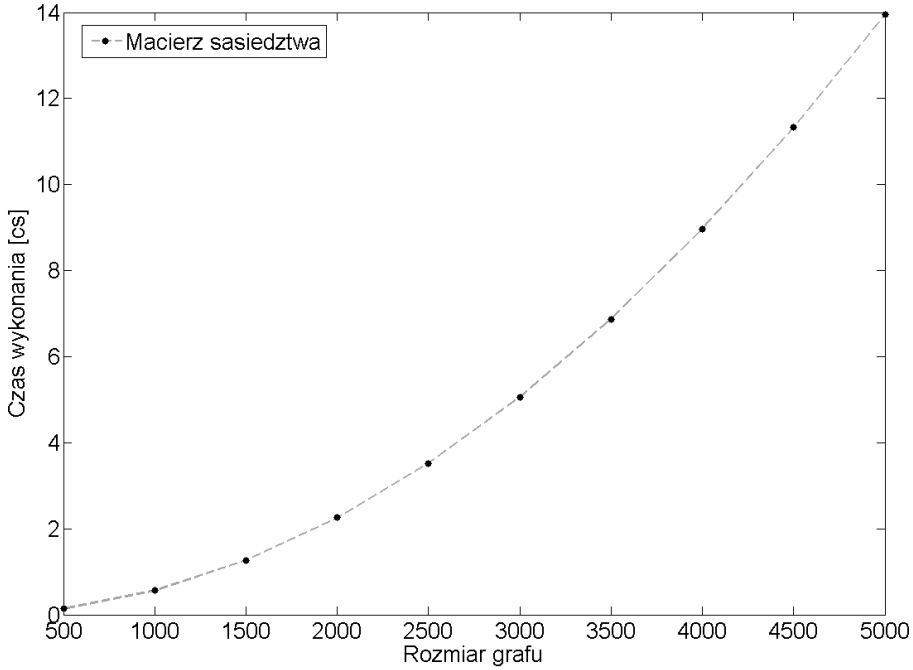
2.2 Macierz sąsiedztwa

Pora na kilka słów na temat złożoności, wad i zalet macierzy sąsiedztwa jako formy reprezentacji grafu. Złożoność algorytmu sortowania topologicznego wynosi w tym wypadku $O(n^2)$ dla przypadku pesymistecznego, który jednak w dosyć gęstych (50% wypełnienia) grafach często ma miejsce. Pozostałe złożoności zawarte są w tabeli:

Typ złożoności	Wartość
Pamięciowa	$O(n^2)$
Znalezienia krawędzi	$O(1)$
Przejrzenia następników	$O(n)$

Jak widać, główną zaletą macierzy sąsiedztwa jest łatwość sprawdzenia, azaliż dane dwa wierzchołki są incydentne. Warto wspomnieć o łatwości implementacji tej struktury – jest ona zupełnie naturalna dla człowieka.

Poniżej wykres:



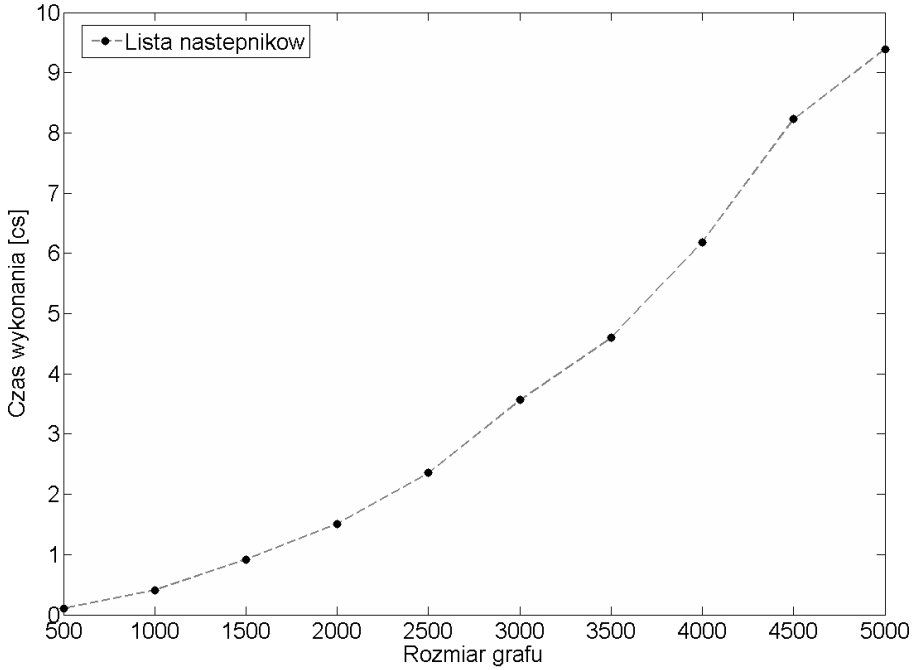
2.3 Lista następników

W tym wypadku złożoność sortowania jest taka sama, jak poprzednio, tj. $O(n^2)$. Pozostałe podano poniżej:

Typ złożoności	Wartość
Pamięciowa	$O(m+n)$
Znalezienia krawędzi	$O(n)$
Przejrzenia następników	$O(n)$

Pozornie wydawać by się mogło, iż struktura ta nie wnosi żadnych korzyści w stosunku do macierzy sąsiedztwa, a nadto charakteryzuje się gorszą złożonością znalezienia krawędzi. W praktyce jednak tak nie jest, zgodnie ze swoją nazwą przeglądanie następników przy użyciu tej struktury jest zazwyczaj szybsze. Jest tak, ponieważ należy mieć na uwadze, iż podana w tabeli powyżej złożoność dotyczy przypadku pesymistycznego (najgorsza możliwość to połączenie danego wierzchołka ze wszystkimi pozostałymi), natomiast dla macierzy sąsiedztwa podano jedyny występujący przypadek. Innymi słowy znalezienie *pierwszego w liście* następnika odbywa się w czasie stałym, a to właśnie typowa potrzeba algorytmu sortowania topologicznego.

Na koniec wykres:



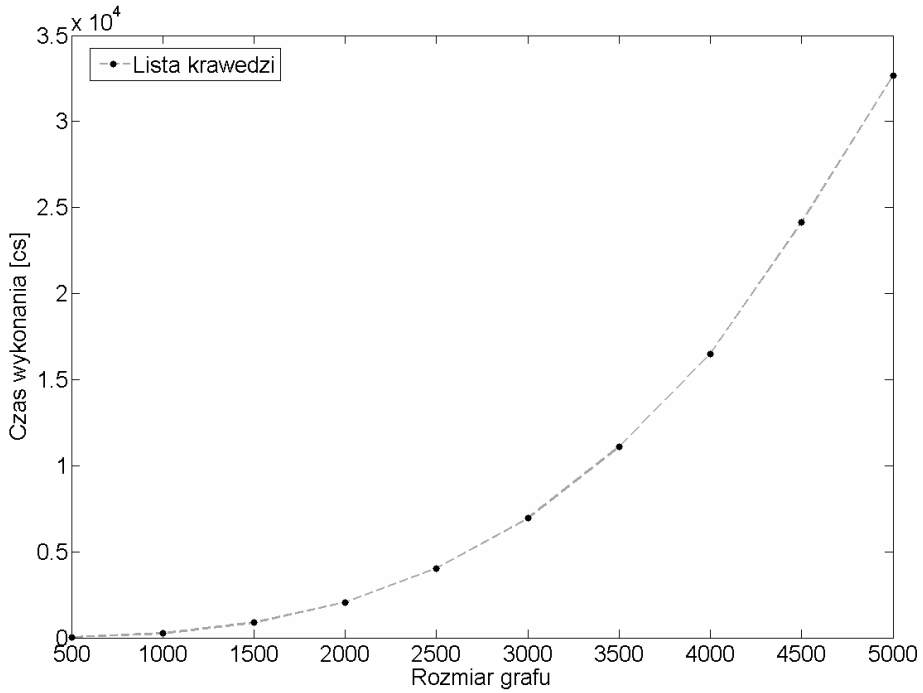
2.4 Lista krawędzi

Lista krawędzi jest wyjątkiem pod względem złożoności sortowania topologicznego, ponieważ wynosi on tu $O(n^2 \cdot m)$, czyli w naszym przypadku $O(n^4)$. Czasy wobec tego są znacznie wydłużone w stosunku do poprzednich. Pozostałe złożoności:

Typ złożoności	Wartość
Pamięciowa	$O(m)$
Znalezienia krawędzi	$O(m)$
Przejrzenia następników	$O(m)$

Jak widać, lista krawędzi jest strukturą przydatną i wydajną dla grafów rzadkich, tj. takich które zawierają niewielką liczbę krawędzi. Ponieważ jednak grafy o stałym i dużym wypełnieniu do takich nie należą, jej zastosowanie w naszym przypadku nie ma większego sensu.

Poniższy wykres obrazuje czas pracy algorytmu:

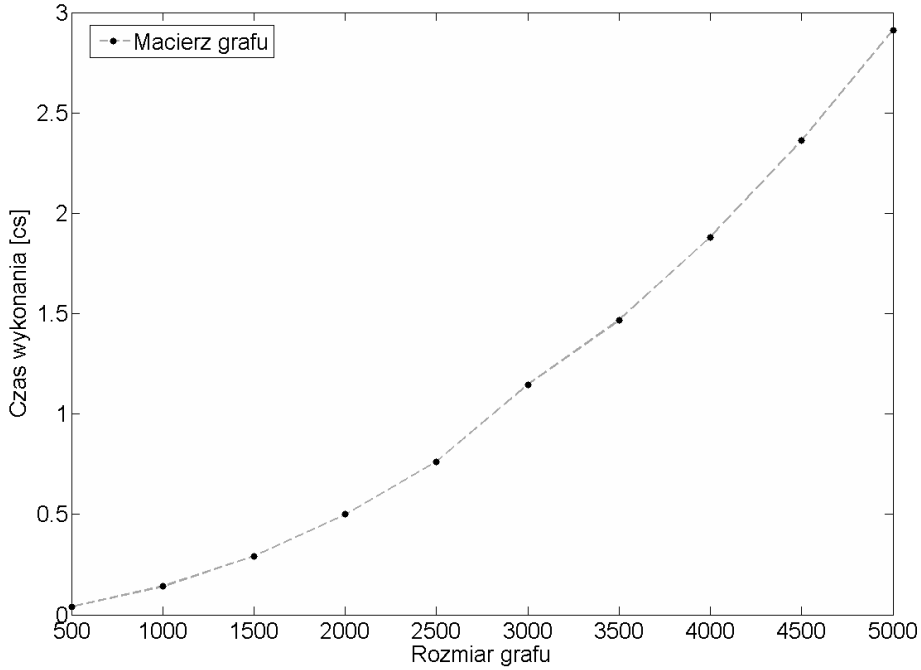


2.5 Macierz grafu

Macierz grafu jest strukturą łączącą zalety wszystkich wcześniej wymienionych, tj. pozwala w stałym czasie powiedzieć o incydentności danych dwóch wierzchołków, jak i podać następniki danego wierzchołka równie efektywnie, co lista następników. W związku z tym jest optymalną strukturą algorytmu sortowania topologicznego, którego złożoność w tym wypadku wynosi $O(n^2)$. Pozostałe zaś poniżej:

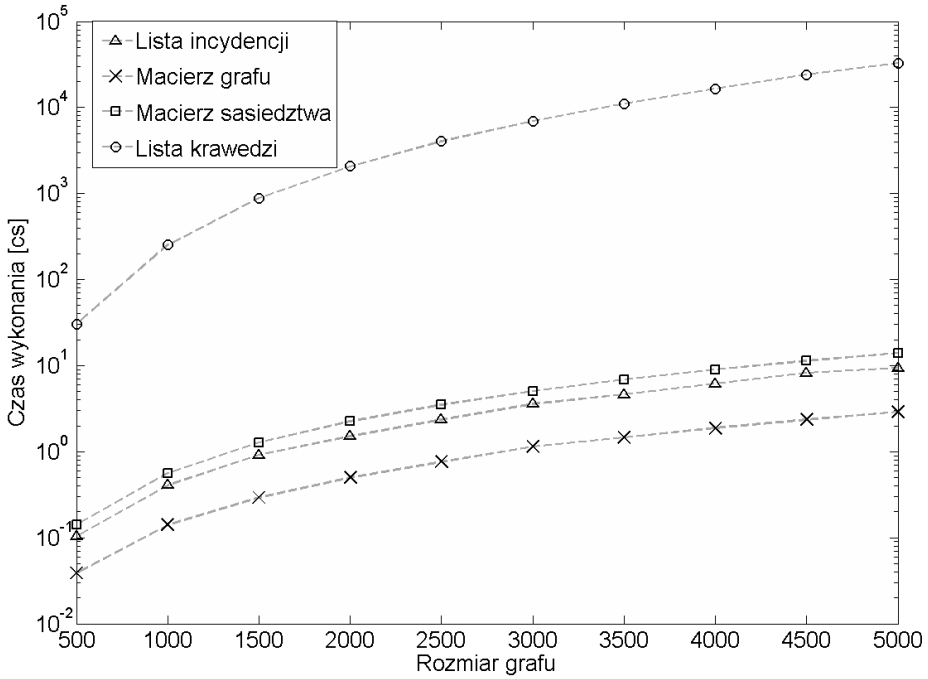
Typ złożoności	Wartość
Pamięciowa	$O(n^2)$
Znalezienia krawędzi	$O(1)$
Przejrzenia następników	$O(n)$

Na koniec warto dodać, że w programie nie zaimplementowano występującej w oryginalnej macierzy grafu listy wierzchołków nie będacych ani poprzednikami, ani następnikami, ponieważ nie było ku temu potrzeby. Poniżej przedstawiony jest wykres:



2.6 Wykres zbiorczy

W ostatniej części sprawozdania zawarty został wykres, który pozwala na łatwe porównanie wszystkich czterech algorytmów oraz podane zostały odchylenia.



Odchylenia wyników w próbie przedstawiają się:

Struktura	Odchylenie względne [%]
Macierz sąsiedztwa	$1,4 \pm 0,04$
Lista następników	$10,75 \pm 2,16$
Lista krawędzi	$2,34 \pm 0,21$
Macierz grafu	$3,48 \pm 2,04$

Z porównania widocznego na zbiorczym wykresie wynikają oczywiste wnioski dotyczące efektywności badanych struktur w zadaniach dotyczących bezpośrednio rozumianego sortowania topologicznego. Przede wszystkim widać ogromną różnicę pomiędzy listą krawędzi a pozostałymi reprezentacjami grafów. Przyczyną tak znacznej różnicy jest widoczna w przedstawianych wcześniej tabelach złożoność operacji dostępu do następników danego wierzchołka. Różnice pomiędzy pozostałymi reprezentacjami są już mniejsze. Spośród nich najwolniejsza okazuje się macierz sąsiedztwa. Przyczyną takiego wyniku jest statyczna natura tablicy, przyporządkowująca każdemu hipotetycznie możliwemu połączeniu wartość logiczną, która sprawia, iż przy przeglądaniu następników wierzchołka należy przeiterować przez cały wiersz macierzy – a więc także wierzchołki w rzeczywistości niepołączone z aktualnie rozpatrywanym wierzchołkiem. Kolejną w hierarchii wydajnościowej jest lista (a właściwie listy) następników. Przewaga jej wynika z usunięcia redundantnych przebiegów przy przeglądaniu listy następników. Połączenia nie występujące w grafie po prostu nie istnieją fizycznie w strukturze danych, toteż problem odróżniania, czy sprawdzane połączenie istnieje, staje się nieistotny. Najszybszą wreszcie strukturą okazała

się macierz grafu. Zachowuje ona zalety dwóch poprzednich: stały czas sprawdzania dowolnej krawędzi jak i najkrótszy możliwy czas przeglądania listy następników danego wierzchołka.