Wtorki 16:50
Grupa I3
Kierunek Informatyka
Wydział Informatyki
Politechnika Poznańska

Algorytmy i struktury danych

Sprawozdanie z zadania w zespołach nr. 3
prowadząca: dr hab. inż. Małgorzata Sterna, prof PP

Algorytmy Grafowe

autorzy:
Piotr Więtczak nr indeksu 132339
Tomasz Chudziak nr indeksu 136691

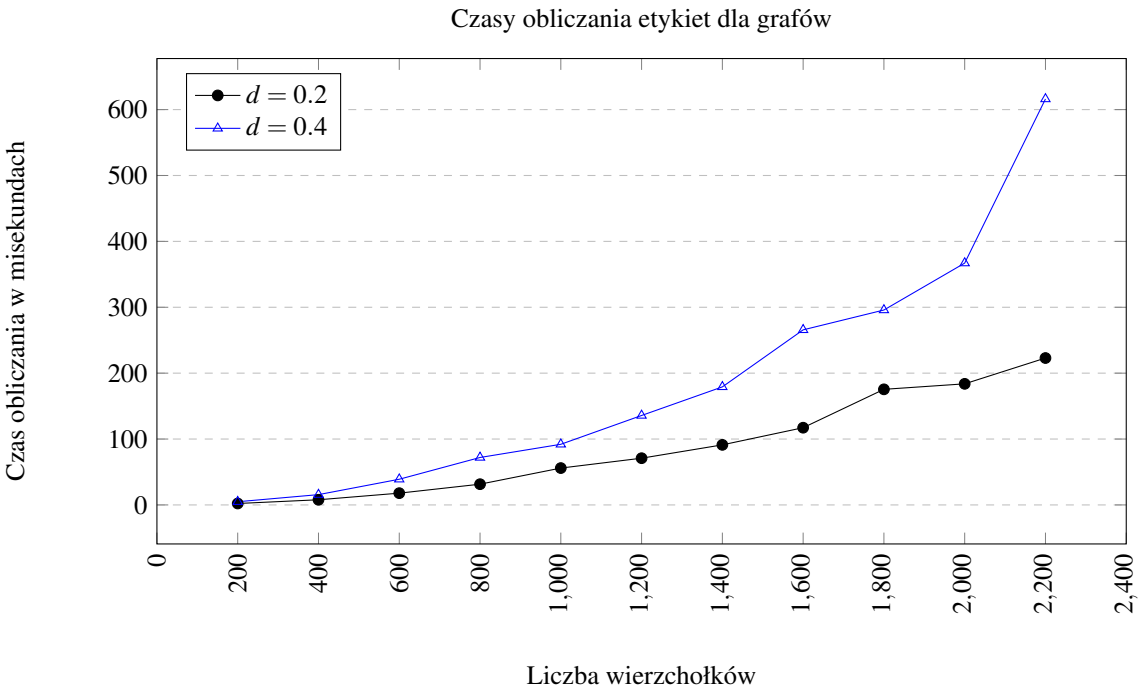
7 maja 2018

1 Opis implementacji

Do tworzenia grafu w trzech reprezentacjach (lista następników, macierz sąsiedztwa, lista łuków), zliczania łuków powrotnych i obliczania etykiet czasowych dla wierzchołków użyliśmy języka C++. Do pomiarów czasu wykorzystaliśmy klasę $std :: chrono :: high\_resolution\_clock$ z biblioteki *chrono*. Do implementacji procedury obliczającej etykiety czasowe rozpoczęcia i zakończenia analizy wierzchołków posłużyliśmy się listą następników. Przy wyborze reprezentacji grafu kierowaliśmy się czasem znalezienia zbioru następników, dlatego wybraliśmy listę następników ze względu na jej najniższą teoretyczną złożoność wyznaczenia następników O(n).

2 Obliczanie etykiet

Liczba wierzchołków	Czasy obliczania etykiet $[ms]$	
	$d = 0.2$	$d = 0.4$
200	2.172	4.841
400	7.830	15.622
600	17.860	38.984
800	31.477	72.013
1000	55.936	92.017
1200	70.922	135.757
1400	91.161	179.157
1600	117.204	265.713
1800	175.443	295.831
2000	183.787	367.042
2200	222.953	616.094



Metoda sortowania topologicznego opiera się na algorytmie DFS – przeszukiwania w głąb. Algorytm ten polega na odwiedzeniu wszystkich wierzchołków. W pierwszej kolejności wybiera on wierzchołki o najmniejszym możliwym numerze względem całej ścieżki, jaką już przebył, jeżeli nie ma już dostępnych wierzchołków to kończy ścieżkę. Następnie ze wszystkich dostępnych wierzchołków wybiera ten najmniejszy nieodwiedzony i powtarza algorytm. Kończy się on natomiast , gdy wszystkie wierzchołki zostały odwiedzone. Złożoność obliczeniowa w tym przypadku to O(n+m). Efektywność algorytmu DFS zależy od reprezentacji grafu. Najczęściej wykonywaną operacją przez ten algorytm jest przeszukiwanie listy poprzedników w celu znalezienia kolejnego wierzchołka. Wynika z tego, że najkorzystniejszą strukturą będzie lista następników, następnie macierz sąsiedztwa, dla których złożoność będzie wynosiła O(n). Mniej do tego algorytmu nadaje się lista łuków i lista poprzedników, dla których złożoność wynosi kolejno O(m) i O(n+m). Najmniej korzystną strukturą jest macierz incydencji, dla której ta operacja może trwać O(n*m). Duży wpływ na czas trwania tej operacji ma równie gęstość grafu. Im ten jest gęstszy, algorytm musi sprawdzić większą ilość wierzchołków, czego konsekwencją jest dłuższy czas pracy.

3 Liczba łuków powrotnych

Liczba wierzchołków	Liczba łuków powrotnych	
	$d = 0.2$	$d = 0.4$
200	4023	8084
400	16077	32174
600	36237	72413
800	64049	127892
1000	100107	199838
1200	144184	288110
1400	195830	392120
1600	255698	512729
1800	324218	647853
2000	399722	801067
2200	484240	967681

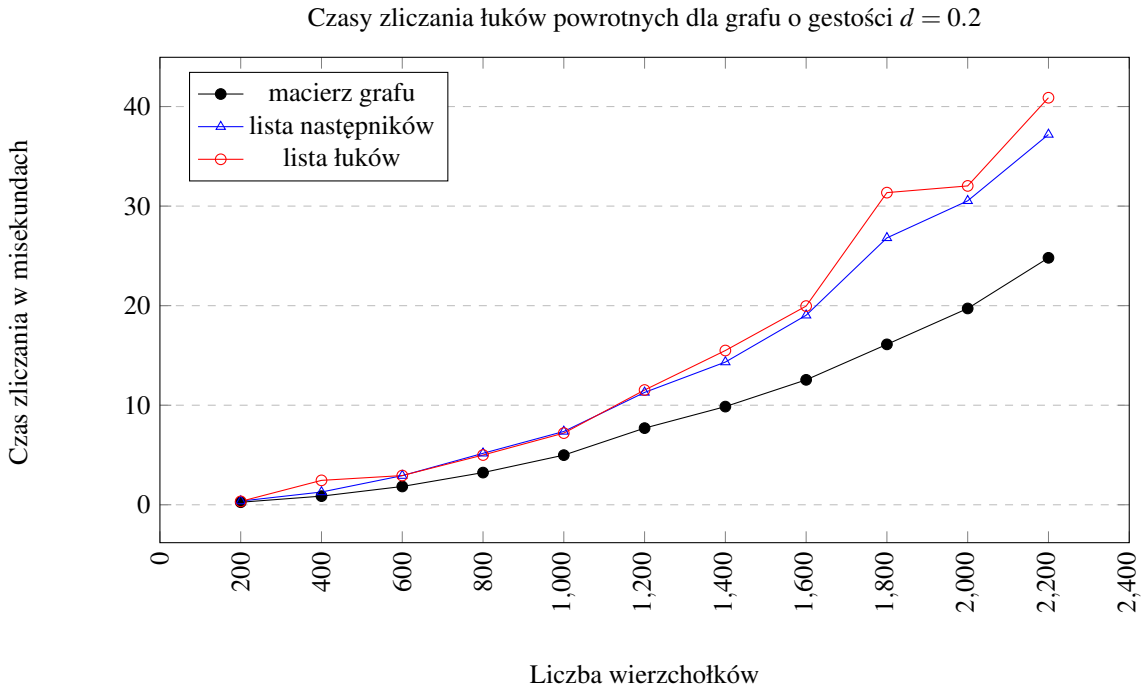
Tablica 1: Tabela prezentująca liczbę łuków powrotnych w grafach

Wszystkie wygenerowane grafy były cykliczne, ponieważ posiadały łuki powrotne. Gdyby któryś z wygenerowanych grafów nie posiadał łuków powrotnych byłby acykliczny. Grafy o dwa razy większej gęstości miały dwa razy więcej łuków powrotnych. Wraz z wzrostem liczby wierzchołków w grafach rosła ilość łuków powrotnych. Nie można sortować grafów skierowanych z cyklami, ponieważ nie można stwierdzić którą czynność z cyklu wykonać jako pierwszą. Wierzchołki w każdym grafie acyklicznym skierowanym można posortować na co najmniej jeden sposób, innych grafów nie można sortować topologiczne.

4 Czasy zliczania łuków powrotnych

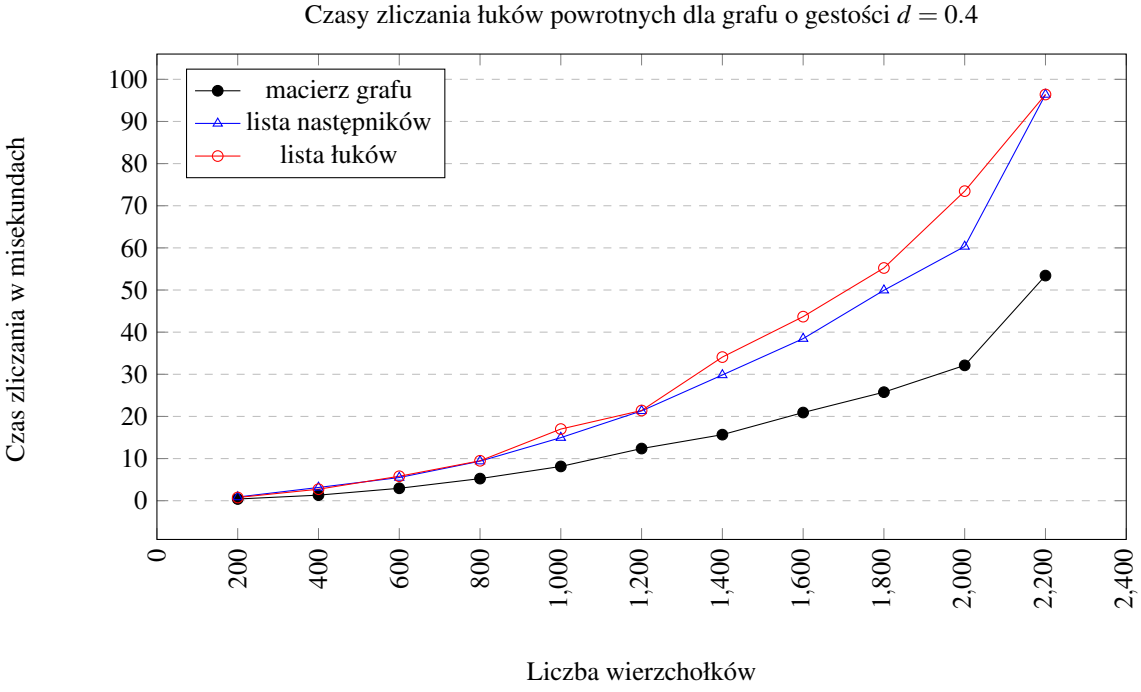
4.1 Prezentacja wyników dla grafu o gęstości $d = 0.2$

Liczba wierzchołków	Czasy zlicznia w grafie o $d = 0.2$ $[ms]$		
	Macierz sąsiedztwa	Lista następników	Lista łuków
200	0.256	0.370	0.341
400	0.867	1.280	2.451
600	1.837	2.924	2.934
800	3.232	5.170	4.996
1000	4.986	7.363	7.198
1200	7.699	11.285	11.526
1400	9.860	14.333	15.494
1600	12.549	19.035	19.957
1800	16.112	26.807	31.353
2000	19.719	30.530	32.027
2200	24.805	37.193	40.884



4.2 Prezentacja wyników dla grafu o gęstości $d = 0.4$

Liczba wierzchołków	Czasy zlicznia w grafie o $d = 0.4$ $[ms]$		
	Macierz sąsiedztwa	Lista następników	Lista łuków
200	0.403	0.873	0.758
400	1.320	3.141	2.727
600	2.945	5.497	5.801
800	5.242	9.379	9.473
1000	8.132	14.965	17.002
1200	12.375	21.315	21.404
1400	15.673	29.834	34.082
1600	20.920	38.461	43.678
1800	25.759	49.943	55.228
2000	32.119	60.332	73.463
2200	53.422	96.386	96.399



4.3 Wnioski

Operacja zliczania łuków powrotnych sprawdza dla każdego łuku (u, v) w grafie czy jest spełniony warunek $b[v] < b[u] < f[u] < f[v]$, gdzie $b[x]$ i $f[x]$ to etykiety rozpoczęcia i zakończenia analizy wierzchołka x obliczone przez algorytm sortowania.

Teoretycznie najlepszą reprezentacja grafu do zliczania łuków powrotnych powinna być lista łuków, ponieważ zapewnia najszybszy dostęp do każdego łuku, przy złożoności przeglądania zbioru łuków $O(m)$. Najgorszym

wyborem powinna być macierz sąsiedztwa, gdzie operacja przeglądania zbioru łuków, ma największą złożoność ze wszystkich badanych reprezentacji $O(n^2)$, ponieważ w tej reprezentacji trzeba sprawdzić nie tylko istniejące łuki, ale wszystkie możliwe.

Z przeprowadzonych badań wynika, że najlepszą reprezentacją grafu, do zliczania łuków powrotnych, jest macierz sąsiedztwa, mimo swojej teoretycznie największej złożoności przeszukiwania zbioru łuków. Stało się tak dlatego, że ta reprezentacja ma formę tablicy dwuwymiarowej, gdzie do wierzchołków odwołujemy się za pomocą indeksów, co jest szybsze od odwoływania się po wskaźniku jak w pozostałych reprezentacjach. Mimo większej złożoności przeszukiwania zbioru łuków ($O(n+m)$) od listy łuków, szybsza okazała się lista następników, najprawdopodobniej dlatego, że lista łuków przechowywana jest w liście, a lista następników w tablicy list, przez co w tej reprezentacji do pierwszego wierzchołka możemy dostać się po indeksie, a nie po wskaźniku.

Można zauważyć, że wraz z wzrostem gęstości grafu różnica między czasem zliczania łuków w macierzy, do reszty reprezentacji, pogłębia się ponieważ przy większej gęstości musimy wykonać więcej operacji porównań, przy czym odwołać się do większej liczby wskaźników.

Warto również wspomnieć, że przy grafie o gęstości $d = 1$ przeglądanie zbioru łuków w każdej reprezentacji miało by złożoność $O(n^2)$, a różnice w czasie zliczania łuków powrotnych wynikały by tylko z czasu odwołania się do poszczególnych wierzchołków.

5 Porównania poznanych reprezentacji grafu (macierzy sąsiedztwa, listy następników, listy poprzedników, listy łuków, macierzy incydencji)

5.1 Złożoność pamięciowa.

Ze wszystkich poznanych struktur najmniej miejsca zajmuje lista łuków O(m), następnie lista poprzedników oraz następników O(n+m). Większe zapotrzebowanie na ten zasób ma macierz sąsiedztwa O(n^2). Najgorzej wypada macierz incydencji O(n*m), pomimo tak dużych wymagań ma ona jednak dość dużą zaletę, tylko ta forma będzie potrafiła w pełni zaprezentować hipergraf. Jednakże w przypadku grafów rzadkich będzie to duża wada.

5.2 Test łuku.

Najszybszą pod tym względem okazuje się macierz grafu O(1), dzięki sprawdzaniu tylko jednej wartości w strukturze. Drugą pod tym względem jest lista łuków i macierz incydencji O(m). Słabiej pod tym względem wypada lista następników i poprzedników O(n).

5.3 Sprawdzanie następników.

Najbardziej wydajną okazuje się lista następników i macierz grafu O(n). Warto by dodać, że dla tylko dla przypadku pesymistycznego czas tych dwóch struktur jest taki sam, w każdym innym lista następników jest szybsza. Wolniejsze okazuje się lista łuków i lista poprzedników z kolejno O(m) i O(n+m). Najwolniejsza okazuje się macierz incydencji O(n*m).

5.4 Sprawdzanie poprzedników.

Najlepszą reprezentacją do tego testu okazuje się lista poprzedników O(n). Drugą co do wydajności jest macierz sąsiedztwa. Wolniejsza okazuje się lista łuków O(m) oraz lista następników O(n+m). Po raz kolejny najmniej wydają okazuje się macierz incydencji O(m*n).

5.5 Zbiór łuków.

Optymalną strukturom do tego testu okazuje się lista łuków O(m). Druga co do wydajności jest lista następników i poprzedników O(m+n). Bardzo słabo wypada macierz sąsiedztwa O(n^2). Najgorzej jednak z tym zadaniem radzi sobie macierz incydencji O(n*m).

Spis treści