Sprawozdanie nr 3
22.04.2022, Ćwiczenia nr 5.
Reprezentacje grafu i algorytmy DFS, BFS, sortowania topologicznego.

ALGORYTMY I STRUKTURY DANYCH semestr letni, rok akademicki 2021/2022 piątek 9:45-11:15

1. Andrei Kartavik, Indeks: 153925
2. Ivan Kaliadzich, Indeuks: 153936

Wstęp

Celem tego sprawozdania jest porównanie różnych reprezentacji grafu (macierzy sąsiedztwa, listy następników, listy poprzedników, listy łuków, macierzy incydencji) na przykładzie metody sortowania topologicznego.

Pomiary były wykonywane dla *n* wierzchołków o wartościach z zakresu [0, n], zaś wielkości *n* od 1000 elementów do 10 000 elementów, z krokiem 1 000 (10 punktów pomiarowych). Gęstości grafu *d,* o wielkości 0,2 i 0,4; oraz *m* liczbie łuków, gdzie $m = n * n * d$.

Metoda sortowania topologicznego

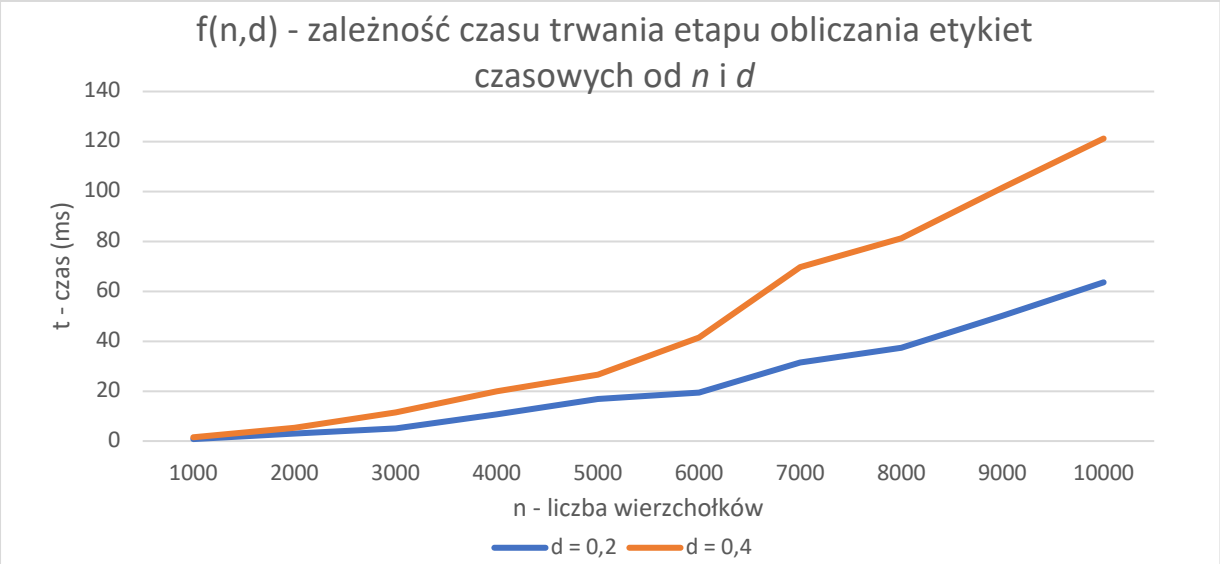
Sortowanie topologiczne grafu skierowanego polega na utworzeniu listy wierzchołków grafu w taki sposób, aby każdy wierzchołek posiadający sąsiadów znalazł się na tej liście przed nimi. W naszym przypadku metoda sortowania topologicznego polega na algorytmie DFS i działa w 2 wyodrębnionych etapach:

1) Obliczanie dla każdego wierzchołka etykiety czasowe rozpoczęcia i zakończenia analizy.
2) Sprawdzenie acykliczności przez zliczanie łuków powrotnych, gdzie łukiem powrotnym jest łuk (v, u), który spełnia warunek in[u] < in[v] < out[v] < out[u], a *in* i *out* są etykietami czasowymi rozpoczęcia i zakończenia analizy.

Czas trwania etapu obliczenia etykiet od liczby wierzchołków *n*

n – liczba wierzchołków	**d - gęstość**	**f(n,d) – zależność czasu trwania etapu obliczania etykiet czasowych od *n* I *d* (ms)**
1000	0,2	0,797
	0,4	1,522
2000	0,2	3,011
	0,4	5,384
3000	0,2	5,014
	0,4	11,563
4000	0,2	10,757
	0,4	19,915
5000	0,2	16,802
	0,4	26,635
6000	0,2	19,341
	0,4	41,416
7000	0,2	31,586
	0,4	69,786
8000	0,2	37,312
	0,4	81,26
9000	0,2	50,241
	0,4	101,509
10000	0,2	63,579
	0,4	121,169

Tablica 1: Zależność czasu trwania etapu obliczania etykiet czasowych od n i d (ms)



Jak już było powiedziano, w naszym przypadku metoda sortowania topologicznego polega na algorytmie przejścia DFS. Podczas wykonania tego algorytmu, gdy odwiedzamy kolejny wierzchołek, nadajemy mu numer wejścia – numer kroku, w którym ten wierzchołek został odwiedzony. Gdy wejdziemy do wierzchołka, który nie ma następników, lub którego następniki już zostały odwiedzone wcześniej, to nadajemy mu numer wyjścia i cofamy się do poprzedniego wierzchołka, powtarzając tą operację rekurencyjnie do momentu, aż wszystkie wierzchołki nie zostaną odwiedzone. I w wyniku, żeby otrzymać porządek sortowania topologicznego, wystarczy odwrócić kolejność numerów wyjść wierzchołków. Wynika to z tego, że przy wykonaniu DFS wierzchołki „najgłębsze" otrzymują numery wyjścia pierwszymi. Ta kolejność ma sens tylko w tych przypadkach, gdy graf jest skierowany i acykliczny (graf acykliczny, to taki graf, który nie zawiera krawędzie powrotne, czyli krawędzie (v, u) łączące wierzchołek *v* z jego przodkiem *u* w grafie).

Ponieważ nasza metoda sortowania topologicznego opiera się na algorytmie DFS z dodatkową operacją numeracji wierzchołków i operacją końcowego odwracania kolejności numerów wyjść, które mają liniową złożoność czasową, to ma ona taką samą złożoność czasową, jak i sam DFS, czyli O(n+m), gdzie n – to liczba wierzchołków, a m – to liczba łuków. Z tego powodu, że przy odwiedzaniu każdego wierzchołka jesteśmy zainteresowani zbiorem go następników, więc bardzo ważna dla nas jest złożoność wyznaczenia zbioru następników i dlatego metoda sortowania topologicznego bardzo zależy od reprezentacji grafu. Najlepszymi reprezentacjami do tego są lista następników, dla której złożoność wyznaczenia zbioru następników wynosi O(n), a średnio nawet O(m/n), oraz macierz sąsiedztwa, dla której złożoność wyznaczenia zbioru następników również wynosi O(n). Przy testowaniu zależności czasu trwania etapu obliczania etykiet skorzystaliśmy z listy następników, zamiast macierzy sąsiedztwa, z tego powodu, że przypadek średni listy następników jest lepszy. A najgorszą reprezentacją do metody sortowania topologicznego opartej o DFS jest macierz incydencji, dla której złożoność wyznaczenia zbioru następników w najgorszym przypadku może wynosić aż O(n*m).

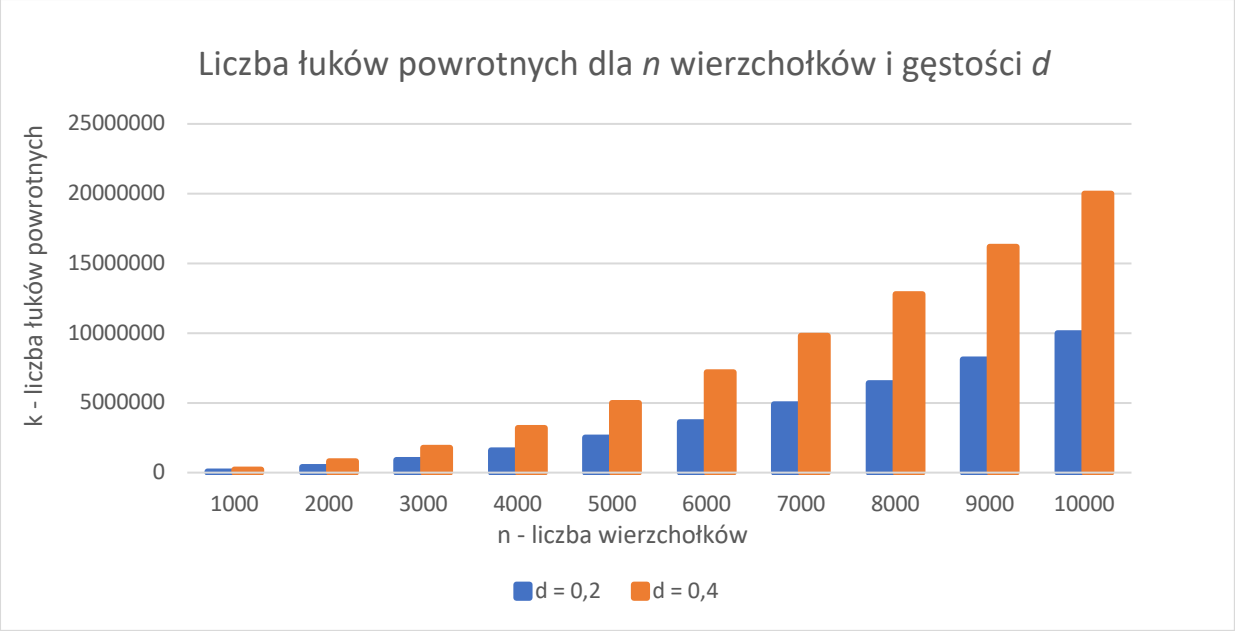
Gęstością grafu skierowanego nazywamy stosunek liczby łuków do największej możliwej liczby łuków, $d = m/n(n-1)$. Stąd możemy wyciągnąć, że liczba łuków m w grafie, przy zadanej gęstości, jest $\approx n * n * d$. Dlatego, im większa jest gęstość grafu, tym większa jest liczba łuków w tym grafie. Stąd i wynika, jak możemy zobaczyć w tabeli i na wykresie, że gęstość grafu ma duży wpływ na czas obliczania etykiet czasowych, bo jak możemy przypomnieć, złożoność czasowa metody sortowania topologicznego wynosi O(n+m), więc dla większej

d mamy większą *m* i mamy dłuższy czas wykonania algorytmu. Dla grafu pełnego (d = 1) złożoność czasowa wynosiłaby aż $O(n^2)$, a dla m = n zaledwie O(n). Dla grafu reprezentowanego przez listę następników, listę poprzedników lub listę łuków jest to najlepiej widoczne z uwagi na to, że im większa jest gęstość grafu, tym większa jest struktura tych reprezentacji, natomiast najsłabiej widoczne by to było dla macierzy sąsiedztwa, dla której rozmiar struktury zależy wyłącznie od liczby wierzchołków.

Liczba łuków powrotnych dla poszczególnych wartości liczby wierzchołków *n* i gęstości *d*

n - liczba wierzchołków	**d - gęstość**	**k - liczba łuków powrotnych**
1000	0,2	100012
	0,4	199471
2000	0,2	399413
	0,4	799347
3000	0,2	900792
	0,4	1795191
4000	0,2	1597781
	0,4	3191120
5000	0,2	2496815
	0,4	4991693
6000	0,2	3599990
	0,4	7194307
7000	0,2	4896219
	0,4	9800733
8000	0,2	6397556
	0,4	12800394
9000	0,2	8096137
	0,4	16188099
10000	0,2	9995949
	0,4	19993008

Tablica 2: Liczba łuków powrotnych dla poszczególnych wartości liczby wierzchołków n i gęstości d



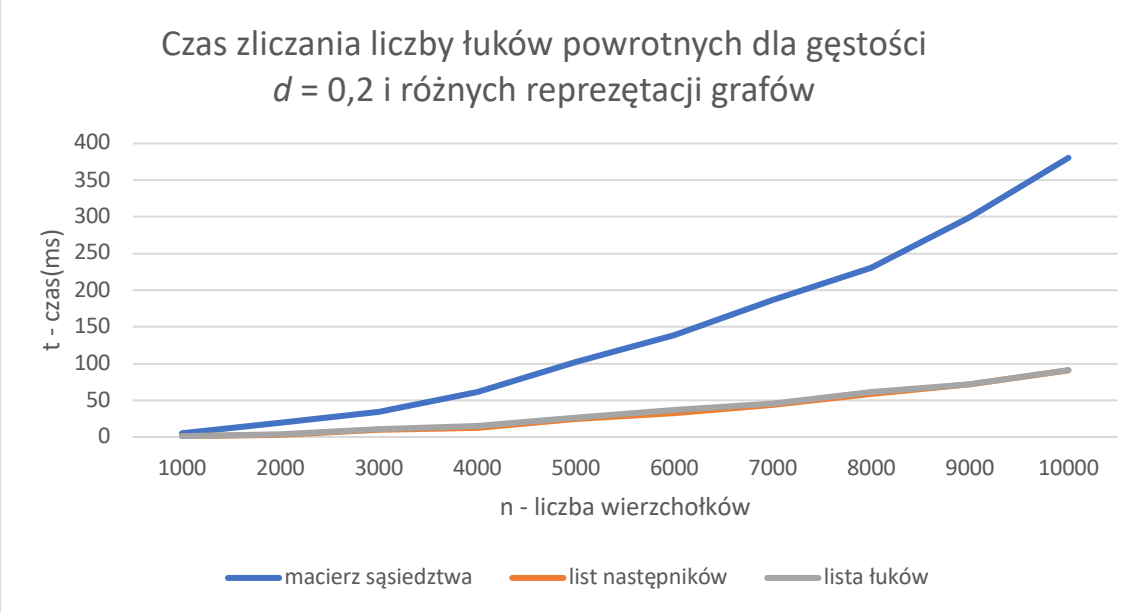
Jak już ustaliliśmy, kolejność sortowania topologicznego ma sens tylko w tym przypadku, gdy graf jest skierowany i acykliczny (graf acykliczny, to taki graf, który nie zawiera krawędzie powrotne, czyli krawędzie (v, u) łączące wierzchołek *v* z jego przodkiem *u* w grafie), bo jeśli wyobrazimy sobie taką sytuację, że chcemy uporządkować trzy zadania A, B i C, a nasz graf wyglądałby tak: A->B->C->A, to nie bylibyśmy w stanie ułożyć nasze zadania w logicznym porządku, bo nie moglibyśmy stwierdzić, jakie zadanie musiałoby być pierwszym.

Z tego powodu, że generujemy całkiem losowy graf, to pojawiają się w nim cykli i jak możemy zobaczyć w tablice, każdy z testowanych grafów był cykliczny. Oraz możemy ustalić, że im większa była gęstość, tym więcej było łuków w grafie, a im więcej jest łuków, tym więcej cykli może pojawić się w grafie.

Zależność czasu trwania etapu zliczania łuków powrotnych od liczby wierzchołków n

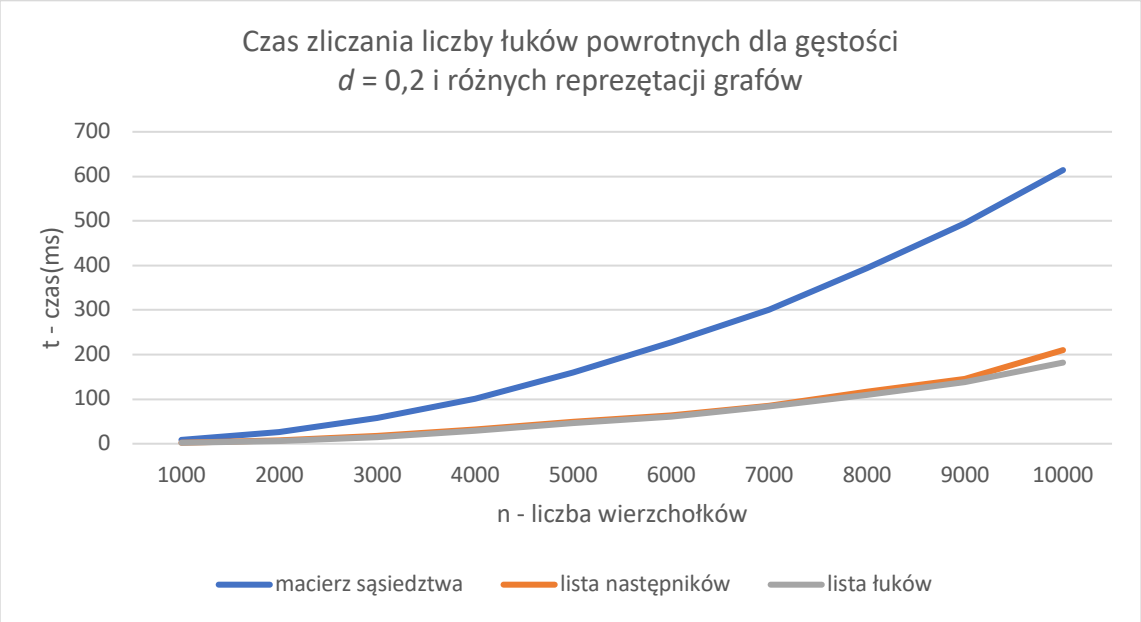
Gęstość *d* = 0,2			
n - liczba wierzchołków	**Czas zliczania liczby łuków powrotnych dla macierzy sąsiedztwa (ms)**	**Czas zliczania liczby łuków powrotnych dla listy następników (ms)**	**Czas zliczania liczby łuków powrotnych dla listy łuków (ms)**
1000	5,112	1,087	1,131
2000	19,296	3,78	3,129
3000	33,944	10,325	9,569
4000	61,158	15,401	12,833
5000	101,947	25,971	24,31
6000	138,715	36,648	32,447
7000	187,308	45,863	43,421
8000	230,865	61,397	58,325
9000	299,641	71,932	71,663
10000	380,393	91,181	90,761

Tablica 3: Czas zliczania liczby łuków powrotnych dla n wierzchołków, gęstości d = 0,2 i różnych reprezentacji grafów



Gęstość *d* = 0,4			
n - liczba wierzchołków	**Czas zliczania liczby łuków powrotnych dla macierzy sąsiedztwa (ms)**	**Czas zliczania liczby łuków powrotnych dla listy następników (ms)**	**Czas zliczania liczby łuków powrotnych dla listy łuków (ms)**
1000	8,567	2,158	2,029
2000	25,676	7,045	6,109
3000	58,339	16,857	15,281
4000	100,639	31,401	29,48
5000	160,407	49,157	46,359
6000	227,258	63,417	61,379
7000	300,459	85,671	83,374
8000	393,629	116,922	109,72
9000	494,257	145,625	138,86
10000	614,162	210,145	182,085

Tablica 4: Czas zliczania liczby łuków powrotnych dla n wierzchołków, gęstości d = 0,4 i różnych reprezentacji grafów



Jak już powiedzieliśmy, łukiem powrotnym jest łuk (v, u), który spełnia warunek in[u] < in[v] < out[v] < out[u], a *in* i *out* są etykietami czasowymi rozpoczęcia i zakończenia analizy, czyli to taki łuk (v, u), który łączy wierzchołek *v* z jego przodkiem *u* w grafie. Operacja zliczania łuków powrotnych analizuje zbiór wszystkich łuków grafu, sprawdzając dla każdego łuku, czy spełnia on powyższy warunek, tym samym licząc, ile w tym zbiorze jest łuków powrotnych.

Możemy zobaczyć na wykresach, że najlepszą reprezentacją dla operacji wyznaczenia zbioru łuków jest lista łuków, która jest trochę lepsza, niż lista następników. Złożoność tej operacji dla listy łuków wynosi O(m), gdy lista następników dla tej samej operacji ma złożoność O(m+n). Różnica może być nie taka znaczna z tego powodu, że liczba wierzchołków *n* jest odnośnie mała do liczby łuków *m* i dlatego złożoność czasowa dla dwóch reprezentacji będzie mniej więcej taka sama. Najgorszą strukturą z podanych w tablicach jest macierz sąsiedztwa, dla której wyznaczenie zbioru łuków wymaga przechodzenia przez całą macierz, co wynosi $O(n^2)$.

Porównując dwa wykresy i znaczenia w podanych tabelach, możemy dojść do wniosku, że gęstość *d* ma wpływ na czas zliczania łuków powrotnych. Wynika to z tego, że im większe jest *d,* tym więcej łuków występuje w grafie, a więc i zbiór łuków jest większy. I jak już stwierdziliśmy, wpływ gęstości grafu na czas działania zależy od reprezentacji grafu.

Porównanie poznanych reprezentacji grafu

Ze względu na złożoność pamięciową najlepszą reprezentacją jest lista łuków – O(m). Lista poprzedników i lista następników też są dobrym wyborem ze względu na pamięć, ich złożoność pamięciowa wynosi O(n+m), a najgorszymi reprezentacjami są macierz sąsiedztwa – $O(n^2)$ i macierz incydencji – O(n*m). Dlatego, jeśli mamy ograniczone źródła pamięciowe, lepiej wybrać listę łuków lub listę następników/poprzedników.

Dla testu łuku najlepszą reprezentacją jest macierz sąsiedztwa, dla której złożoność tej operacji wynosi O(1), czyli natychmiastowe. Lista następników i lista poprzedników mają złożoność O(n), a lista łuków i macierz incydencji mają złożoność O(m). Dlatego, jeśli nas interesuję ciągle sprawdzanie istnienia łuku, najlepszym wyborem jest macierz sąsiedztwa, a wybór listy poprzedników/następników lub listy łuków/macierzy incydencji zależy od tego, jaka duża jest różnica między wartością *n* i *m.*

Dla operacji wyznaczenia zbioru następników, jak mogliśmy zobaczyć na wykresach wyżej, najlepszymi reprezentacjami są lista następników – O(n) w przypadku najgorszym i O(m/n) w przypadku średnim, i macierz sąsiedztwa – O(n). Dobrym wyborem jest lista łuków, dla której złożoność tej operacji wynosi O(m), a najgorszym wyborem jest lista poprzedników, dla której wyznaczenie zbioru następników wymagałoby przechodzenia przez całą strukturę, więc ona ma złożoność O(n+m), oraz macierz incydencji, z najgorszą dla tej operacji złożonością – O(n*m).

Dla operacji wyznaczenia zbioru poprzedników sytuacja jest bardzo podobna do operacji wyznaczenia zbioru następników. Macierz sąsiedztwa, lista łuków i macierz incydencji mają taką samą złożoność, jak i dla powyższej operacji… Różnica jest w tym, że dla listy poprzedników złożoność wyznaczenia zbioru poprzedników wynosi O(n) w przypadku najgorszym, oraz O(m/n) w przypadku średnim, co i robi tą reprezentację najlepszą dla tej operacji. A dla listy następników złożoność ta wynosi O(n+m), co robi ją jedną z najgorszych reprezentacji dla tej operacji.

Ze względu na operację wyznaczenia zbioru łuków, najlepszą reprezentacją jest lista łuków – O(m). Dobrym wyborem jest również lista następników i lista poprzedników, dla których złożoność tej operacji wynosi O(n+m).
W przypadkach, kiedy *n* jest odnośnie mała do *m*, jak mogliśmy zobaczyć na powyższych wykresach, czas wykonania operacji wyznaczenia zbioru łuków dla tych reprezentacji jest prawie taki sam, jak i dla listy łuków. A najgorszymi reprezentacjami dla tej operacji są macierz sąsiedztwa – $O(n^2)$, oraz macierz incydencji – O(n*m), dla których wyznaczenie zbioru łuków wymaga przechodzenie przez całą macierz.

Wnioski

Macierz sąsiedztwa jest bardzo uniwersalną reprezentacją grafu. Jest ona prostą w implementacji, najlepszą ze względu na test łuku, dobrą dla operacji wyznaczenia zbioru następników i zbioru poprzedników, ale jest ona jedną z najgorszych reprezentacji, ze względu na złożoność pamięciową i ze względu na operację wyznaczenia zbioru łuków. Dlatego, warto ją stosować, gdy potrzebujemy częstego sprawdzania istnienia łuku, chcemy efektywnie wyznaczać zbiór następników i zbiór poprzedników, nie zależy nam na wyznaczanie zbioru łuków i nie jesteśmy ograniczeni ze względu na źródła pamięciowe.

Lista następników jest bardzo efektywną reprezentacją grafu. Jest lepsza ze względu na pamięć niż macierz sąsiedztwa i macierz incydencji. Jest gorsza niż macierz sąsiedztwa dla operacji testu łuku, ale lepsza, niż lista łuków i macierz incydencji. Jest najlepszą reprezentacją dla wyznaczenia zbioru następników, ale jedną z najgorszych dla wyznaczania zbioru poprzedników oraz jest ona jedną z najlepszych reprezentacji dla operacji wyznaczenia zbioru łuków. Dlatego, warto ją używać, kiedy głównym naszym celem jest wyznaczenie zbioru następników, jak na przykład w metodzie sortowania topologicznego lub kiedy jesteśmy trochę ograniczeni ze względu na pamięć i nie jesteśmy zainteresowani wyznaczeniem zbioru poprzedników.

Lista poprzedników zarówno jest efektywną reprezentacją grafu. Jest prawie taka sama, jak i lista następników, oprócz operacji wyznaczenia zbioru następników i zbioru poprzedników. Jest najlepszą reprezentacją dla wyznaczenia zbioru poprzedników i jedną z najgorszych reprezentacji dla wyznaczenia zbioru następników. Dlatego warto korzystać z niej, kiedy

głównym naszym celem jest wyznaczenie zbioru poprzedników i nie jesteśmy zainteresowani wyznaczeniem zbioru następników.

Lista łuków jest efektywną reprezentacją grafu. Jest najlepszą ze względu na złożoność pamięciową, ale jest ona gorszą reprezentacją dla operacji testu łuku w porównaniu z powyższymi reprezentacjami. Jest ona dobra dla operacji wyznaczenia zbioru następników i zbioru poprzedników i jest najlepszą strukturą dla wyznaczenia zbioru łuków. Dlatego warto ją stosować, gdy głównym naszym celem jest wyznaczenie zbioru łuków, kiedy jesteśmy zainteresowani wyznaczeniem zbioru poprzedników i zbioru następników lub w sytuacjach, kiedy jesteśmy ograniczeni ze względu na pamięć.

Macierz incydencji jest najgorszą reprezentacją grafu ze względu na efektywność wszystkich operacji i ze względu na złożoność pamięciową, oraz jest dosyć skomplikowana w implementacji. Ale wykorzystuje się ona dla rozwiązania różnych problemów teorii grafów, na przykład: 1) Twierdzenie o macierzach, które mówi, że liczba drzew opinających grafu jest równa mniejszej z Laplace'a grafu, który jest macierzą ściśle powiązaną z macierzą incydencji. 2) Rozważanie przepływów na wykresie, np. przepływ prądu w obwodzie elektrycznym i związane z nim potencjały. Zarówno jest ona używana w teorii grafów spektralnych.