

Temat: Struktury danych do reprezentacji grafów. Wybrane algorytmy grafowe.

Oznaczenia

- $G = \langle V, E \rangle$ - graf bez wag, gdzie V - zbiór wierzchołków, E - zbiór krawędzi
- $|V| = n$ - liczba wierzchołków grafu G
- $|E| = m$ - liczba krawędzi grafu G
- (i, j) - krawędź (para uporządkowana) z wierzchołka i do wierzchołka j w grafie skierowanym
- $\{i, j\}$ - krawędź (podzbiór) łącząca wierzchołki i oraz j w grafie nieskierowanym
- $G = \langle V, E, f \rangle$ - graf z wagami, gdzie V - zbiór wierzchołków, E - zbiór krawędzi, $f: V \rightarrow R$

Założenia

- Graf skierowany nie ma pętli, czyli dla każdej krawędzi (i, j) grafu zachodzi $i \neq j$,
- Wierzchołki grafu są reprezentowane numerami: $1, 2, \dots, n$

1. Struktury danych do reprezentacji grafu bez wag

a) macierz sąsiedztwa

Definiujemy tablicę dwuwymiarową G o rozmiarach $n \times n$. Jeżeli graf jest nieskierowany to:

$$G[i, j] = \begin{cases} 0 & \text{dla } \{i, j\} \notin E \\ 1 & \text{dla } \{i, j\} \in E \end{cases}$$

Przykład

G :

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 |

Złożoność pamięciowa: n^2

Macierz sąsiedztwa jest symetryczna względem głównej przekątnej dla grafu nieskierowanego.

- Jeżeli graf jest skierowany to:

$$G[i, j] = \begin{cases} 0 & \text{dla } (i, j) \notin E \\ 1 & \text{dla } (i, j) \in E \end{cases}$$

Przykład

G :

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 | 1 |
| 5 | 1 | 1 | 0 | 0 | 0 |

Złożoność pamięciowa: n^2

Wady:

- duża złożoność pamięciowa dla grafów rzadkich (tj. takich dla których $m \ll n^2$),
- nieefektywna realizacja algorytmów grafowych wymagających dotarcia do wszystkich węzłów incydentnych z danym węzłem.

Zalety:

- efektywna realizacja (stały koszt) dodawania i usuwania krawędzi grafu,
- efektywna realizacja (stały koszt) sprawdzenia, czy w grafie jest dana krawędź

b) Listy incydencji

Strukturą danych jest w tym przypadku jednowymiarowa tablica wskaźników na listy wierzchołków incydentnych. W indeksie i takiej tablicy znajduje się adres pierwszego elementu listy zawierającej numery tych wierzchołków grafu, które incydują z wierzchołkiem numer i . Rozmiar tablicy jest równy n .

Przykład

Przykład dla grafu nieskierowanego, T – tablica incydencji

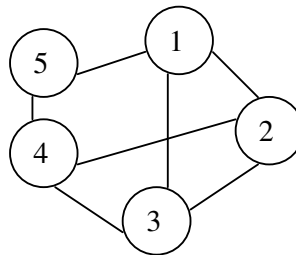
T[1]: 2, 3, 5

T[2]: 1, 3, 4

T[3]: 1, 2, 4

T[4]: 2, 3, 5

T[5]: 4, 1



Przykład grafu skierowanego

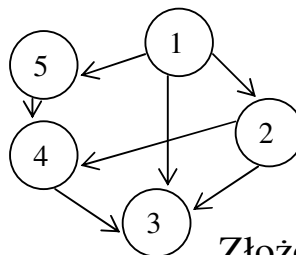
T[1]: 2, 3, 5

T[2]: 3, 4

T[3]:

T[4]: 3

T[5]: 4



Złożoność pamięciowa: $n+m$

Wady:

- nieefektywna realizacja dodawania (usuwania) krawędzi grafu (Trzeba sprawdzić całą listę incydencji, aby ustalić, czy krawędzi wstawianej (usuwanej) nie ma już (jest) w grafie.
- nieefektywna realizacja sprawdzenia, czy w grafie jest dana krawędź (Trzeba przejrzeć całą listę incydencji).
- struktura trudniejsza w realizacji

Zalety:

- efektywna złożoność pamięciowa dla skierowanych grafów rzadkich ,
- efektywna realizacja algorytmów grafowych wymagających dotarcia do wszystkich węzłów incydentnych z danym węzłem.

2. Struktury danych do reprezentacji grafu z wagami

a) macierz sąsiedztwa

Definiujemy tablicę dwuwymiarową G o rozmiarach $n \times n$. Jeżeli graf jest skierowanego to:

$$G[i, j] = \begin{cases} \infty & \text{dla } (i, j) \notin E \\ f((i, j)) & \text{dla } (i, j) \in E \end{cases}$$

Wartość ∞ ustalana jest indywidualnie dla konkretnego grafu i konkretnego algorytmu grafowego. Na przykład dla problemu najkrótszych ścieżek $\infty \geq n \cdot f_{\max}$, gdzie f_{\max} to maksimum funkcji wag.

Przykład

$G: \infty \geq 12 \cdot 5$

| | 1 | 2 | 3 | 4 | 5 |
|---|----------|----------|----------|----------|----------|
| 1 | ∞ | 5 | 3 | ∞ | 4 |
| 2 | 3 | ∞ | -6 | 8 | 2.5 |
| 3 | 4 | 12 | ∞ | 11 | ∞ |
| 4 | 0 | 6 | 1 | ∞ | 1 |
| 5 | 1.3 | 8 | ∞ | 4 | ∞ |

Złożoność pamięciowa: n^2

Analogicznie definiujemy macierz sąsiedztwa dla nieskierowanego grafu z wagami.

c) Listy incydencji

Strukturą danych jest w tym przypadku jednowymiarowa tablica wskaźników na listy wierzchołków incydentnych. W indeksie i takiej tablicy znajduje się adres pierwszego elementu listy zawierającej numery tych wierzchołków grafu, które incydują z wierzchołkiem numer i oraz wagi tych krawędzi. Rozmiar tablicy jest równy n .

3. Algorytmy bazowe dla grafów bez wag i ich przykładowe zastosowanie

a) Przeglądania grafu metodą „wszerz” (*breadth-first search* (BFS))

WP: G – graf bez wag, s - wierzchołek startowy

WK: Odwiedzenie każdego wierzchołka grafu G , do którego istnieje co najmniej jedna ścieżka z s .

Struktura do reprezentacji grafu G - tablica struktur z polami:

- *link* - wskaźnik na listę incydencji,
- *color* – typ z trzema wartościami: *biały*, *szary*, *czarny*,

Wskaźnik listy incydencji zawiera pola:

- *nr* – numer wierzchołka
- *next* – adres kolejnego elementu listy

Struktura pomocnicza – kolejka z operacjami: *in*, *out*, *first*, *empty*.

(*in*(e, Q)- wstawia do kolejki Q element e , *out*(Q) – usuwa element e z kolejki Q , *first*(Q) – zwraca dane z pierwszego elementu niepustej kolejki Q , *empty*(Q) – sprawdza, czy kolejka Q jest niepusta)

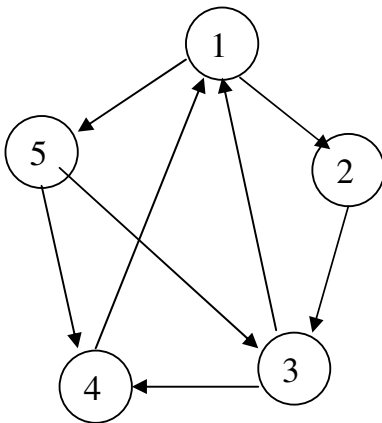
Algorytm

I Krok: Inicjalizacja tablicy grafu

Przykład

G:

| | | | | | |
|----------------|--------------|--------------|--------------|--------------|--------------|
| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
| <i>color</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> |
| <i>link</i> | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



II Krok: Przeglądanie grafu z wierzchołka *s*

BFS(*G*, *s*)

```
G[s].color = szary;  
Q = ∅;  
in(s, Q);  
while (!empty(Q))  
{  
  u = first(Q);  
  temp = G[u].link;  
  while (temp != NULL)  
  {  
    v = temp->nr;  
    if (G[v].color == biały)  
    {
```

```

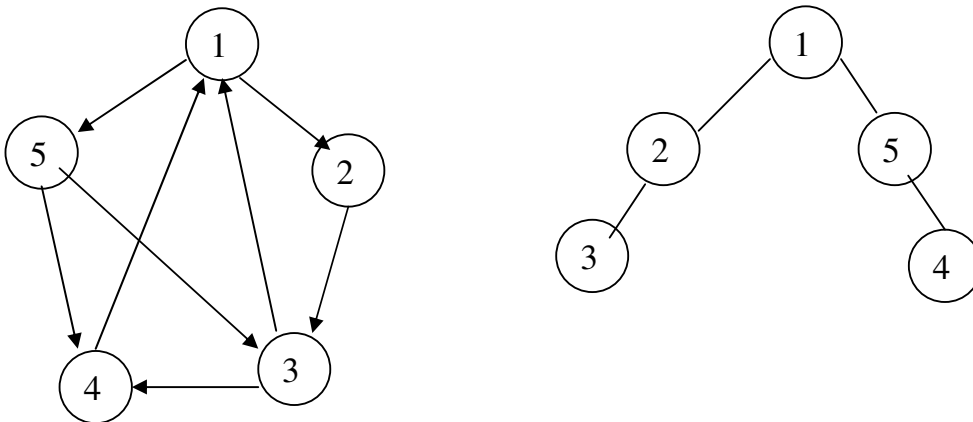
// odwiedzenie wierzchołka v
G[v].color = szary;
in(v,Q);
}
temp = temp->next;
}
out(Q);
G[u].color = czarny;
}}

```

Przykład

G:

| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
|----------------|--------|--------|--------|--------|--------|
| color | czarny | czarny | czarny | czarny | czarny |
| Link | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



b) Zastosowanie przeglądania grafu metodą „wszerz” (*breadth-first search* (BFS)) do ustalania najkrótszych ścieżek w grafie bez wag

WP: G – graf bez wag, s - wierzchołek startowy

WK: najkrótsze ścieżki z wierzchołka s do pozostałych wierzchołków grafu

Struktura do reprezentacji grafu G - tablica struktur z polami:

- *link* - wskaźnik na listę incydencji,
- *color* – typ z trzema wartościami: *biały*, *szary*, *czarny*,
- *d* – odległość od wierzchołka startowego,

- pi – poprzednik na ścieżce

Wskaźnik listy incydencji zawiera pola:

- nr – numer wierzchołka
- $next$ – adres kolejnego elementu listy

Struktura pomocnicza – kolejka z operacjami: in , out , $first$, $empty$.

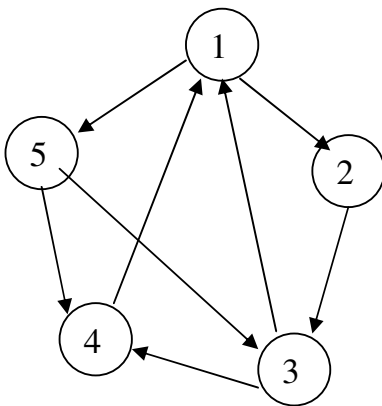
($in(e,Q)$ - wstawia do kolejki Q element e , $out(Q)$ – usuwa element e z kolejki Q , $first(Q)$ – zwraca dane z pierwszego elementu niepustej kolejki Q , $empty(Q)$ – sprawdza, czy kolejka Q jest niepusta)

Algorytm

I Krok: Inicjalizacja tablicy grafu

Przykład G :

| | | | | | |
|----------------|--------------|--------------|--------------|--------------|--------------|
| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
| $color$ | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> |
| D | ∞ | ∞ | ∞ | ∞ | ∞ |
| pi | -1 | -1 | -1 | -1 | -1 |
| $link$ | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



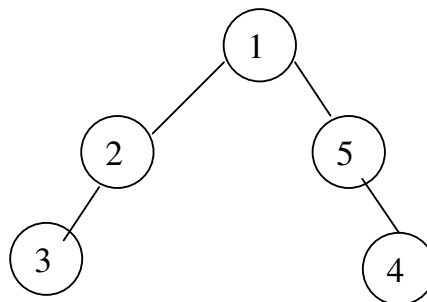
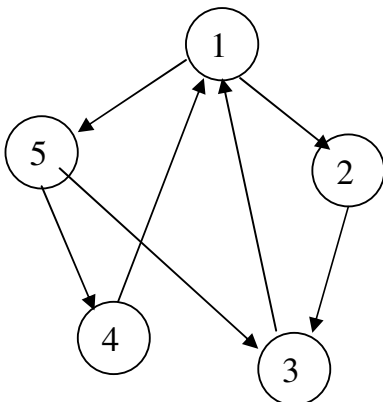
II Krok: Przeglądanie grafu z wierzchołka s z jednoczesnym ustawianiem etykiet d i pi

BFS(G, s)

```
 $G[s].color = szary;$   
 $G[s].d = 0;$   
 $Q = \emptyset;$   
 $in(s, Q);$   
while ( $!empty(Q)$ ) {  
   $u = first(Q);$   
   $temp = G[u].link;$   
  while ( $temp \neq NULL$ ) {  
     $v = temp \rightarrow nr;$   
    if ( $G[v].color == biały$ )  
    {  
       $G[v].color = szary;$   
       $G[v].d = G[u].d + 1;$   
       $G[v].pi = u;$   
       $in(v, Q);$   
    }  
     $temp = temp \rightarrow next;$   
  }  
   $out(Q);$   
   $G[u].color = czarny;$   
}
```

Przykład G :

| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
|----------------|---------------|---------------|---------------|---------------|---------------|
| <i>color</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> |
| <i>d</i> | 0 | 1 | 2 | 2 | 2 |
| <i>pi</i> | -1 | 1 | 2 | 5 | 1 |
| <i>link</i> | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



Złożoność czasowa BFS

Jako operację elementarną przyjmujemy wykonanie operacji *in* lub *out* na kolejce Q oraz operację przeglądania list incydencji grafu.

Każdy wierzchołek jest wstawiany co najwyżej raz i co najwyżej raz jest z niego usuwany. Stąd łączny czas wykonania operacji na kolejce wynosi $O(n)$. Ponieważ lista incydencji każdego wierzchołka jest przeglądana tylko wtedy, gdy wierzchołek zostanie usunięty z kolejki, lista incydencji każdego wierzchołka jest przeglądana co najwyżej raz. Suma długości wszystkich list wynosi $\Theta(m)$. Łączny czas przeglądania list wynosi zatem $O(m)$. Inicjowanie grafu zabiera czas $O(n)$.

Reasumując: koszt BFS wynosi $O(m+n)$.

c)Przełgądania grafu metodą „w głąb” (*depth-first search (DFS)*)

WP: G – graf bez wag, s - wierzchołek startowy

WK: Odwiedzenie każdego wierzchołka grafu G , dla którego istnieje co najmniej jedna ścieżka z s

Struktura do reprezentacji grafu G - tablica struktur z polami:

- *link* - wskaźnik na listę incydencji,
- *color* – typ z trzema wartościami: *biały*, *szary*, *czarny*,

Wskaźnik listy incydencji zawiera pola:

- *nr* – numer wierzchołka
- *next* – adres kolejnego elementu listy

Struktura pomocnicza – stos z operacjami: *push*, *pop*, *top*, *empty*.

(*push(e,S)*- wstawia do stosu S element e , *pop(S)* – usuwa element e ze stosu S , *top(S)* – zwraca dane z wierzchołka niepustego stosu S , *empty(S)* – sprawdza, czy stos S jest niepusty)

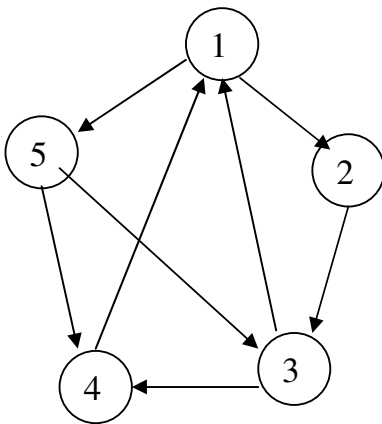
Algorytm

I Krok: Inicjalizacja tablicy grafu

Przykład

G:

| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
|----------------|--------------|--------------|--------------|--------------|--------------|
| <i>color</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> |
| <i>link</i> | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



II Krok: Przeglądanie grafu z wierzchołka *s*

DFS_VISIT(*u*) (funkcja rekurencyjna)

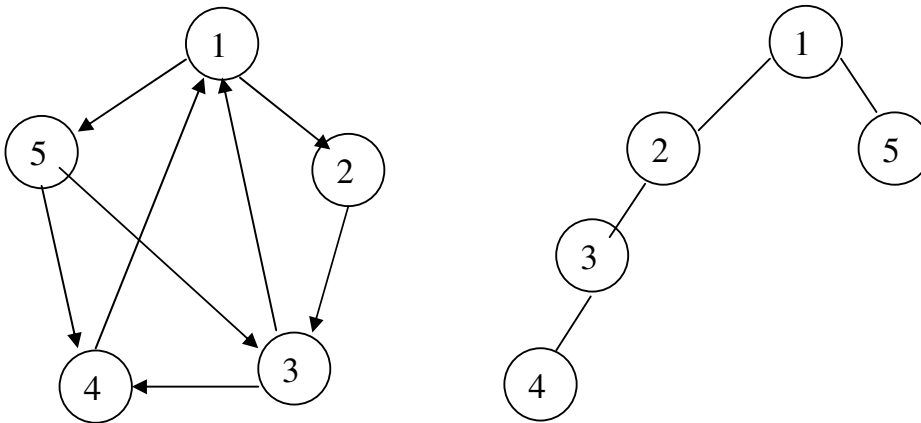
```
G[u].color = szary;  
temp=G[u].link;  
while (temp!=NULL)  
{  
    v=temp->nr;  
    if (G[v].color ==biały )  
    {  
        // odwiedzenie wierzchołka v  
        DFS_VISIT(v);  
    }  
    temp = temp->link;  
}  
G[u].color = czarny;
```

DFS (G, s):
 DFS_VISIT(s);

Przykład

G :

| nr wierzchołka | 1 | 2 | 3 | 4 | 5 |
|----------------|---------------|---------------|---------------|---------------|---------------|
| <i>color</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> | <i>czarny</i> |
| <i>link</i> | 2, 5 | 3 | 1, 4 | 1 | 3, 4 |



d) Zastosowanie przeglądania grafu metodą „w głąb” (*depth-first search* (DFS)) do sortowania topologicznego grafu

WP: G – graf bez wag, skierowany i acykliczny (bez cykli)

WK: Nowa numeracja wierzchołków taka, że jeżeli graf G zawiera krawędź (u, v) , to w tym porządku wierzchołek u występuje przed wierzchołkiem v (wierzchołek u ma mniejszy numer niż wierzchołek v). Numery wierzchołków w nowej numeracji zapamiętane są w stosie wynikowym W .

Jeżeli graf nie jest acykliczny to takie uporządkowanie nie jest możliwe.

Struktura do reprezentacji grafu G - tablica struktur z polami:

- *link* - wskaźnik na listę incydencji,
- *color* – typ z trzema wartościami: *biały*, *szary*, *czarny*,

Wskaźnik listy incydencji zawiera pola:

- *nr* – numer wierzchołka
- *next* – adres kolejnego elementu listy

Struktura pomocnicza – stos z operacjami: *push*, *pop*, *top*, *empty*.

(*push(e,S)*- wstawia do stosu *S* element *e*, *pop(S)* – usuwa element *e* ze stosu *S*, *top(S)* – zwraca dane z wierzchołka niepustego stosu *S*, *empty(S)* – sprawdza, czy stos *S* jest niepusty)

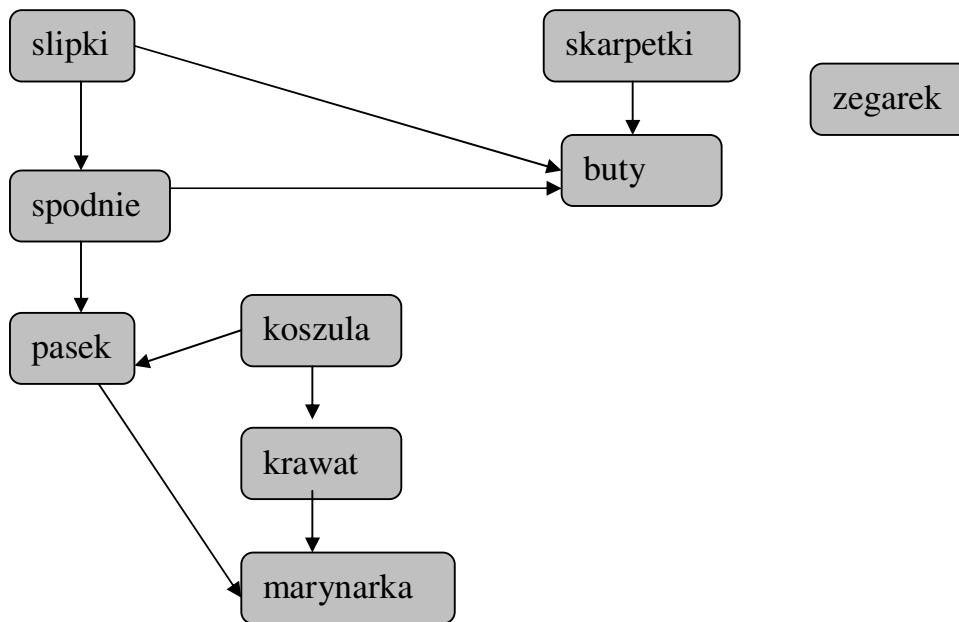
Algorytm

I Krok: Inicjalizacja tablicy grafu

Przykład

Graf *G*:

Krawędź (*u, v*) oznacza, że część garderoby *u* powinna być założona przed częścią garderoby *v*.



| <i>nr wierzchołka</i> | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| <i>dane wierzchołka</i> | skarpetki | slipki | spodnie | Buty | zegarek | koszula | pasek | krawat | marynarka |
| <i>color</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> | <i>biały</i> |
| <i>link</i> | 4 | 3, 4 | 4, 7 | - | - | 7, 8 | 9 | 9 | - |

II Krok: Przeglądanie grafu metodą „w głąb” z wierzchołka startowego s . Wierzchołki, które stają się czarne są wstawiane do stosu wynikowego W . Kolejność wierzchołków w stosie W określa nowa numerację wierzchołków grafu po sortowaniu topologicznym.

DFS_VISIT(u) (funkcja rekurencyjna)

```

W=∅;
G[u].color = szary;
temp=G[u].link;
while (temp!=NULL)
{
    v=temp->nr;
    if (G[v].color ==biały )
    {
        DFS_VISIT(v);
    }
    temp = temp->link;
}
G[u].color = czarny;
push(u,W);
}

```

DFS(G, s);

```

W=∅;
DFS_VISIT(s);
for (i=1;i<=n;i++)
    if (G[i].color==biały) DFS_VISIT(i);

```

Przykład

| | | | | | | | | | |
|-------------------------|-----------|--------|---------|--------|---------|----------|--------|--------|-----------|
| <i>nr wierzchołka</i> | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| <i>dane wierzchołka</i> | skarpetki | slipki | spodnie | buty | zegarek | koszulka | pasek | krawat | marynarka |
| <i>color</i> | czarny | czarny | czarny | czarny | czarny | czarny | czarny | czarny | czarny |
| <i>link</i> | 4 | 3, 4 | 4, 7 | - | - | 7, 8 | 9 | 9 | - |
| <i>nowa numeracja</i> | 2 | 5 | 6 | 7 | 1 | 3 | 8 | 4 | 9 |

Złożoność czasowa DFS

Jako operację elementarną przyjmujemy liczbę wywołań rekurencyjnych procedury DFS_VISIT (operacje stosowe realizowane w stosie wywołań) oraz operację przeglądania list incydencji grafu. Koszt procedury DFS wynosi $O(m+n)$ - uzasadnienie analogiczne jak dla BFS.