Algorytmy Grafowe

dr hab. Bożena Woźna-Szcześniak, prof. UJD

Uniwersytet Jana Długosza w Częstochowie

b.wozna@ujd.edu.pl

Wykład 5 i 6

Spis treści

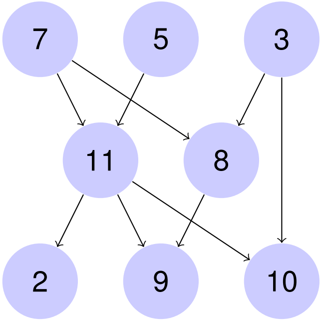
Sortowanie Topologiczne - sformułowanie problemu

- Wejście: Acykliczny graf skierowany $G = (V, E)$, tzw. DAG (ang. *directed acyclic graph*).
- Wyjście: Liniowy porządek wierzchołków z V taki, że jeśli graf G zawiera krawędź (u, v), to w tym porządku wierzchołek u występuje przed wierzchołkiem v.

Sortowanie topologicznie - przykład

Wierzchołki w każdym grafie acyklicznym skierowanym można posortować topologicznie na jeden lub więcej sposobów:

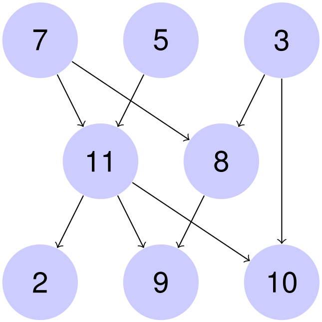
- 7,5,3,11,8,2,9,10



Sortowanie topologicznie - przykład

Wierzchołki w każdym grafie acyklicznym skierowanym można posortować topologicznie na jeden lub więcej sposobów:

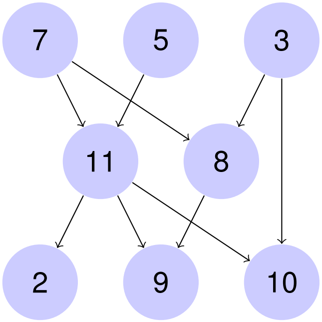
- 7,5,3,11,8,2,9,10
- 7,5,11,2,3,10,8,9



Sortowanie topologicznie - przykład

Wierzchołki w każdym grafie acyklicznym skierowanym można posortować topologicznie na jeden lub więcej sposobów:

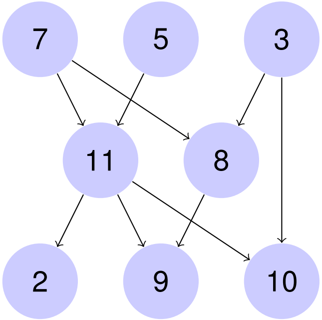
- 7,5,3,11,8,2,9,10
- 7,5,11,2,3,10,8,9
- 3,7,8,5,11,10,9,2



Sortowanie topologicznie - przykład

Wierzchołki w każdym grafie acyklicznym skierowanym można posortować topologicznie na jeden lub więcej sposobów:

- 7,5,3,11,8,2,9,10
- 7,5,11,2,3,10,8,9
- 3,7,8,5,11,10,9,2
- 5,7,11,2,3,8,9,10



Algorytm DFS - graf G=(V,E) reprezentowany przez listy sąsiedztwa

$DFS(G = (V, E))$**:**

for each vertex $u \in V$ **do**
    $color[u] = WHITE$
end for
$time = 0$
for each vertex $u \in V$ **do**
    if $color[u] == WHITE$ **then**
        $VISIT(G, u)$
    end if
end for

$VISIT(G, u)$

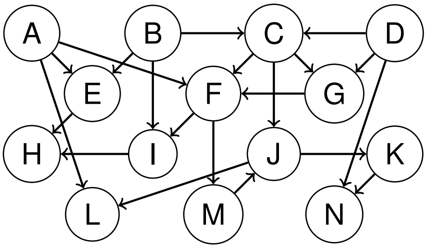
$time = time + 1$
$d[u] = time$
$color[u] = GRAY$
for each $v \in Adj[u]$ **do**
    if $color[v] == WHITE$ **then**
        $VISIT(G, v)$
    end if
end for
$color[u] = RED$
$time = time + 1$
$f[u] = time$

Algorytm sortowania topologicznego bazujący na DFS

TOPOLOGICAL-SORT(G)

- Wykonaj algorytm $DFS(G)$ na wejściowym DAG-u $G = (V, E)$, (reprezentowanym przez listy sąsiedztwa) w celu obliczenia czasów przetworzenia $f[v]$ dla wszystkich wierzchołków v.
- Wypisz wierzchołki w porządku malejącym ze względu na ich "czas przetworzenia", umieszczony w tablicy f.
- Złożoność: $\Theta(|V| + |E|)$ – ponieważ DFS można wykonać w czasie $\Theta(|V| + |E|)$.

Sortowanie topologiczne - przykład



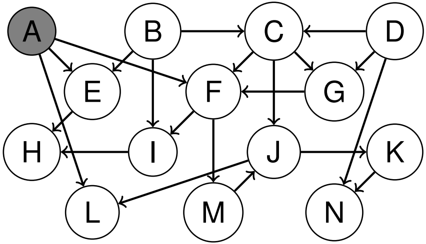
`time = 0`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
W	W	W	W	W	W	W	W	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$d[v]$	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sortowanie topologiczne - przykład



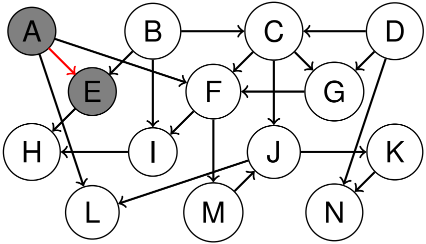
time = 1

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	W	W	W	W	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$d[v]$	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Sortowanie topologiczne - przykład



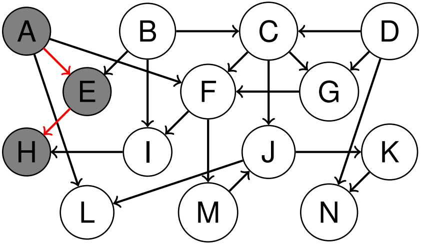
`time = 2`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	G	W	W	W	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	0	0	0	0	0	0	0	0	0

Sortowanie topologiczne - przykład



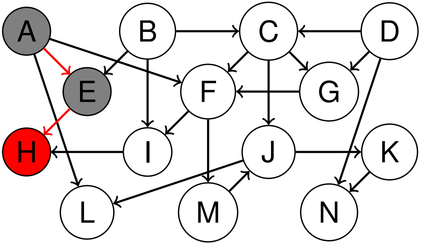
`time = 3`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	G	W	W	G	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	0	0	3	0	0	0	0	0	0

Sortowanie topologiczne - przykład



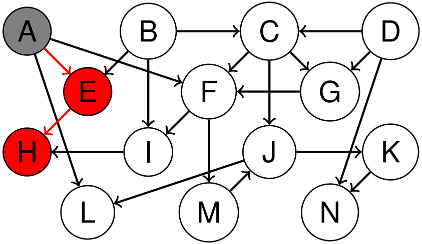
time = 4

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	G	W	W	R	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	0	0	0	4	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	0	0	3	0	0	0	0	0	0

Sortowanie topologiczne - przykład



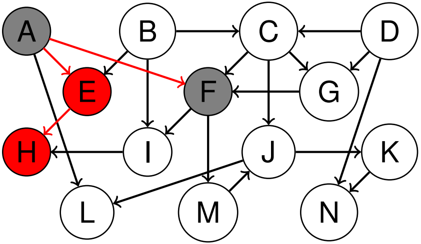
`time = 5`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	W	W	R	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	0	0	3	0	0	0	0	0	0

Sortowanie topologiczne - przykład



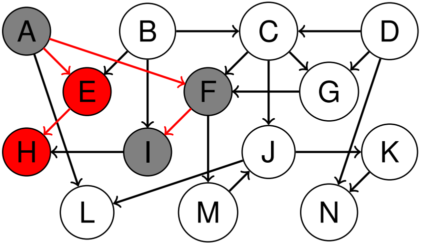
time = 6

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	W	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	0	0	0	0	0	0

Sortowanie topologiczne - przykład



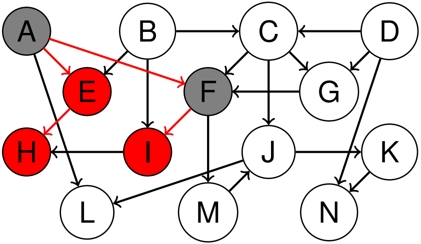
time = 7

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	G	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	0	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	0	0	0	0	0

Sortowanie topologiczne - przykład



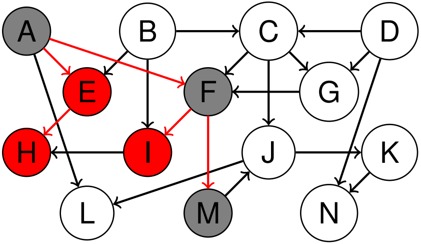
`time = 8`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	W	W	W	W	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	0	0	0	0	0

Sortowanie topologiczne - przykład



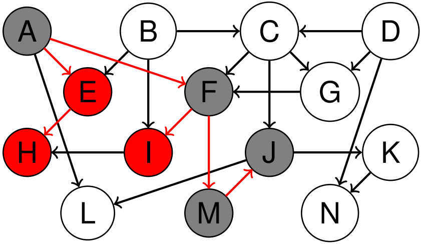
`time = 9`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	W	W	W	G	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	0	0	0	9	0

Sortowanie topologiczne - przykład



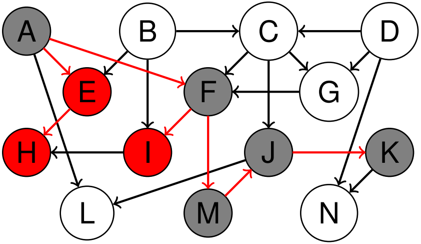
time = 10

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	W	W	G	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	10	0	0	9	0

Sortowanie topologiczne - przykład



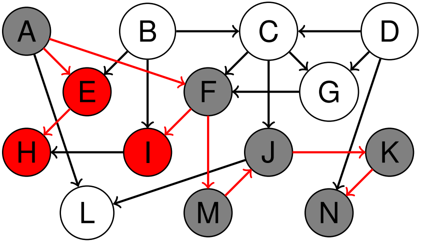
`time = 11`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	G	W	G	W

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	0	9	0

Sortowanie topologiczne - przykład



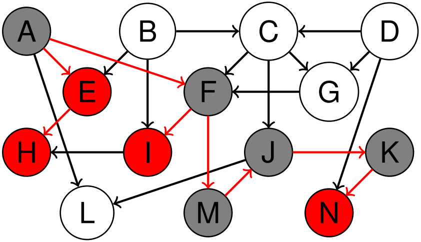
`time = 12`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	G	W	G	G

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	0
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	0	9	12

Sortowanie topologiczne - przykład



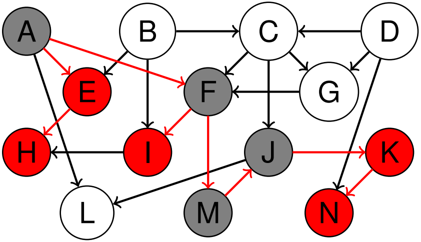
time = 13

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	G	W	G	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	0	0	0	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	0	9	12

Sortowanie topologiczne - przykład



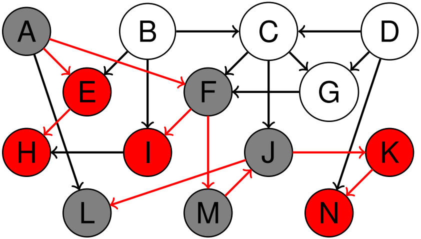
time = 14

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	R	W	G	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	14	0	0	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	0	9	12

Sortowanie topologiczne - przykład



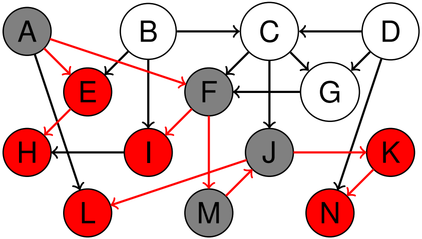
time = 15

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	R	G	G	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	14	0	0	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



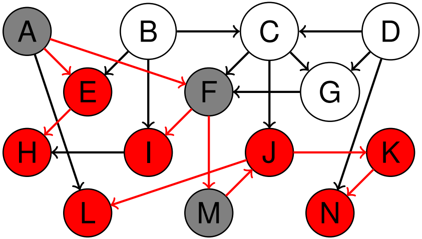
`time = 16`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	G	R	R	G	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	0	14	16	0	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



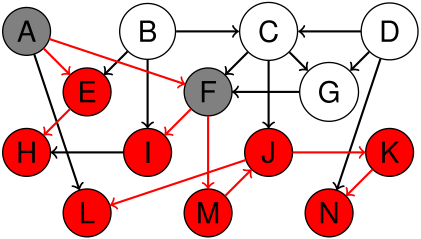
`time = 17`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	R	R	R	G	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	17	14	16	0	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



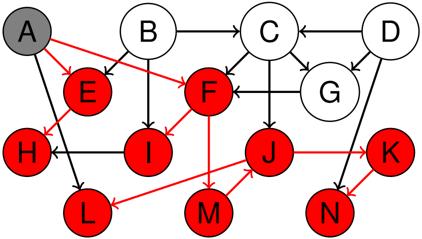
time = 18

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	G	W	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	0	0	4	8	17	14	16	18	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



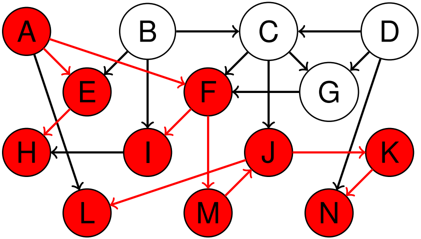
time = 19

A	B	C	D	E	F	G	H	I	J	K	L	M	N
G	W	W	W	R	R	W	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	0	0	0	0	5	19	0	4	8	17	14	16	18	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



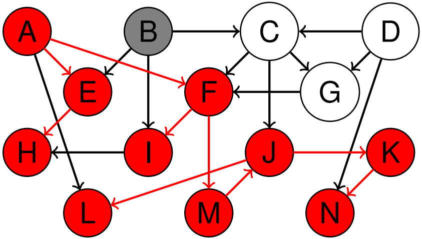
`time = 20`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	W	W	W	R	R	W	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	0	0	5	19	0	4	8	17	14	16	18	13
$d[v]$	1	0	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



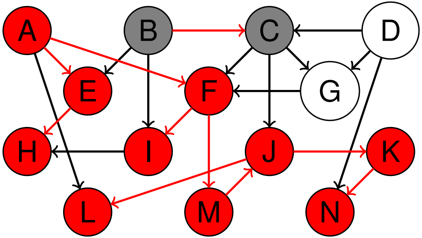
`time = 21`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	G	W	W	R	R	W	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	0	0	5	19	0	4	8	17	14	16	18	13
$d[v]$	1	21	0	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



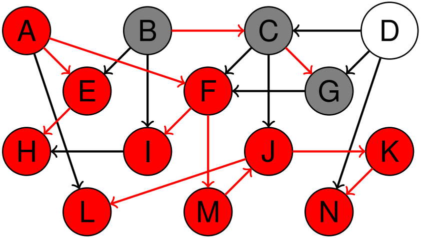
`time = 22`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	G	G	W	R	R	W	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	0	0	5	19	0	4	8	17	14	16	18	13
$d[v]$	1	21	22	0	2	6	0	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



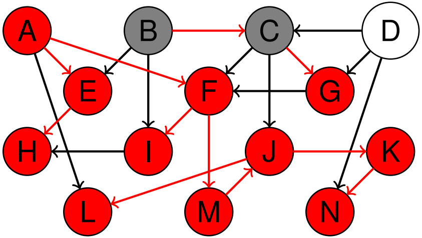
`time = 23`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	G	G	W	R	R	G	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	0	0	5	19	0	4	8	17	14	16	18	13
$d[v]$	1	21	22	0	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



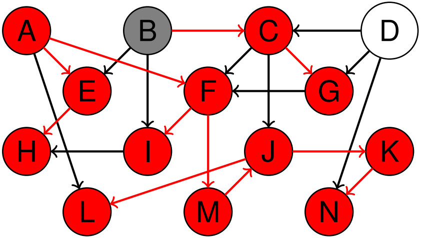
`time = 24`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	G	G	W	R	R	R	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	0	0	5	19	24	4	8	17	14	16	18	13
$d[v]$	1	21	22	0	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



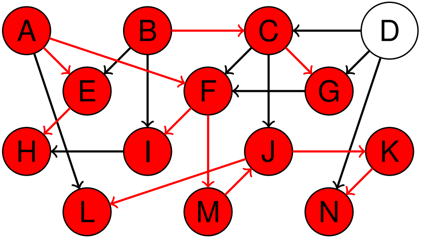
`time = 25`

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	G	R	W	R	R	R	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	0	25	0	5	19	24	4	8	17	14	16	18	13
$d[v]$	1	21	22	0	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



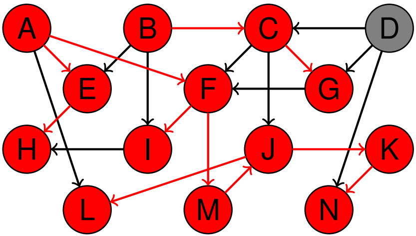
time = 26

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	R	R	W	R	R	R	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	26	25	0	5	19	24	4	8	17	14	16	18	13
$d[v]$	1	21	22	0	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



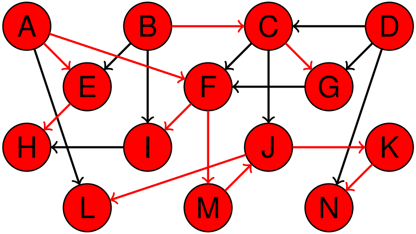
time = 27

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	R	R	G	R	R	R	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	26	25	0	5	19	24	4	8	17	14	16	18	13
$d[v]$	1	21	22	27	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład



time = 28

A	B	C	D	E	F	G	H	I	J	K	L	M	N
R	R	R	R	R	R	R	R	R	R	R	R	R	R

Tabela: Parametr `color`. W - oznacza WHITE, G - GRAY, R - RED

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$f[v]$	20	26	25	28	5	19	24	4	8	17	14	16	18	13
$d[v]$	1	21	22	27	2	6	23	3	7	10	11	15	9	12

Sortowanie topologiczne - przykład

Wynikowe sortowanie topologiczne:

`D B C G A F M J L K N I E H`

Zauważ, że wierzchołki do listy wchodzą w porządku malejącym ze względu na „końcowy czas przetwarzania” ich listy sąsiadów - tj. czas umieszczony w etykiecie $f[u]$:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
20	26	25	28	5	19	24	4	8	17	14	16	18	13

Sortowanie topologiczne. Reprezentacja macierzowa

Badanie nieodwiedzonego wierzchołka

```
// zwraca nieodwiedzony wierzchołek przyległy do a
// zwraca -1, jeżeli takiego wierzchołka nie ma
int getUnVisitedVertex(Vertex a, BoolVector visited)
{
 for (b = 0; b < n; b=b+1) do
  if (edge[a][b] == true // jest krawędź
      and  visited[b]==false) // b nie były odwiedzony
      then return b;
  endif
 endfor
 return -1;
}
```

Sortowanie topologiczne. Reprezentacja macierzowa

```
topological_DFS(DAG G) {
  BoolVector visited = [n];
  //końcowy czas przetworzenia wierzchołków
  int fin[n];
  time = 0;
  for (k = 0; k < n; k=k+1) do
    visited[k] = false;
    fin[k] = 0;
  endfor
  for (k = 0; k < n; k=k+1) do
    if (visited[k] == false)
       G.topological_visit(k,time,visited,fin);
    endif
  endfor
  for (k = 0; k < n; k=k+1) do
     idx = max(fin,n); // wyszukuje maksymalny w tablicy
     displayVertex(idx);
     fin[idx] = -1;
  endfor
}
```

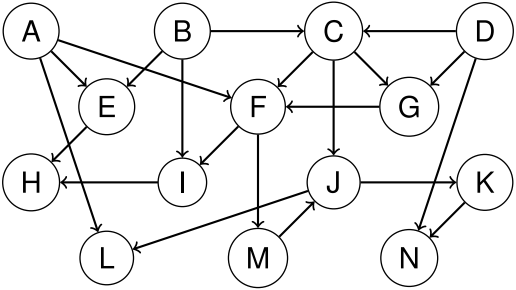
Sortowanie topologiczne. Reprezentacja macierzowa

```
topological_visit(int a,int time,
                  BoolVector visited, int fin[])
{
  visited[a] = true;
  c = getUnVisitedVertex(a, visited);
  while (c != -1) do
     if (visited [c] == false)
        topological_visit(G,c,time,visited,fin);
     endif
     c = getUnVisitedVertex(a, visited);
  endwhile
  time = time+1;
  fin[a] = time;
}
```

Działanie algorytmu - Przykład

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	0	0	0	0	1	1	0	0	0	0	0	1	0	0
B	0	0	1	0	1	0	0	0	1	0	0	0	0	0
C	0	0	0	0	0	1	1	0	0	1	0	0	0	0
D	0	0	1	0	0	0	1	0	0	0	0	0	0	1
E	0	0	0	0	0	0	0	1	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	1	0	0	0	1	0
G	0	0	0	0	0	1	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	1	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	1	1	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	1
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	1	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Działanie algorytmu - Przykład I



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Nr. v	0	1	2	3	4	5	6	7	8	9	10	11	12	13
visited	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fin	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Działanie algorytmu - Przykład II

1. $time = 0;$
2. $visited[0] == 0$? true;
3. `topological_visit(0,time,visited,fin)`
4. $visited[0] = 1;$
5. $c = getUnVisitedVertex(0, visited) = 4;$// (0–>4)
6. $visited[4] == 0$? true;
7. `topological_visit(4,time,visited,fin);`
8. $visited[4] = 1;$
9. $c = getUnVisitedVertex(4, visited) = 7;$// (4–>7)
10. $visited[7] == 0$? true;
11. `topological_visit(7,time,visited,fin);`
12. $visited[7] = 1;$

Działanie algorytmu - Przykład III

13. $c = getUnVisitedVertex(7, visited) = -1;$
14. $time = time + 1 = 1;$ $fin[7] = 1;$
15. $c = getUnVisitedVertex(4, visited) = -1$
16. $time = time + 1 = 2;$ $fin[4] = 2;$
17. $c = getUnVisitedVertex(0, visited) = 5;$// (0–>5)
18. $visited[5] == 0$? true;
19. `topological_visit(5,time,visited,fin);`
20. $visited[5] = 1;$
21. $c = getUnVisitedVertex(5, visited) = 8;$// (5–>8)
22. $visited[8] == 0$? true;
23. `topological_visit(8,time,visited,fin);`
24. $visited[8] = 1;$

Działanie algorytmu - Przykład IV

25. $c = getUnVisitedVertex(8, visited) = -1;$
26. $time = time + 1 = 3;$ $fin[8] = 3;$
27. $c = getUnVisitedVertex(5, visited) = 12;$// (5–>12)
28. $visited[12] == 0$? true;
29. `topological_visit(12,time,visited,fin);`
30. $visited[12] = 1;$
31. $c = getUnVisitedVertex(12, visited) = 9;$ //(12–>9)
32. $visited[9] == 0$? true;
33. `topological_visit(9,time,visited,fin);`
34. $visited[9] = 1;$
35. $c = getUnVisitedVertex(9, visited) = 10;$ //(9–>10)
36. $visited[10] == 0$? true;

Działanie algorytmu - Przykład V

37. `topological_visit(10,time,visited,fin);`
38. $visited[10] = 1;$
39. $c = getUnVisitedVertex(10, visited) = 13;$ //(10–>13)
40. $visited[13] == 0$? true;
41. `topological_visit(13,time,visited,fin);`
42. $visited[13] = 1;$
43. $c = getUnVisitedVertex(13, visited) = -1;$
44. $time = time + 1 = 4; fin[13] = 4;$
45. $c = getUnVisitedVertex(10, visited) = -1;$
46. $time = time + 1 = 5; fin[10] = 5;$
47. $c = getUnVisitedVertex(9, visited) = 11;$ //(9–>11)
48. $visited[11] == 0$? true;

Działanie algorytmu - Przykład VI

49. `topological_visit(11,time,visited,fin);`
50. $visited[11] = 1;$
51. $c = getUnVisitedVertex(11, visited) = -1;$
52. $time = time + 1 = 6; fin[11] = 6;$
53. $c = getUnVisitedVertex(9, visited) = -1;$
54. $time = time + 1 = 7; fin[9] = 7;$
55. $c = getUnVisitedVertex(12, visited) = -1;$
56. $time = time + 1 = 8; fin[12] = 8;$
57. $c = getUnVisitedVertex(5, visited) = -1;$
58. $time = time + 1 = 9; fin[5] = 9;$
59. $c = getUnVisitedVertex(0, visited) = -1;$
60. $time = time + 1 = 10; fin[0] = 10;$

Działanie algorytmu - Przykład VII

61. $visited[1] == 0$? true;
62. `topological_visit(1,time,visited,fin)`
63. $visited[1] = 1;$
64. $c = getUnVisitedVertex(1, visited) = 2;$// (1–>2)
65. $visited[2] == 0$? true;
66. `topological_visit(2,time,visited,fin)`
67. $visited[2] = 1;$
68. $c = getUnVisitedVertex(2, visited) = 6;$// (2–>6)
69. $visited[6] == 0$? true;
70. `topological_visit(6,time,visited,fin)`
71. $visited[6] = 1;$
72. $c = getUnVisitedVertex(6, visited) = -1;$

Działanie algorytmu - Przykład VIII

73. $time = time + 1 = 11; fin[6] = 11;$
74. $c = getUnVisitedVertex(2, visited) = -1;$
75. $time = time + 1 = 12; fin[2] = 12;$
76. $c = getUnVisitedVertex(1, visited) = -1;$
77. $time = time + 1 = 13; fin[2] = 11;$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Nr. v	0	1	2	3	4	5	6	7	8	9	10	11	12	13
fin	10	13	12	14	2	9	11	1	3	7	5	6	8	4

Wynikowe sortowanie topologiczne:

D B C G A F M J L K N I E H

Sortowanie topologiczne - wersja nierekurencyjna

Metoda usuwania wierzchołków o stopniu wejściowym równym zero

- **Wykorzystywana własność:** jeśli graf jest acyklicznym grafem skierowanym, to posiada przynajmniej jeden wierzchołek o stopniu wejściowym równym zero.
- **Idea:** Dopóki graf posiada wierzchołki o stopniu wejściowym zero, znajdujemy taki wierzchołek, usuwamy go z grafu wraz ze wszystkimi wychodzącymi z niego krawędziami i umieszczamy go na liście wierzchołków posortowanych topologicznie.
- Jeśli w grafie pozostaną jakieś wierzchołki, to graf posiada cykle i sortowania topologicznego nie można wykonać.
- Złożoność: $O(|V| + |E|)$

Sortowanie topologiczne - Metoda usuwania wierzchołków I

- $G = (V, E)$ - acykliczny graf skierowany, tzw DAG
- G - reprezentowany jest przez listy sąsiedztwa, tj. $G.Adj(v)$ oznacza listę sąsiadów w wierzchołka v w grafie G.
- Rozmiar V jest n.
- Wierzchołki ponumerowane są od 0 do $n - 1$.
- $in\_degree(v)$ - stopień wejściowy wierzchołka $v \in V$
- *order* - kolejka, która będzie zawierać wynikowy porządek topologiczny, jeśli istnieje.

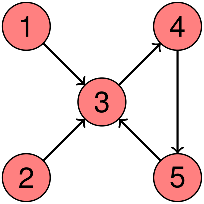
Sortowanie topologiczne - Metoda usuwania wierzchołków II

```
TopologicalSort(DAG G) {
 Queue Q, Order;
 count = 0;
 int indegree [n];
 for(v = 0; v < n; v = v+1)
     indegree[v] = in_degree(v);
 endfor
 //wstaw do kolejki Q wszystkie wierzchołki
 //ze stopniem wejsciowym = 0
 for (v = 0; v < n; v++)
    if (indegree[v] == 0) then Q.EnQueue(v);
 endfor
 while (not Q.Empty()) do
```

Sortowanie topologiczne - Metoda usuwania wierzchołków III

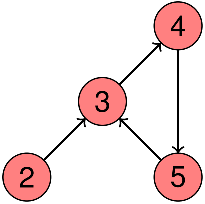
```
    v = Q.DeQueue();
    Order.EnQueue(v);
    count = count + 1;
    forall( w in G.Adj(v))
      indegree[w] = indegree[w]-1;
      if (indegree[w] == 0) then Q.EnQueue(w);
    endforall
 endwhile
 // Istnieje cykl w grafie.
  if (count != n) then  Order = null;
  else return Order;
}
```

Nierekurencyjne sortowanie topologiczne - przykład



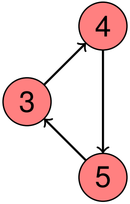
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 3$, $indegree[4] = 1$, $indegree[5] = 1$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta

Nierekurencyjne sortowanie topologiczne - przykład



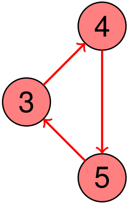
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 3$, $indegree[4] = 1$, $indegree[5] = 1$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
   1. $v = 1$; $Q = \{2\}$;
   2. $order = \{1\}$;
   3. $count = 1$;
   4. Krawędź $(1, 3)$; $indegree[3] = 2$;

Nierekurencyjne sortowanie topologiczne - przykład



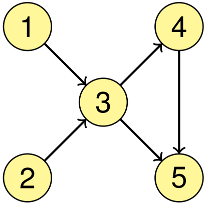
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 3$, $indegree[4] = 1$, $indegree[5] = 1$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
    1. $v = 1$; $Q = \{2\}$;
    2. $order = \{1\}$;
    3. $count = 1$;
    4. Krawędź $(1, 3)$; $indegree[3] = 2$;
    5. $v = 2$; $Q = \{\}$;
    6. $order = \{1, 2\}$;
    7. $count = 2$;
    8. Krawędź $(2, 3)$; $indegree[3] = 1$;

Nierekurencyjne sortowanie topologiczne - przykład



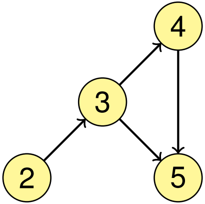
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 3$, $indegree[4] = 1$, $indegree[5] = 1$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
    1. $v = 1$; $Q = \{2\}$;
    2. $order = \{1\}$;
    3. $count = 1$;
    4. Krawędź $(1, 3)$; $indegree[3] = 2$;
    5. $v = 2$; $Q = \{\}$;
    6. $order = \{1, 2\}$;
    7. $count = 2$;
    8. Krawędź $(2, 3)$; $indegree[3] = 1$;
4. $Q = \{\}$ jest pusta. Wychodzimy w WHILE
5. $count = 2$, $n = 5 \Rightarrow$ w grafie jest cykl.

Nierekurencyjne sortowanie topologiczne - przykład



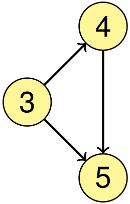
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta

Nierekurencyjne sortowanie topologiczne - przykład



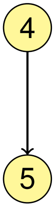
1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
   1. $v = 1$; $Q = \{2\}$; $order = \{1\}$; $count = 1$;
   2. Krawędź $(1, 3)$; $indegree[3] = 1$;

Nierekurencyjne sortowanie topologiczne - przykład



1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
   1. $v = 1$; $Q = \{2\}$; $order = \{1\}$; $count = 1$;
   2. Krawędź $(1, 3)$; $indegree[3] = 1$;
   3. $v = 2$; $Q = \{\}$; $order = \{1, 2\}$; $count = 2$;
   4. Krawędź $(2, 3)$; $indegree[3] = 0$; $Q = \{3\}$;

Nierekurencyjne sortowanie topologiczne - przykład



1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
   1. $v = 1$; $Q = \{2\}$; $order = \{1\}$; $count = 1$;
   2. Krawędź $(1, 3)$; $indegree[3] = 1$;
   3. $v = 2$; $Q = \{\}$; $order = \{1, 2\}$; $count = 2$;
   4. Krawędź $(2, 3)$; $indegree[3] = 0$; $Q = \{3\}$;
   5. $v = 3$; $Q = \{\}$; $order = \{1, 2, 3\}$; $count = 3$;
   6. Krawędź $(3, 4)$; $indegree[4] = 0$; $Q = \{4\}$;
   7. Krawędź $(3, 5)$; $indegree[5] = 1$; $Q = \{4\}$;

Nierekurencyjne sortowanie topologiczne - przykład

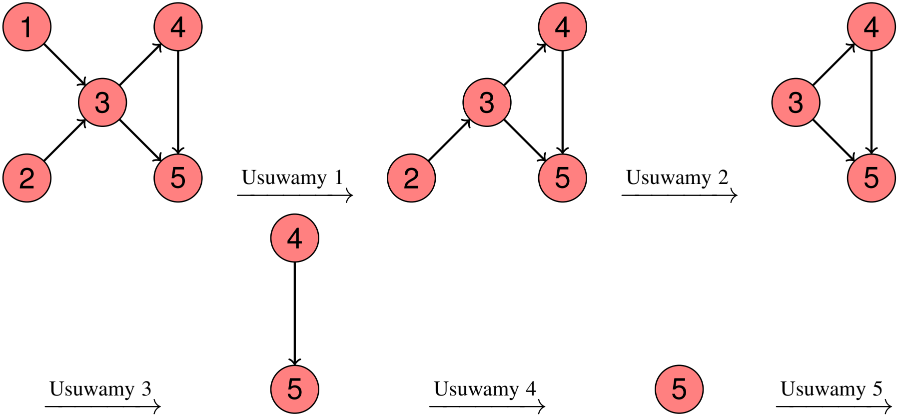
5

1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
   1. $v = 1$; $Q = \{2\}$; $order = \{1\}$; $count = 1$;
   2. Krawędź $(1, 3)$; $indegree[3] = 1$;
   3. $v = 2$; $Q = \{\}$; $order = \{1, 2\}$; $count = 2$;
   4. Krawędź $(2, 3)$; $indegree[3] = 0$; $Q = \{3\}$;
   5. $v = 3$; $Q = \{\}$; $order = \{1, 2, 3\}$; $count = 3$;
   6. Krawędź $(3, 4)$; $indegree[4] = 0$; $Q = \{4\}$;
   7. Krawędź $(3, 5)$; $indegree[5] = 1$; $Q = \{4\}$;
   8. $v = 4$; $Q = \{\}$; $order = \{1, 2, 3, 4\}$; $count = 4$;
   9. Krawędź $(4, 5)$; $indegree[5] = 0$; $Q = \{5\}$;

Nierekurencyjne sortowanie topologiczne - przykład

1. *order* – pusta; count = 0
2. $indegree[1] = 0$, $indegree[2] = 0$, $indegree[3] = 2$, $indegree[4] = 1$, $indegree[5] = 2$.
3. $Q = \{1, 2\}$ // Dopóki kolejka nie jest pusta
    1. $v = 1$; $Q = \{2\}$; $order = \{1\}$; $count = 1$;
    2. Krawędź $(1, 3)$; $indegree[3] = 1$;
    3. $v = 2$; $Q = \{\}$; $order = \{1, 2\}$; $count = 2$;
    4. Krawędź $(2, 3)$; $indegree[3] = 0$; $Q = \{3\}$;
    5. $v = 3$; $Q = \{\}$; $order = \{1, 2, 3\}$; $count = 3$;
    6. Krawędź $(3, 4)$; $indegree[4] = 0$; $Q = \{4\}$;
    7. Krawędź $(3, 5)$; $indegree[5] = 1$; $Q = \{4\}$;
    8. $v = 4$; $Q = \{\}$; $order = \{1, 2, 3, 4\}$; $count = 4$;
    9. Krawędź $(4, 5)$; $indegree[5] = 0$; $Q = \{5\}$;
    10. $v = 5$; $Q = \{\}$; $order = \{1, 2, 3, 4, 5\}$; $count = 5$;
4. $count = n$ // TRUE
5. Porządek topologiczny: 1,2,3,4,5.

Nierekurencyjne sortowanie topologiczne - przykład



Sortowanie topologiczne: 1, 2, 3, 4, 5