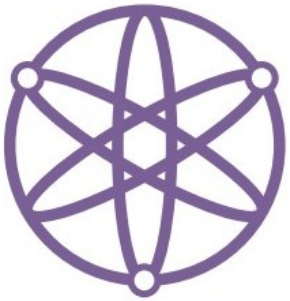


WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ

Najkrótsze drogi

M. Rzeźnikiewicz, P. Ryba

11 kwietnia 2025

Spis treści

Wprowadzenie

Optymalizacja dyskretna

Optymalizacja dyskretna zajmuje się rozwiązywaniem problemów optymalizacyjnych, które mają charakter dyskretny, czyli wartości zmiennych decyzyjnych są ograniczone do skończonego zbioru możliwych wartości.

Problem najkrótszych dróg

Problem najkrótszych dróg polega na znalezieniu najkrótszej ścieżki między dwoma wierzchołkami w grafie ważonym, gdzie wagi krawędzi reprezentują odległości lub koszty podróży między wierzchołkami.

Algorytm Dijkstry

Opis algorytmu

Algorytm Dijkstry jest podstawowym problemem w teorii grafów i sieci, służącym do znajdowania najkrótszych połączeń pomiędzy dwoma wybranymi wierzchołkami w grafie ważonym, gdzie wagi krawędzi są nieujemne.

Działanie algorytmu - opis słowny

1. Inicjalizacja odległości od wierzchołka źródłowego do pozostałych jako ∞, a odległość od samego siebie jako 0.
2. Iteracyjne odwiedzanie sąsiednich wierzchołków, aktualizując ich odległości oraz informacje o poprzednikach w ścieżce.
3. Powtarzanie kroków 1 i 2, aż wszystkie wierzchołki zostaną odwiedzone, czyli uniknięcie sytuacji, w której któremuś z wierzchołków przypisane jest nadal ∞.

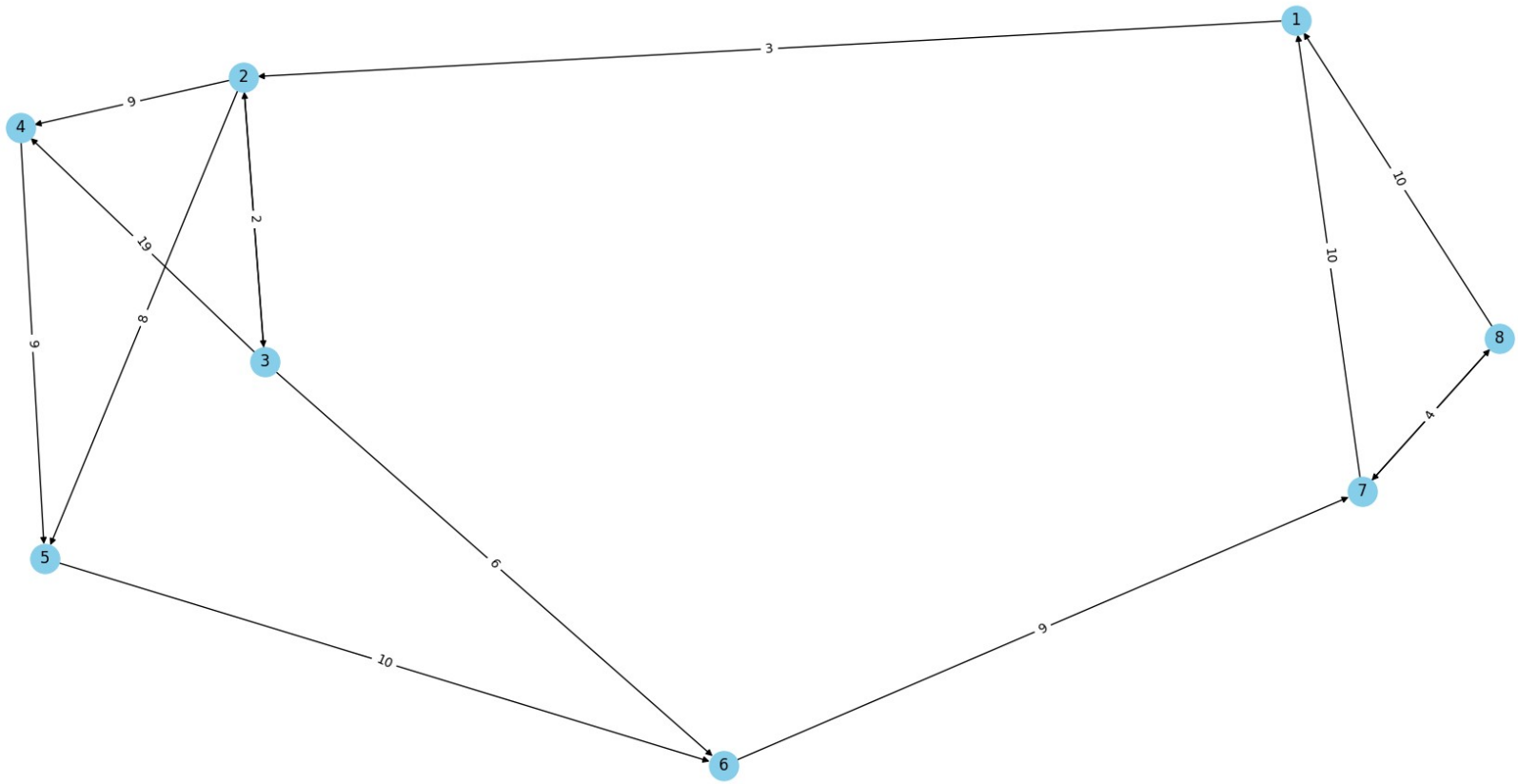
Algorytm Dijkstry - model matematyczny

Rozważmy sieć $D = (V, E, c)$ z nieujemnymi wagami na krawędziach. Algorytm Dijkstry służy do ustalenia odległości między wybranym wierzchołkiem źródłowym s i każdym innym wierzchołkiem w sieci. W zdefiniowanym wcześniej modelu - V oznacza zbiór wierzchołków, E oznacza zbiór krawędzi, a c reprezentuje wagi przypisane do poszczególnych krawędzi. W kontekście dalszych rozważań ważne jest, aby wagi na krawędziach były nieujemne.

Przyjmijmy, że mamy digraf $G = (V, E)$, gdzie zbiór wierzchołków V został podzielony na dwa rozłączne zbiory S i Q. Dla każdego wierzchołka $v \in V$ przypiszmy parametr d_v. Początkowo wszystkie wierzchołki znajdują się w zbiorze Q. W trakcie kolejnych kroków algorytmu, wierzchołki osiągalne z ustalonego wierzchołka s są przenoszone ze zbioru Q do S. Gdy wierzchołek v należy do zbioru Q, parametr d_v stanowi górne ograniczenie odległości pomiędzy s a v. Przenosząc wierzchołek v do zbioru S, aktualizujemy d_v jako najkrótszą drogą z s do v.

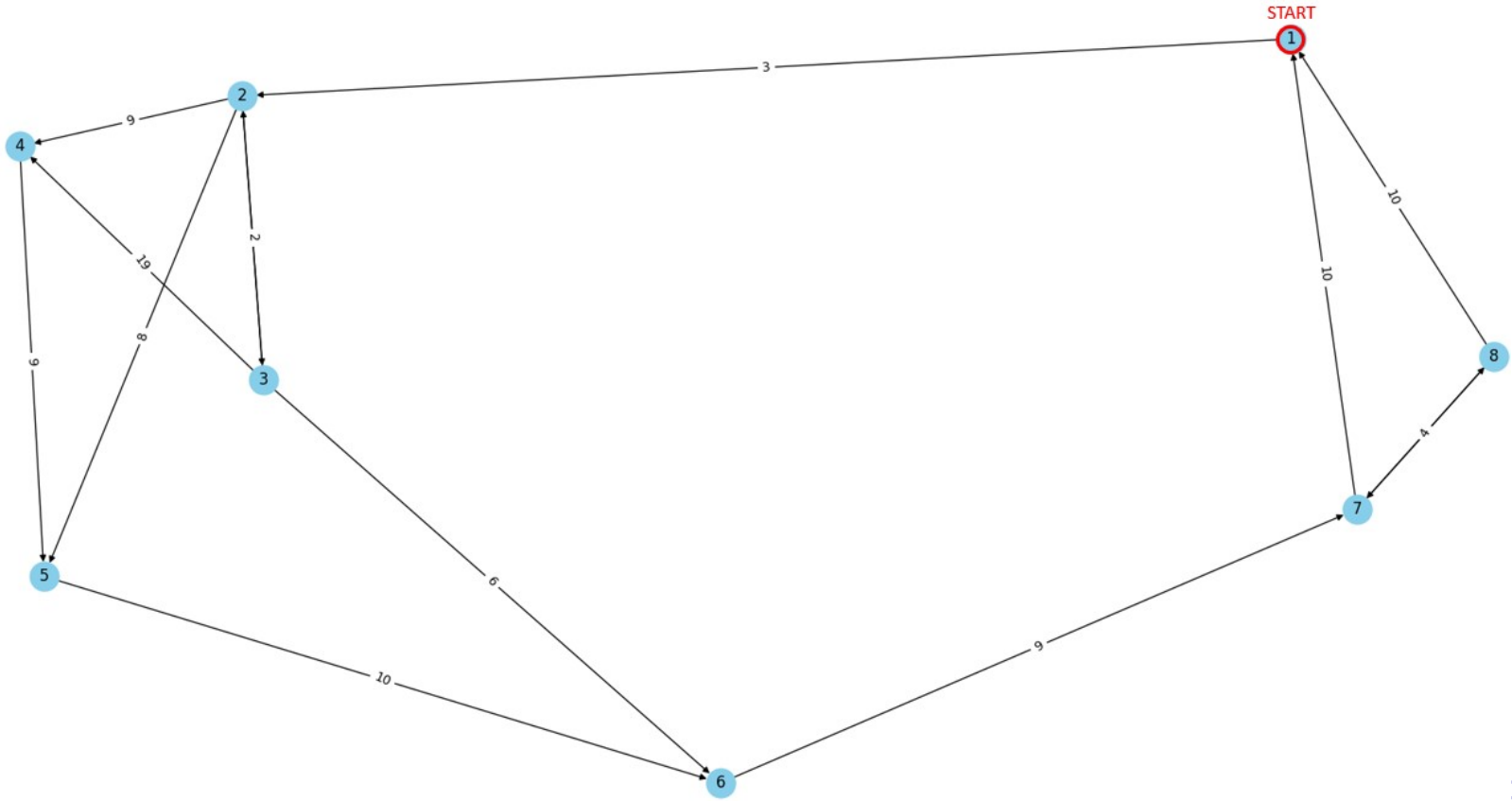
Przykład użycia algorytmu Dijkstry

Wyznaczmy odległości od wierzchołka 1 do wszystkich pozostałych wierzchołków w zadanym digrafie G.



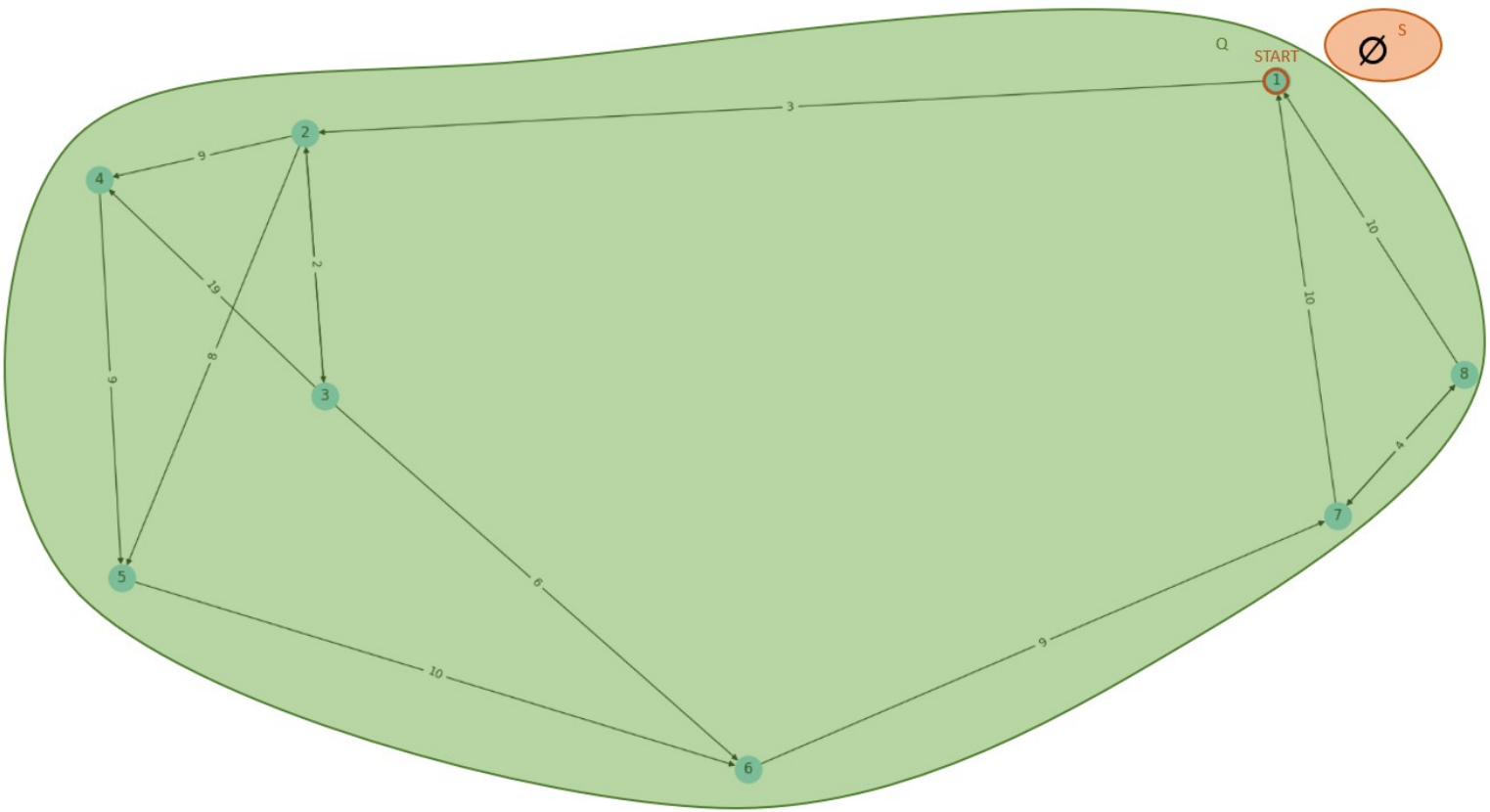
Przykład użycia algorytmu Dijkstry - krok 1

- Mamy ważony graf skierowany z wierzchołkiem startowym $v = 1$. Będziemy wyznaczać najniższe koszty dojścia od wyróżnionego wierzchołka źródłowego do wszystkich pozostałych wierzchołków w grafie oraz najkrótsze ścieżki pomiędzy wyróżnionym wierzchołkiem, a wszystkimi pozostałymi.



Przykład użycia algorytmu Dijkstry - krok 2

- Tworzymy dwa zbiory S i Q. Zbiór S jest początkowo pusty, a zbiór Q obejmuje wszystkie wierzchołki grafu. W zbiorze S znajdą się wierzchołki przetworzone przez algorytm Dijkstry, a w zbiorze Q będą wierzchołki wciąż czekające na przetworzenie.



Przykład użycia algorytmu Dijkstry - krok 2

- Tworzymy tabelę posiadającą trzy wiersze: v, d_v i p_v, złożoną z n kolumn, gdzie n to liczba wierzchołków w grafie (nie bierzemy pod uwagę kolumny z nazwami wierszy). Wiersz d_v przechowuje minimalne koszty dotarcia do każdego wierzchołka z wierzchołka startowego. Na początku, każdy element tego wiersza jest inicjalizowany wartością oznaczającą ∞, z wyjątkiem elementu d_1, który przyjmuje wartość 0.
- Wiersz p_v zawiera numery poprzedzających wierzchołków w ścieżce od wierzchołka v. Przechodząc wstecz przez te numery, możemy dotrzeć do wierzchołka startowego. Wartością początkową dla wszystkich elementów tablicy p_v jest liczba, która nie może reprezentować numeru wierzchołka, na przykład 0. Przyjeliśmy konwencję, że zaczynamy numerowanie wierzchołków od jedynki.

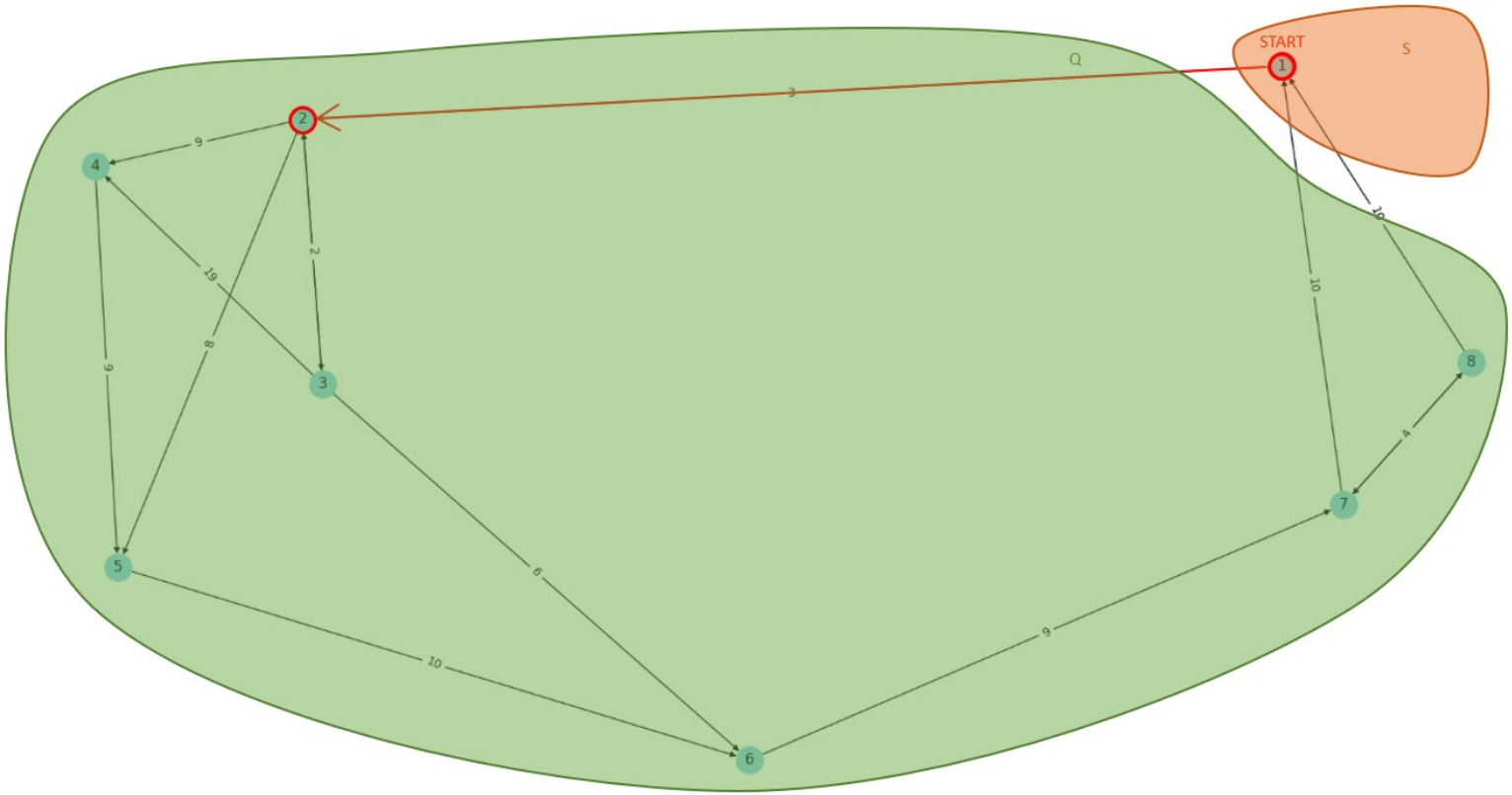
Przykład użycia algorytmu Dijkstry - krok 2

Tabela: Tabela dla kroku 2

v	1	2	3	4	5	6	7	8
d_v	0	∞	∞	∞	∞	∞	∞	∞
p_v	0	0	0	0	0	0	0	0

Przykład użycia algorytmu Dijkstry - krok 3

- W zbiorze Q poszukujemy wierzchołka v o najmniejszym koszcie dojścia d_v. Oczywiście, jest to wierzchołek startowy 1, dla którego $d_1 = 0$. Wierzchołek 1 przenosimy ze zbioru Q do S.
  Następnie przeglądamy wszystkich sąsiadów przeniesionego wierzchołka. Jest nim tylko wierzchołek 2.



Przykład użycia algorytmu Dijkstry - krok 3

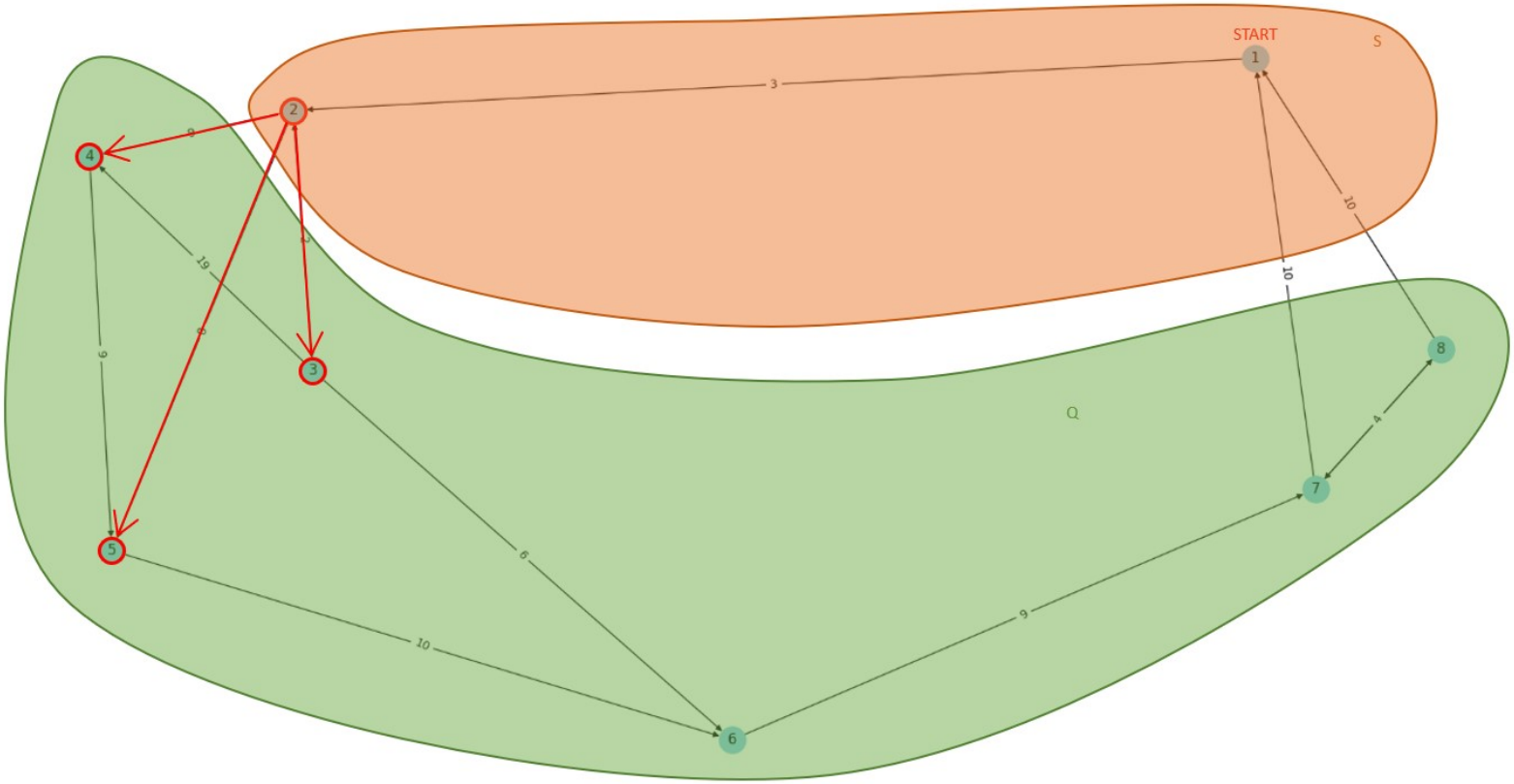
$d_2 > d_1 + 3 \Rightarrow d_2 := d_1 + 3 = 3$, więc $p_2 = 1$

Tabela: Tabela dla kroku 3

v	1	2	3	4	5	6	7	8
d_v	0	3	∞	∞	∞	∞	∞	∞
p_v	0	1	0	0	0	0	0	0

Przykład użycia algorytmu Dijkstry - krok 4

- Po raz kolejny w zbiorze Q poszukujemy wierzchołka v o najmniejszym koszcie dojścia d_v. Tym razem jest to wierzchołek 2, więc przenosimy go ze zbioru Q do S. W kolejnym etapie przeglądamy wszystkich sąsiadów przeniesionego wierzchołka, a dokładniej wierzchołki 3, 4 oraz 5.



Przykład użycia algorytmu Dijkstry - krok 4

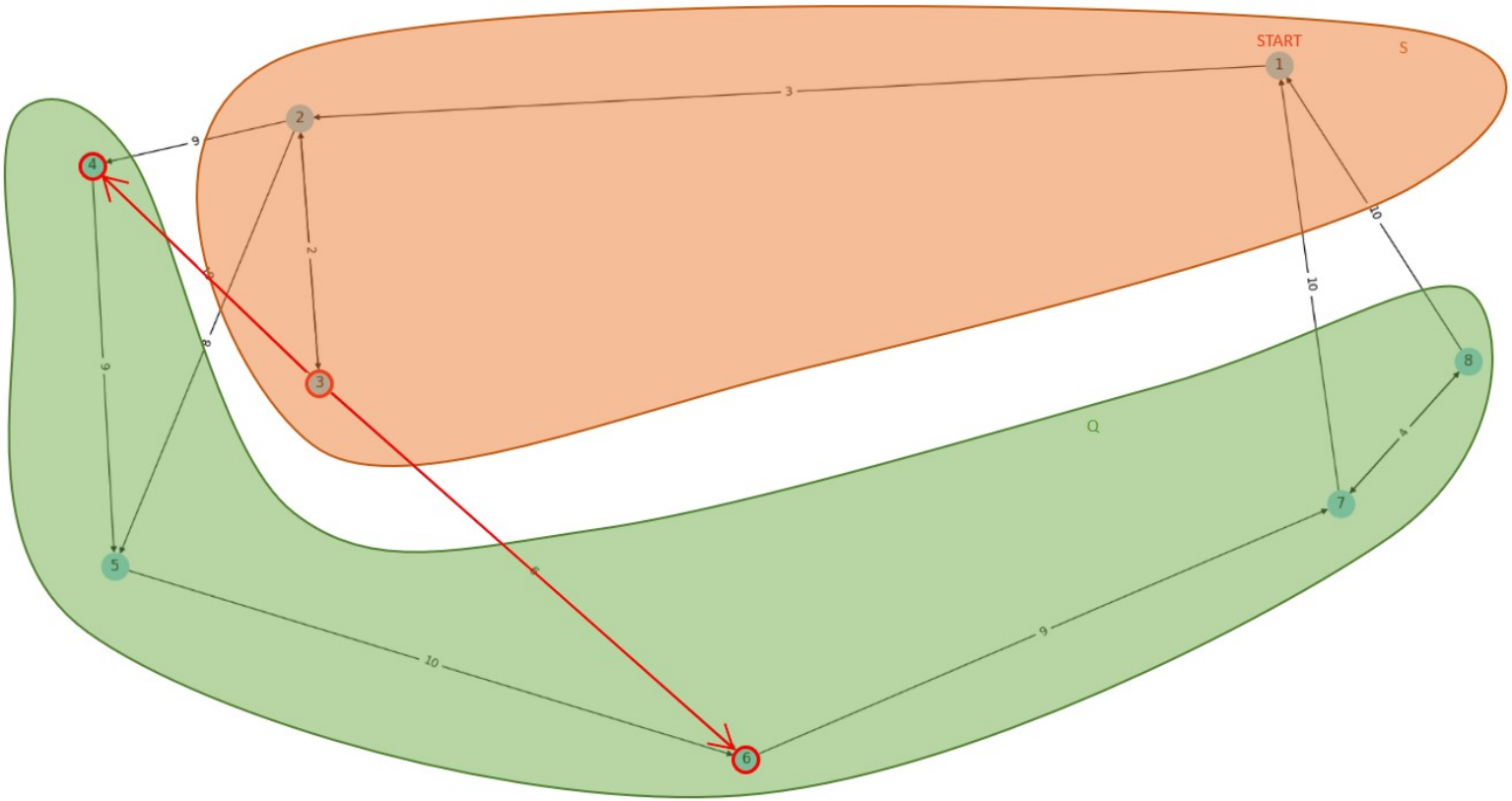
$d_3 > d_2 + 2 \Rightarrow d_2 := d_1 + 2 = 5$
$d_4 > d_2 + 9 \Rightarrow d_4 := d_2 + 9 = 12$
$d_5 > d_2 + 8 \Rightarrow d_5 := d_2 + 8 = 11$
$p_3 = p_4 = p_5 = 2$

Tabela: Tabela dla kroku 4

v	1	2	3	4	5	6	7	8
d_v	0	3	5	12	11	∞	∞	∞
p_v	0	1	2	2	2	0	0	0

Przykład użycia algorytmu Dijkstry - krok 5

- W dalszej kolejności rozważamy wierzchołek 3 i przenosimy go ze zbioru Q do S. Jego sąsiadami są wierzchołki 4 oraz 6.



Przykład użycia algorytmu Dijkstry - krok 5

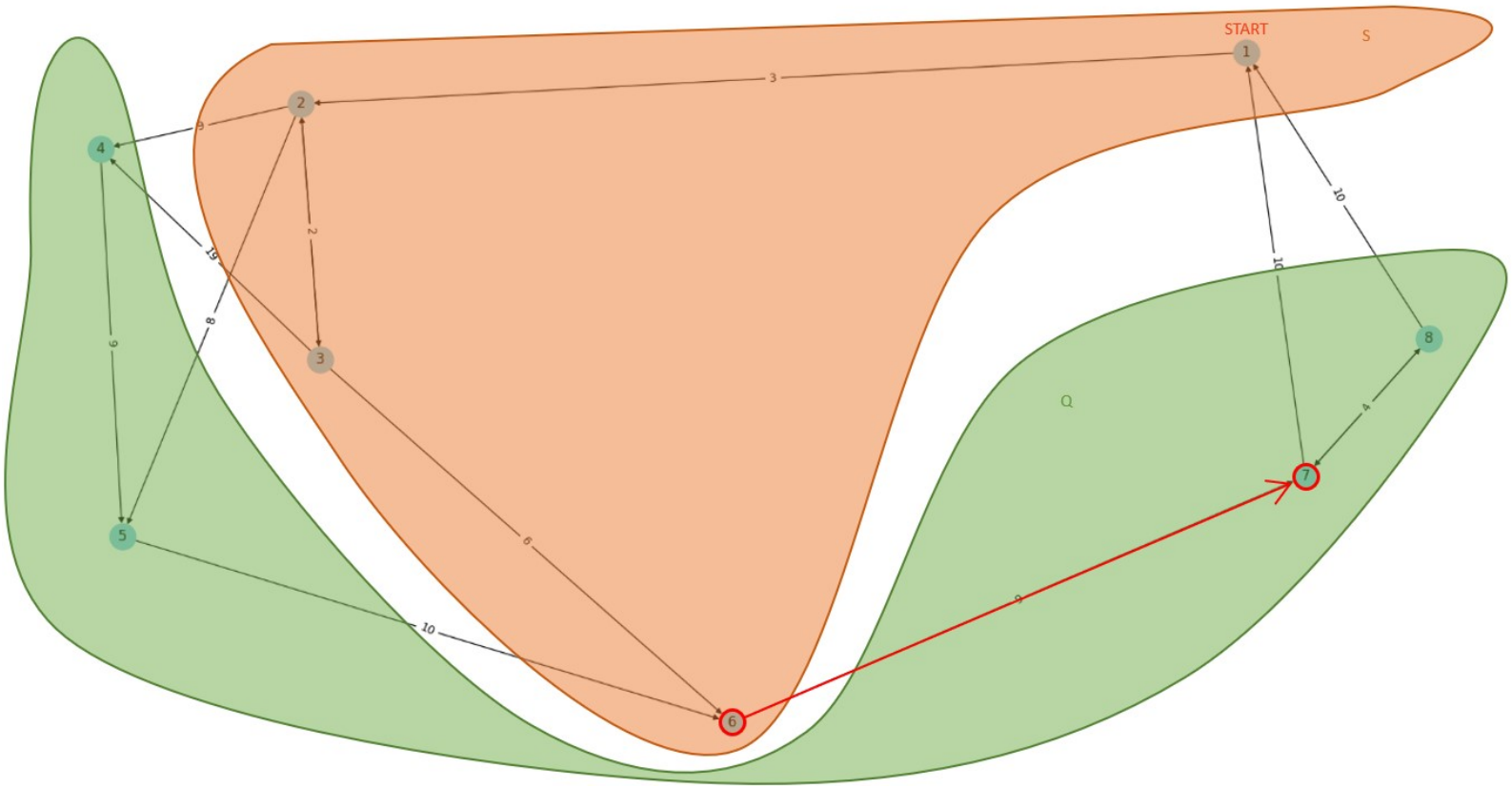
$d_6 > d_3 + 6 \Rightarrow d_6 := d_3 + 6 = 11$, więc $p_6 = 3$
$d_4 < d_3 + 19$

Tabela: Tabela dla kroku 5

v	1	2	3	4	5	6	7	8
d_v	0	3	5	12	11	11	∞	∞
p_v	0	1	2	2	2	3	0	0

Przykład użycia algorytmu Dijkstry - krok 6

- W tym kroku to wierzchołek 6 przenosimy ze zbioru Q do S. Posiada on tylko jednego sąsiada - wierzchołek 7.



Przykład użycia algorytmu Dijkstry - krok 6

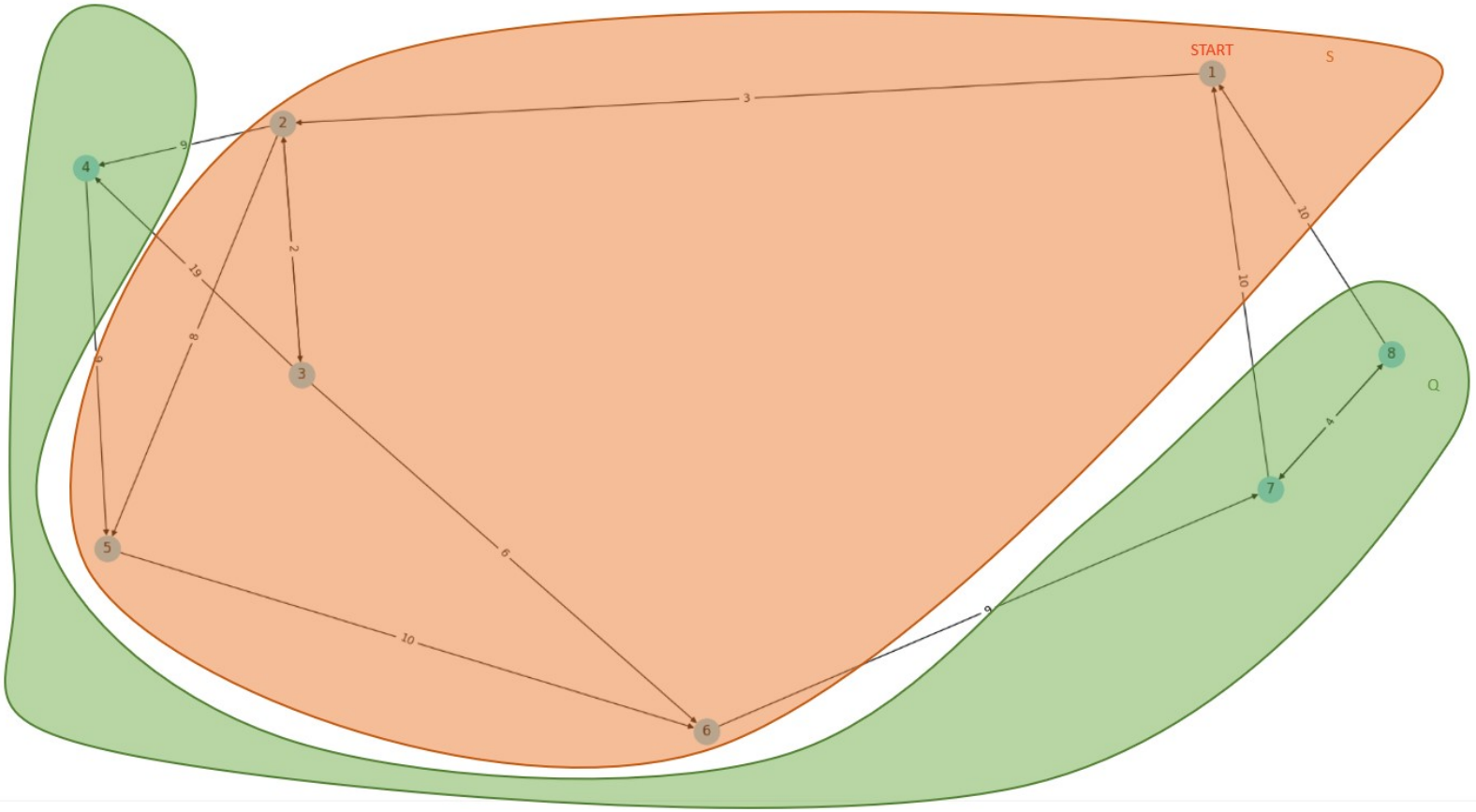
$d_7 > d_6 + 9 \Rightarrow d_7 := d_6 + 9 = 20$, więc $p_7 = 6$

Tabela: Tabela dla kroku 6

v	1	2	3	4	5	6	7	8
d_v	0	3	5	12	11	11	20	∞
p_v	0	1	2	2	2	3	6	0

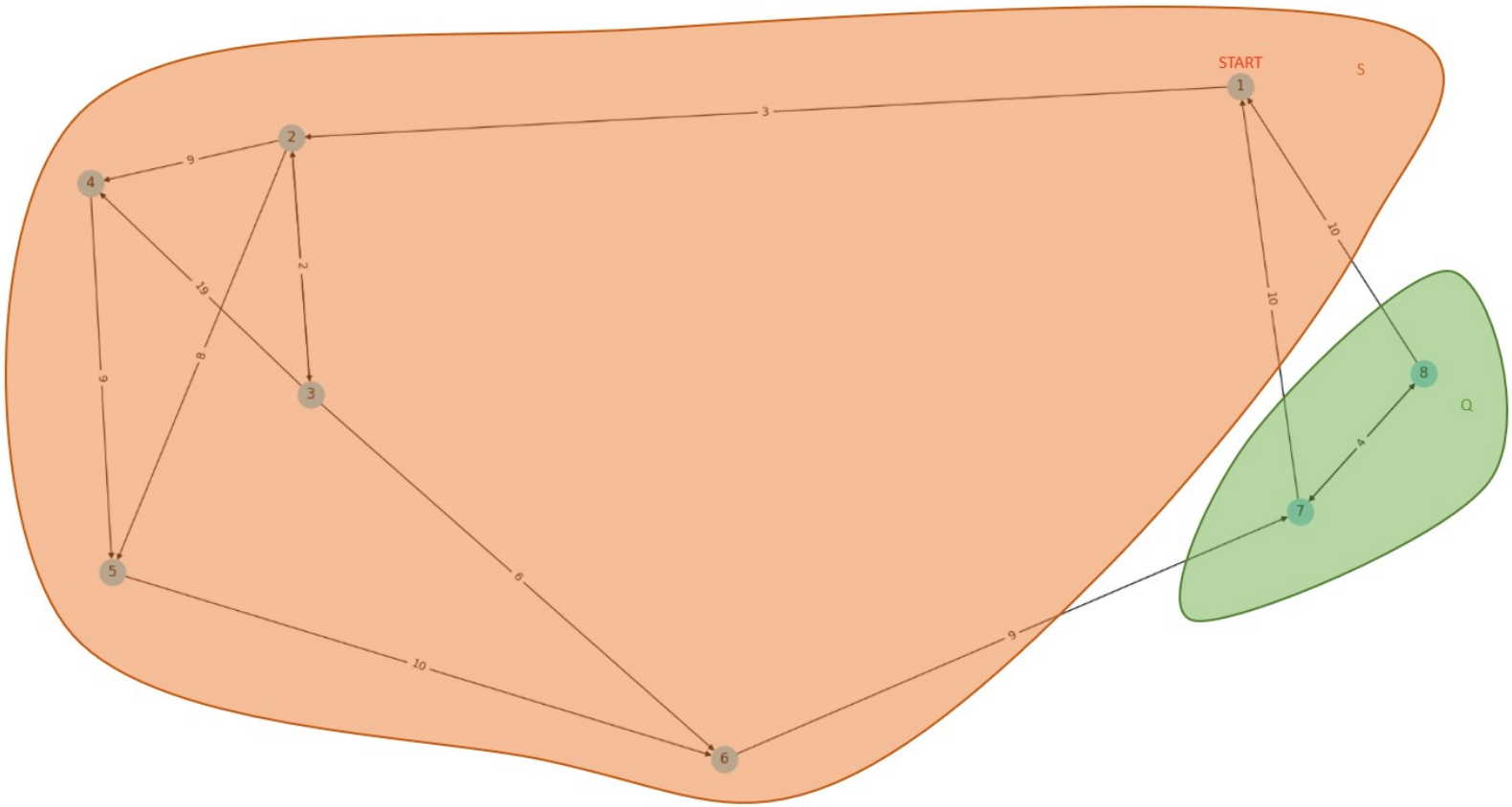
Przykład użycia algorytmu Dijkstry - krok 7

- Wracamy do wierzchołka 5 i przenosimy go ze zbioru Q do S. Niestety nie posiada on na ten moment sąsiadów w zbiorze Q, więc nasza tabela po tym kroku nie ulegnie żadnym zmianom.



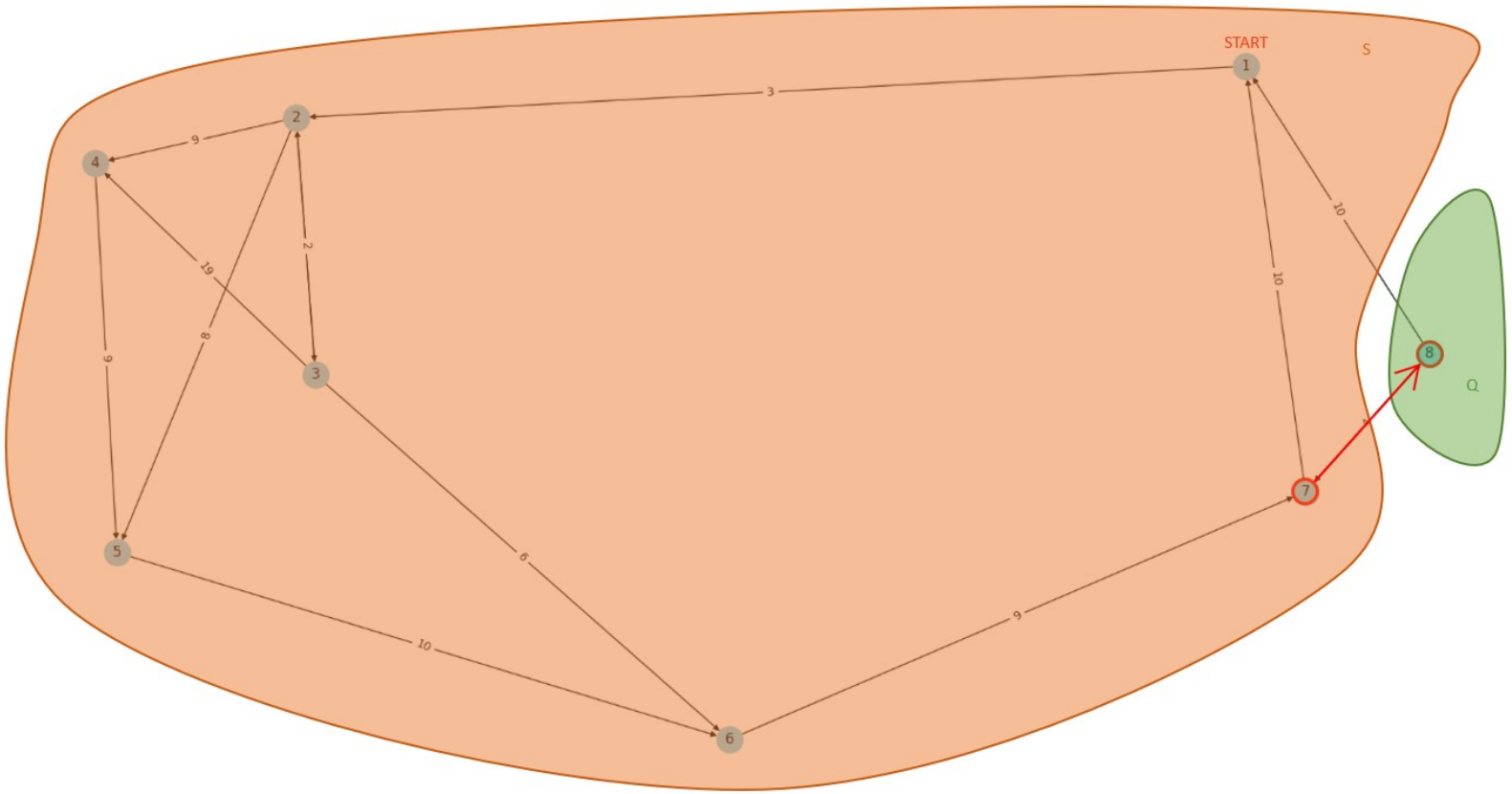
Przykład użycia algorytmu Dijkstry - krok 8

- Analogiczna sytuacja zachodzi w przypadku wierzchołka 4.



Przykład użycia algorytmu Dijkstry - krok 9

- W przedostatnim kroku do zbioru S przenosimy wierzchołek 7. Jego sąsiadem jest wierzchołek 8.



Przykład użycia algorytmu Dijkstry - krok 9

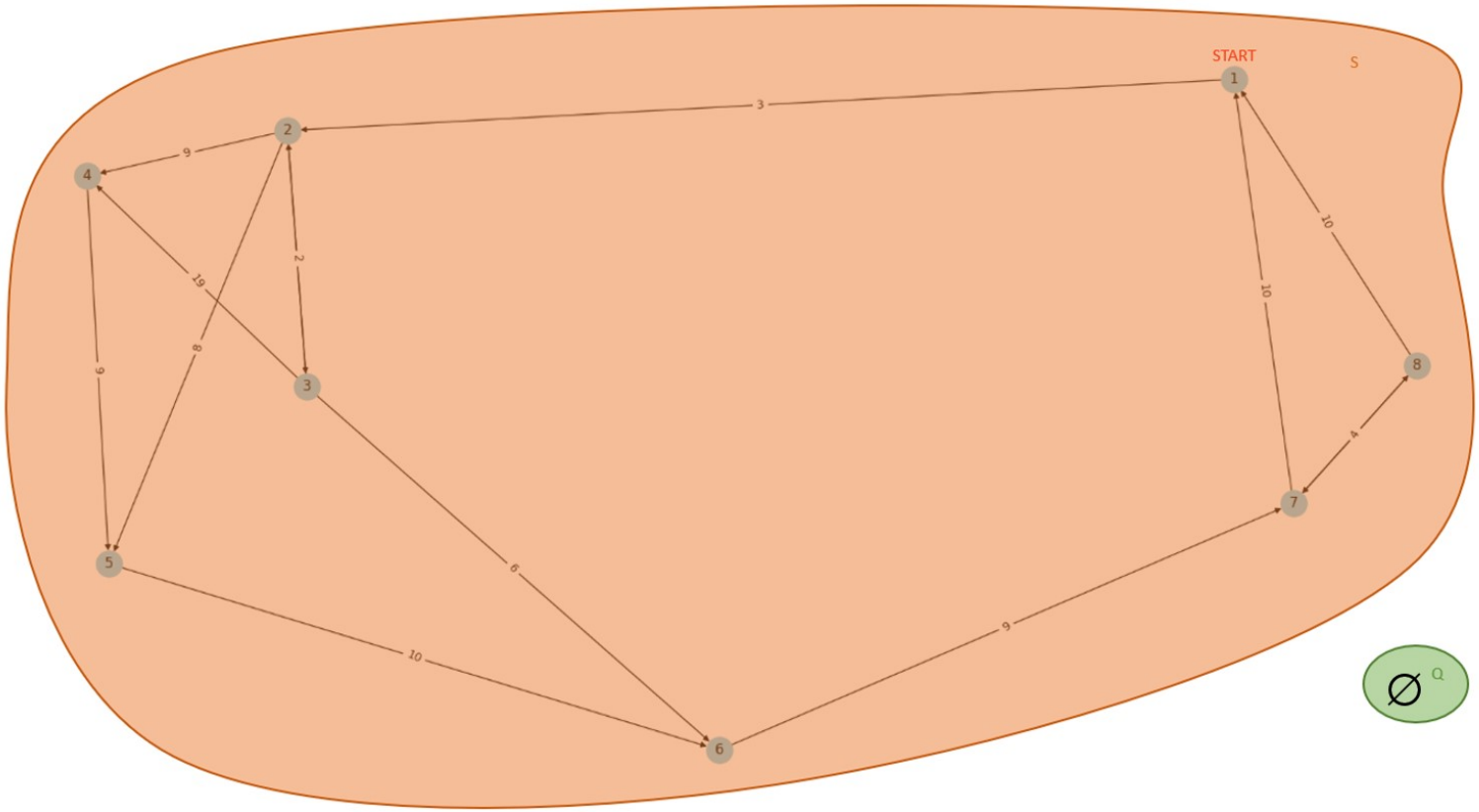
$d_8 > d_7 + 4 \Rightarrow d_8 := d_7 + 4 = 24$, więc $p_8 = 7$

Tabela: Tabela dla kroku 9

v	1	2	3	4	5	6	7	8
d_v	0	3	5	12	11	11	20	24
p_v	0	1	2	2	2	3	6	7

Przykład użycia algorytmu Dijkstry - krok 10

- Zbiór Q stał się pusty, zatem przeniesiony wierzchołek 8 nie ma w zbiorze Q żadnych sąsiadów. Algorytm kończy się.



Przykład użycia algorytmu Dijkstry - podsumowanie

Tabela: Ścieżki i odległości od wierzchołka źródłowego

Wierzchołek	Najkrótsza ścieżka dojścia	Koszt dojścia
1	Ścieżka pusta	0
2	1 - 2	3
3	1 - 2 - 3	5
4	1 - 2 - 4	12
5	1 - 2 - 5	11
6	1 - 2 - 3 - 6	11
7	1 - 2 - 3 - 6 - 7	20
8	1 - 2 - 3 - 6 - 7 - 8	24

Algorytm Bellmana-Forda-Moora

Opis algorytmu

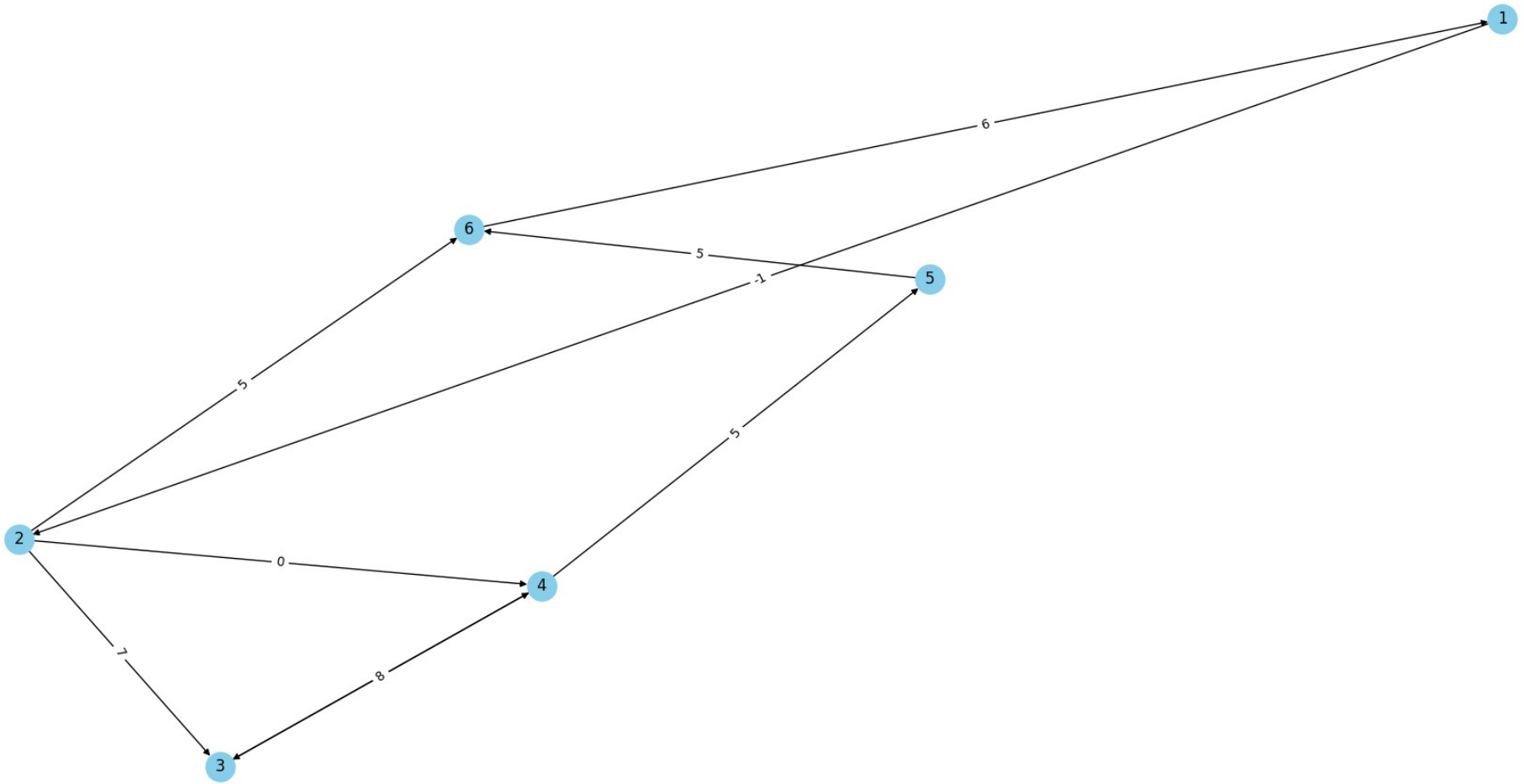
Algorytm Bellmana-Forda-Moora służy do znajdowania najkrótszych ścieżek w grafie ważonym, nawet w przypadku występowania krawędzi o ujemnych wagach.

Działanie algorytmu - opis słowny

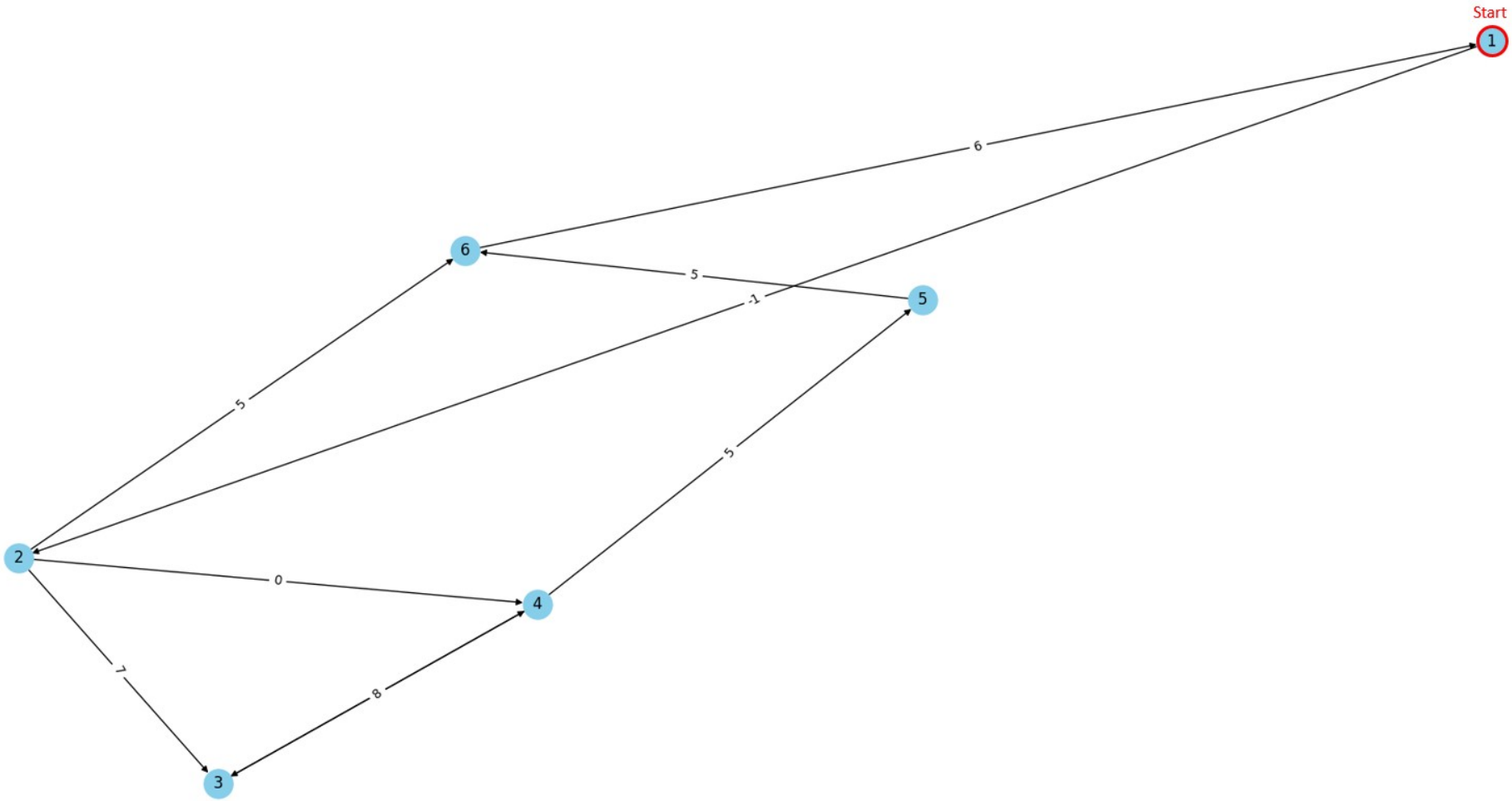
1. Inicjalizacja odległości od źródłowego wierzchołka do pozostałych jako nieskończoność, a odległość od samego siebie jako 0.
2. Iteracyjne relaksowanie krawędzi, aktualizacja odległości.
3. Powtarzanie kroków 1 i 2 $|V| - 1$ razy, gdzie $|V|$ to liczba wierzchołków w grafie.

Przykład użycia algorytmu Bellmana-Forda-Moora

Wyznaczmy odległości od wierzchołka 1 do wszystkich pozostałych wierzchołków w zadanym grafie skierowanym G.



Przykład użycia algorytmu Bellmana-Forda-Moora - krok 1

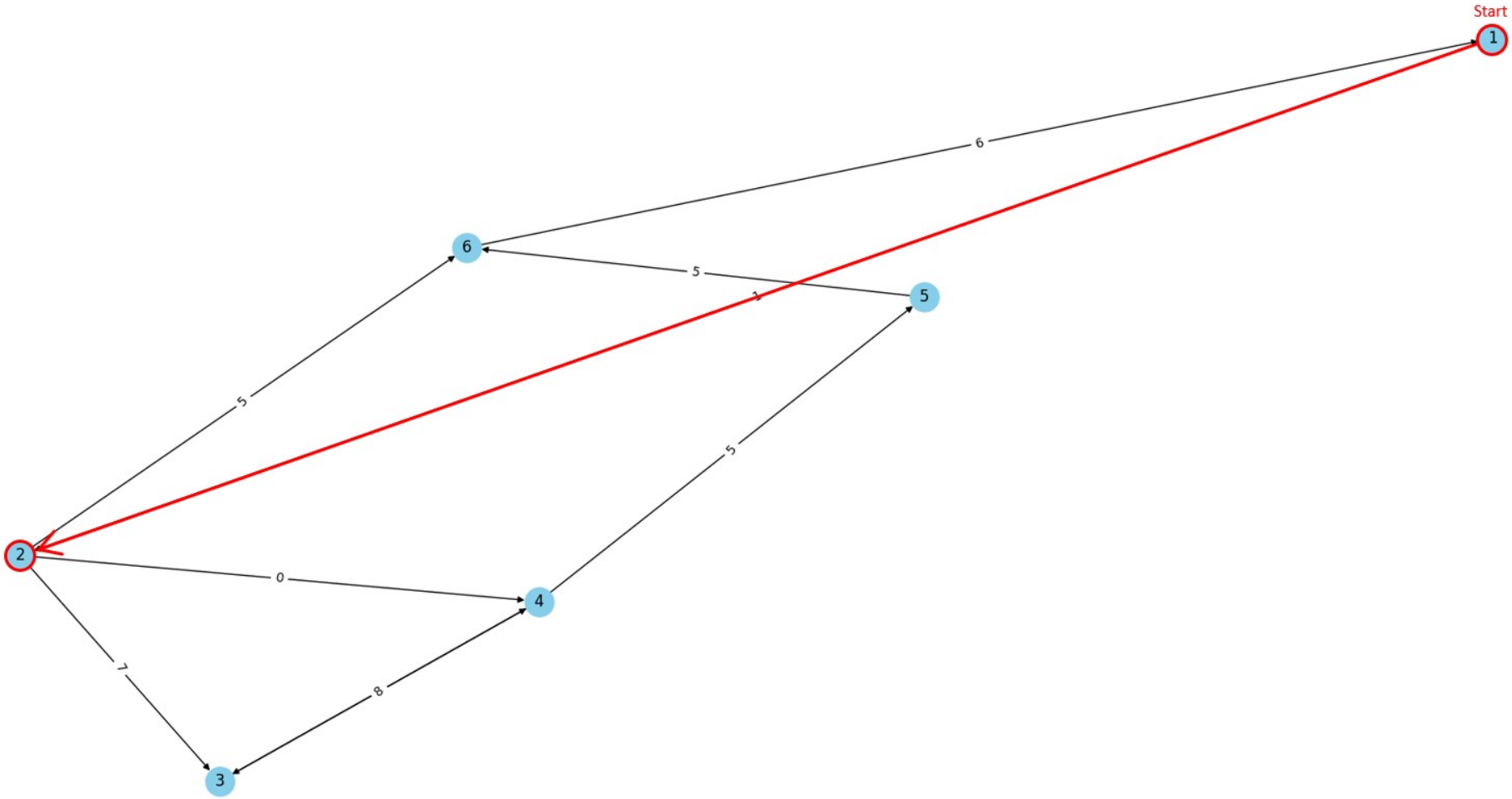


Przykład użycia algorytmu Bellmana-Forda-Moora - krok 1

Tabela: Tabela dla kroku 1

v	1	2	3	4	5	6
d_v	0	∞	∞	∞	∞	∞
p_v	0	0	0	0	0	0

Przykład użycia algorytmu Bellmana-Forda-Moora - krok 2



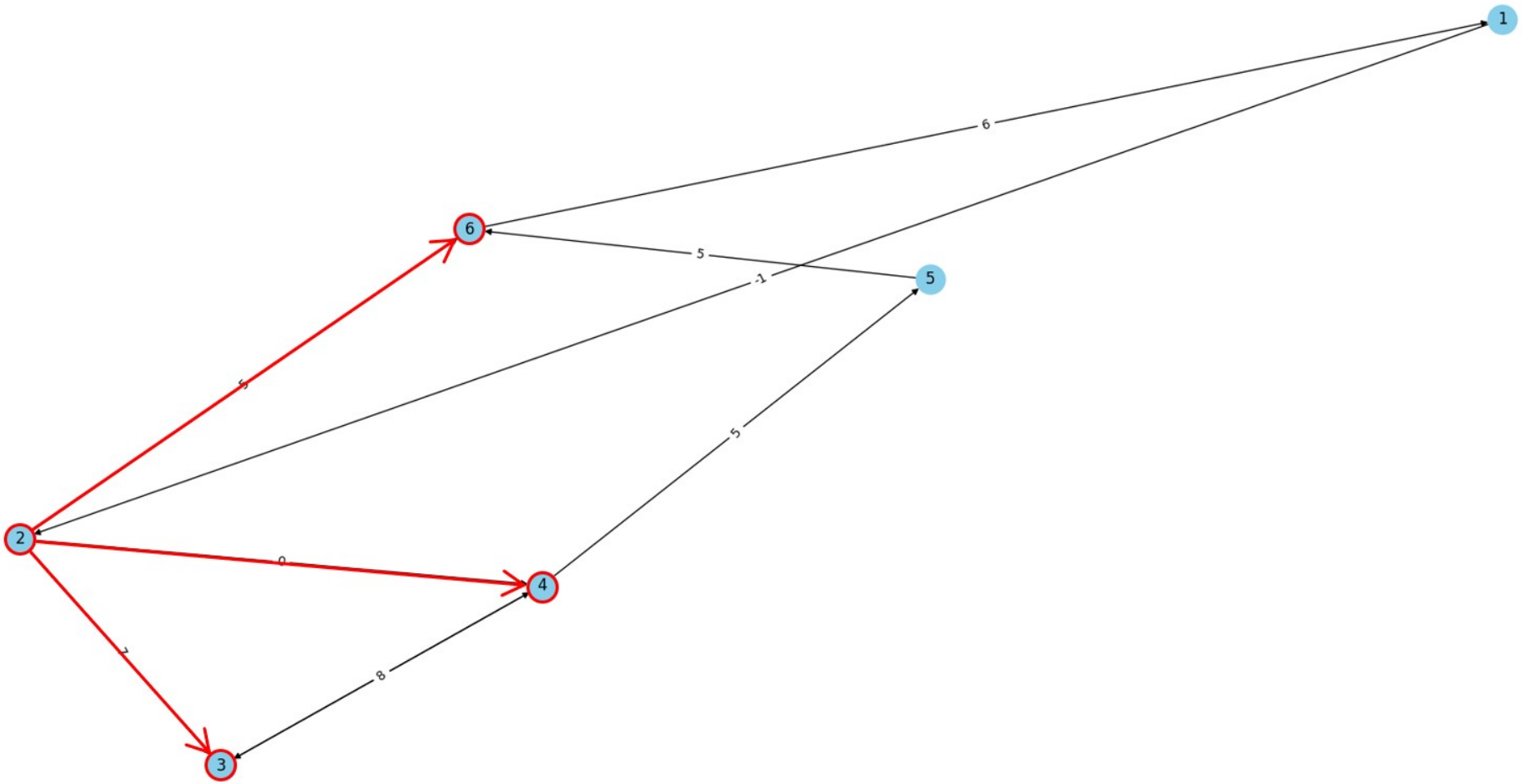
Przykład użycia algorytmu Bellmana-Forda-Moora - krok 2

$d_2 > d_1 - 1 \Rightarrow d_2 := d_1 - 1 = -1$, więc $p_1 = -1$

Tabela: Tabela dla kroku 2

v	1	2	3	4	5	6
d_v	0	-1	∞	∞	∞	∞
p_v	0	1	0	0	0	0

Przykład użycia algorytmu Bellmana-Forda-Moora - krok 3



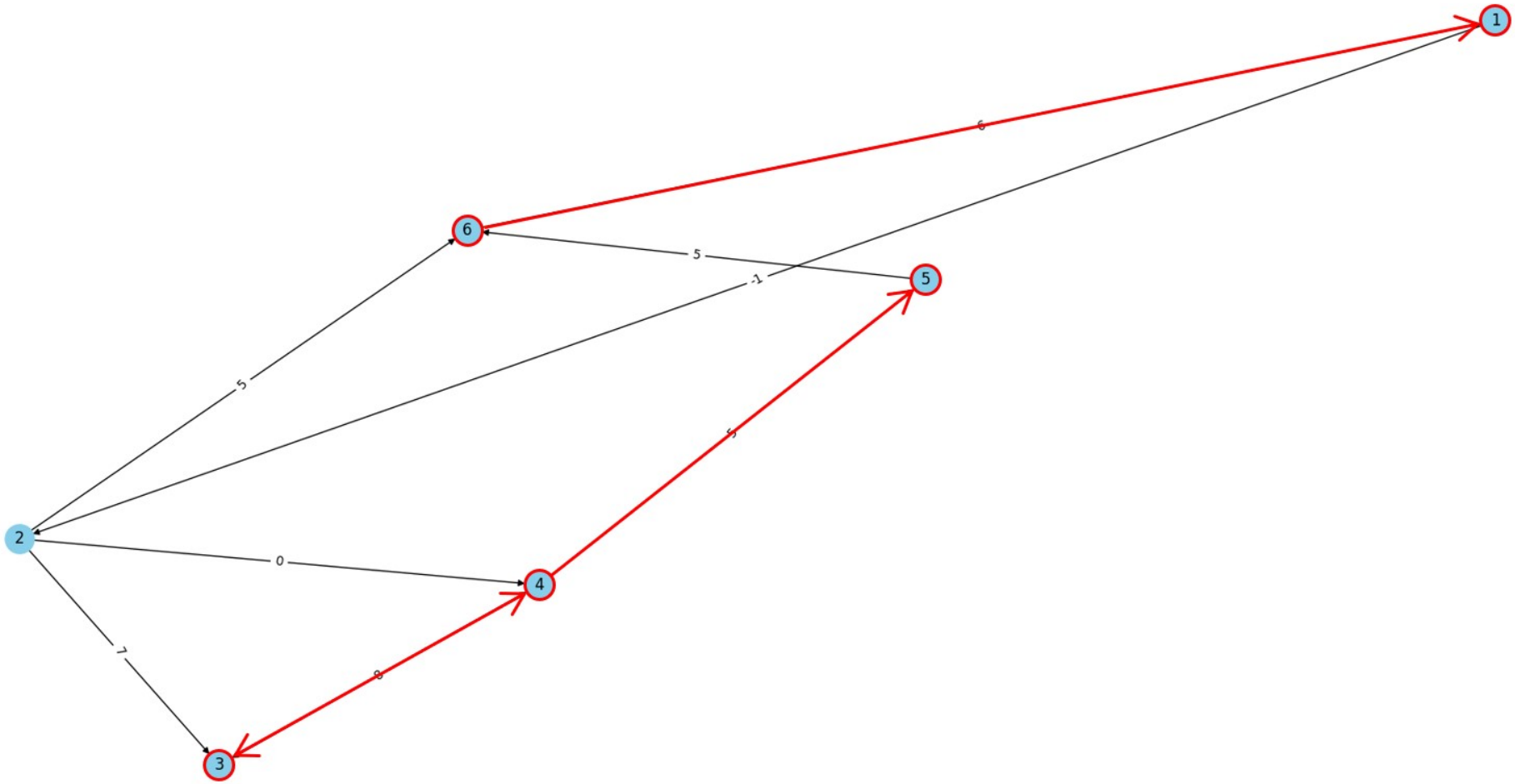
Przykład użycia algorytmu Bellmana-Forda-Moora - krok 3

$d_3 > d_2 + 7 \Rightarrow d_6 := d_2 + 7 = 6$
$d_4 > d_2 + 0 \Rightarrow d_6 := d_2 + 0 = -1$
$d_6 > d_2 + 5 \Rightarrow d_6 := d_2 + 5 = 4$
$p_3 = p_4 = p_6 = 2$

Tabela: Tabela dla kroku 3

v	1	2	3	4	5	6
d_v	0	-1	6	-1	∞	4
p_v	0	1	2	2	0	2

Przykład użycia algorytmu Bellmana-Forda-Moora - krok 4



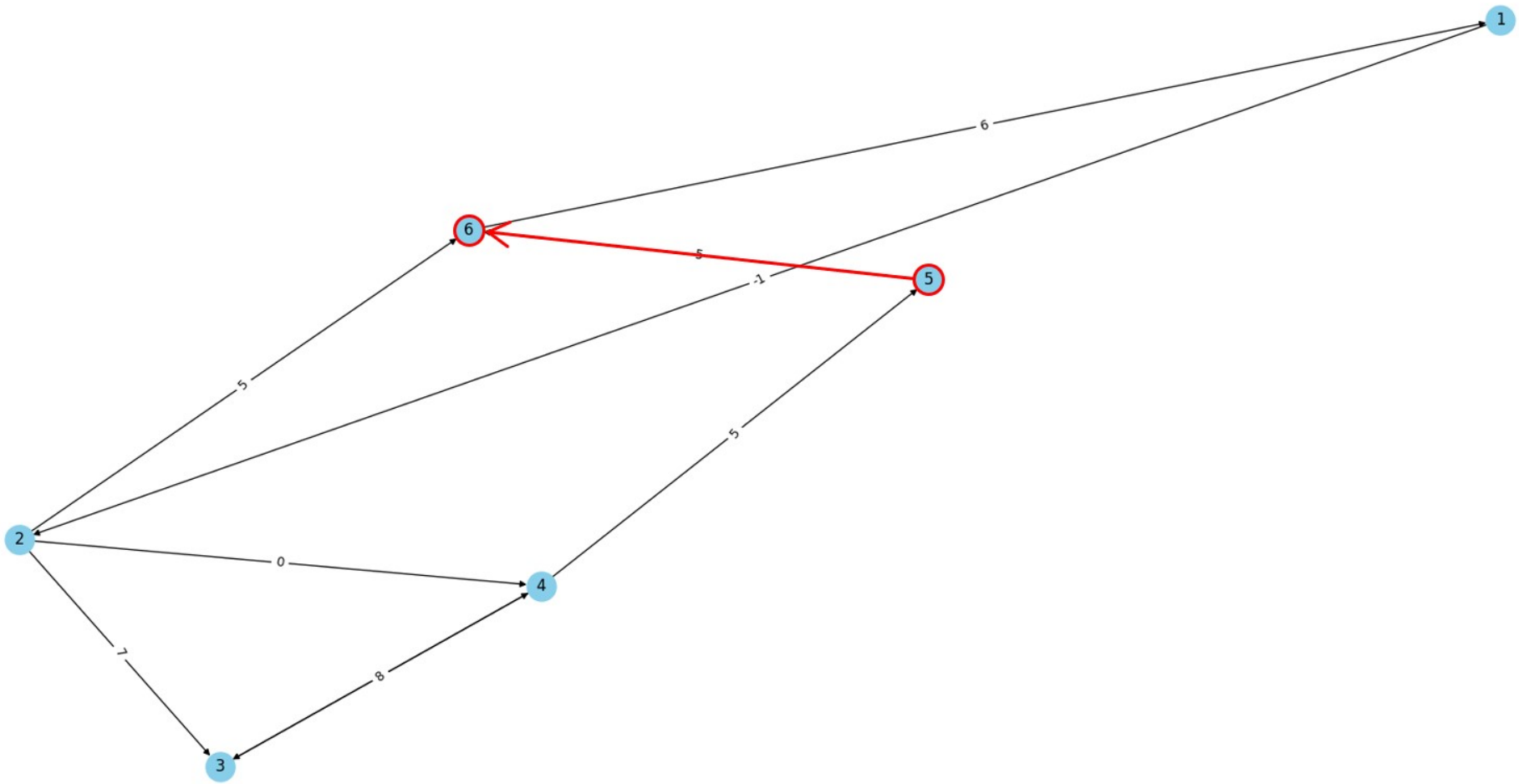
Przykład użycia algorytmu Bellmana-Forda-Moora - krok 4

$d_3 < d_4 + 7$
$d_4 < d_3 + 9$
$d_1 < d_6 + 6$
$d_5 > d_4 + 5 \Rightarrow d_5 := d_4 + 5 = 4$, , więc $p_5 = 4$

Tabela: Tabela dla kroku 4

v	1	2	3	4	5	6
d_v	0	-1	6	-1	4	4
p_v	0	1	2	2	4	2

Przykład użycia algorytmu Bellmana-Forda-Moora - krok 5



Przykład użycia algorytmu Bellmana-Forda-Moora - krok 5

$d_6 < d_5 + 5$

Tabela: Tabela dla kroku 5

v	1	2	3	4	5	6
d_v	0	-1	6	-1	4	4
p_v	0	1	2	2	4	2

Dalsze sprawdzanie nie wprowadzi już żadnych zmian, ponieważ warunek relaksacji nie będzie spełniony dla żadnej z krawędzi. Tabela po obiegu piątym nie ulegnie żadnym zmianom w stosunku do tej po obiegu czwartym.

Algorytm Floyda-Warshalla

Opis algorytmu

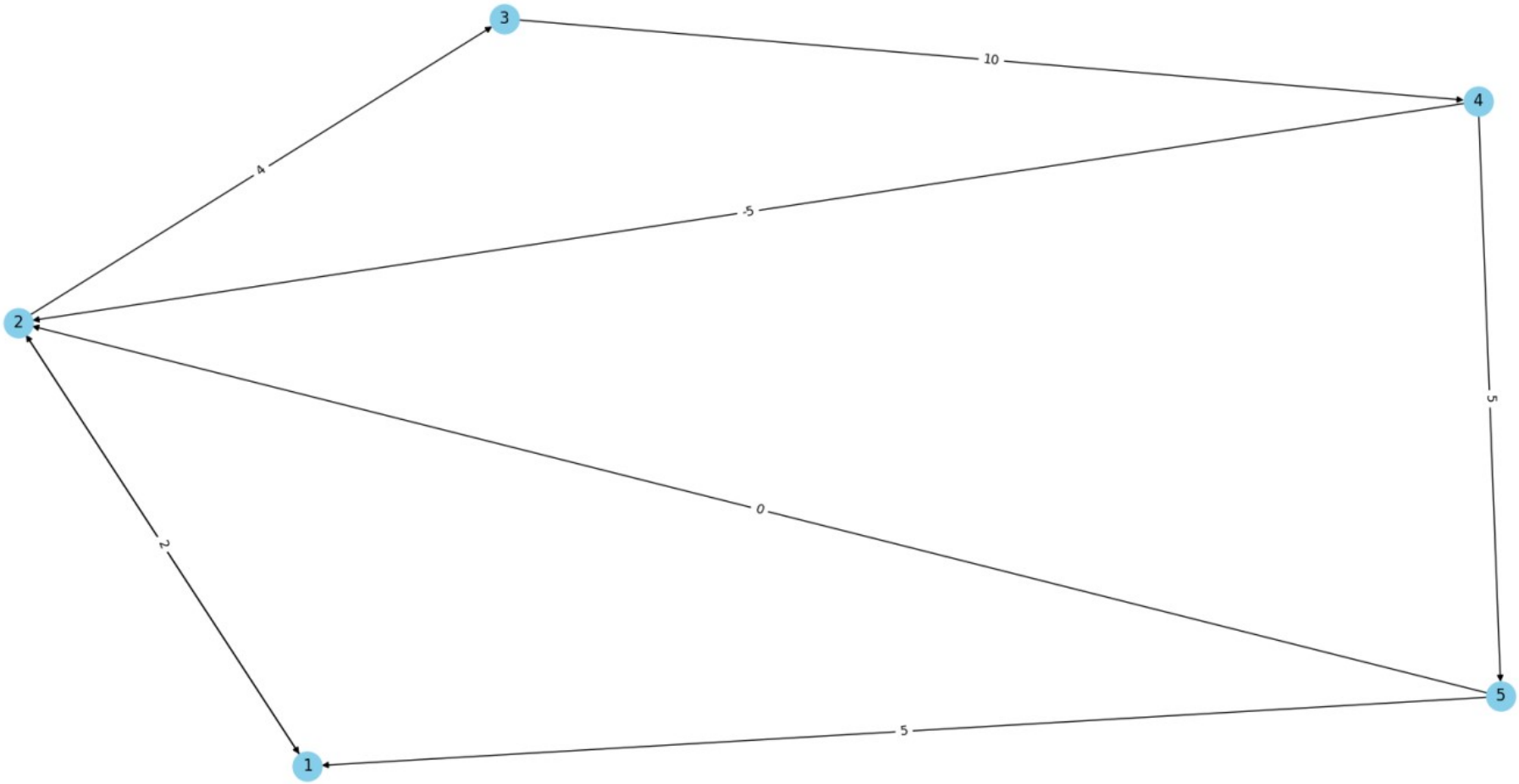
Algorytm Floyda-Warshalla służy do znajdowania najkrótszych ścieżek między wszystkimi parami wierzchołków w grafie ważonym, nawet w przypadku występowania krawędzi o ujemnych wagach.

Działanie algorytmu - opis słowny

1. Inicjalizacja macierzy odległości, gdzie element (i, j) oznacza wagę krawędzi między wierzchołkami i i j lub ∞, jeśli nie istnieje krawędź.
2. Iteracyjne relaksowanie krawędzi poprzez dodanie kolejnych wierzchołków jako pośrednich w ścieżkach.
3. Powtarzanie kroku 2 dla wszystkich par wierzchołków.

Przykład użycia algorytmu Floyda-Warshalla

Wyznaczmy najkrótsze ścieżki między wszystkimi parami wierzchołków w zadanym grafie skierowanym G.



Przykład użycia algorytmu Floyda-Warshalla - krok 1

Przygotowujemy macierz odległości D o wymiarach n x n, gdzie każdy jej element (i, j) oznacza wagę krawędzi między wierzchołkami i i j.
Na ten momement, na głównej przekątnej umieszczamy 0, a w pozostałych komórkach umieszczamy ∞.

Przykładowo, gdyby istniała krawędź o początku w wierzchołku 1 i końcu w wierzchołku 2, to w macierzy odległości uzupełnilibyśmy element d_{12} wartością, będącą wagą tej krawędzi.

$$D = \begin{bmatrix} 0 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 2

Wedle reguły opisanej w drugiej części kroku 1, uzupełniamy macierz odległości D odpowiednimi wagami krawędzi ze sobą incydentnych. Widząc, że istnieje krawędź o początku w wierzchołku 5 i końcu w wierzchołku 1, element d_{51} uzupełnimy wagą tej krawędzi, czyli w tym przypadku 5.

Macierz odległości po odczytaniu wag z grafu:

$$D = \begin{bmatrix} 0 & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & \infty & \infty \\ \infty & \infty & 0 & 10 & \infty \\ \infty & -5 & \infty & 0 & 5 \\ 5 & 0 & \infty & \infty & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 3

Rozpoczynamy analizę kolejnych wierzchołków grafu, zaczynając od wierzchołka $k = 1$. Dla każdego elementu d_{ij} sprawdzamy, czy zachodzi nierówność:

$$d_{ij} > d_{ik} + d_{kj}.$$

Jeśli tak, to $d_{ij} \leftarrow d_{ik} + d_{kj}$.

Macierz odległości po 1 iteracji:

$$D = \begin{bmatrix} 0 & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & \infty & \infty \\ \infty & \infty & 0 & 10 & \infty \\ \infty & -5 & \infty & 0 & 5 \\ 5 & 0 & \infty & \infty & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 4

$$d_{13} = \infty > d_{12} + d_{23} = 2 + 4 = 6$$
$$d_{41} = \infty > d_{42} + d_{21} = -5 + 2 = -3$$
$$d_{43} = \infty > d_{42} + d_{21} = -5 + 4 = -1$$
$$d_{51} = 5 > d_{52} + d_{21} = 0 + 2 = 2$$
$$d_{53} = \infty > d_{52} + d_{23} = 0 + 4 = 4$$

Macierz odległości po 2 iteracji:

$$D = \begin{bmatrix} 0 & 2 & 6 & \infty & \infty \\ 2 & 0 & 4 & \infty & \infty \\ \infty & \infty & 0 & 10 & \infty \\ -3 & -5 & -1 & 0 & 5 \\ 2 & 0 & 4 & \infty & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 5

$$d_{14} = \infty > d_{13} + d_{34} = 6 + 10 = 16$$
$$d_{24} = \infty > d_{23} + d_{34} = 4 + 10 = 14$$
$$d_{54} = \infty > d_{53} + d_{34} = 4 + 10 = 14$$

Macierz odległości po 3 iteracji:

$$D = \begin{bmatrix} 0 & 2 & 6 & 16 & \infty \\ 2 & 0 & 4 & 14 & \infty \\ \infty & \infty & 0 & 10 & \infty \\ -3 & -5 & -1 & 0 & 5 \\ 2 & 0 & 4 & 14 & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 6

$$d_{15} = \infty > d_{14} + d_{45} = 16 + 5 = 21$$
$$d_{25} = \infty > d_{24} + d_{45} = 14 + 5 = 19$$
$$d_{31} = \infty > d_{34} + d_{41} = 10 - 3 = 7$$
$$d_{32} = \infty > d_{34} + d_{42} = 10 - 5 = 5$$
$$d_{35} = \infty > d_{34} + d_{45} = 10 + 5 = 15$$

Macierz odległości po 4 iteracji:

$$D = \begin{bmatrix} 0 & 2 & 6 & 16 & 21 \\ 2 & 0 & 4 & 14 & 19 \\ 7 & 5 & 0 & 10 & 15 \\ -3 & -5 & -1 & 0 & 5 \\ 2 & 0 & 4 & 14 & 0 \end{bmatrix}$$

Przykład użycia algorytmu Floyda-Warshalla - krok 7

Żaden z elementów macierzy odległości D nie ulega zmianom.

Macierz odległości po 5 iteracji:

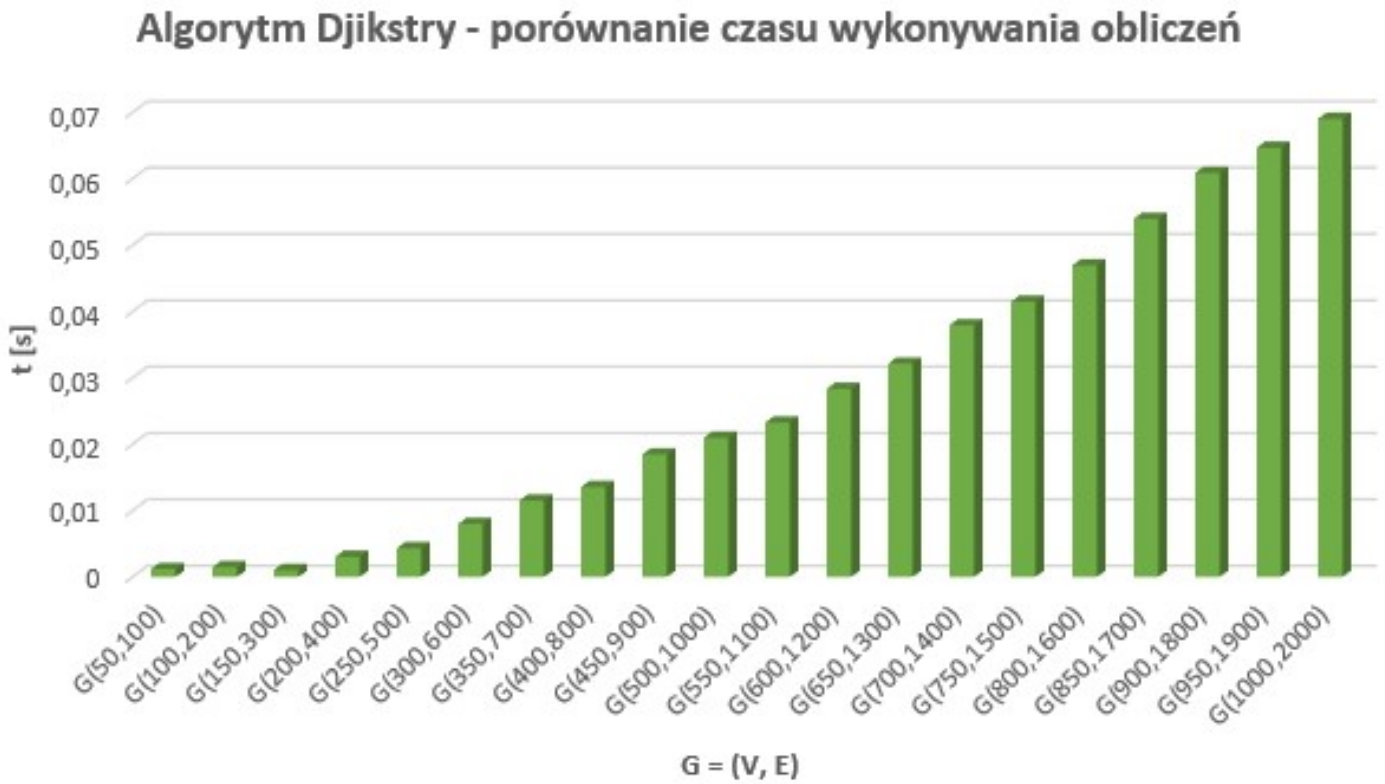
$$D = \begin{bmatrix} 0 & 2 & 6 & 16 & 21 \\ 2 & 0 & 4 & 14 & 19 \\ 7 & 5 & 0 & 10 & 15 \\ -3 & -5 & -1 & 0 & 5 \\ 2 & 0 & 4 & 14 & 0 \end{bmatrix}$$

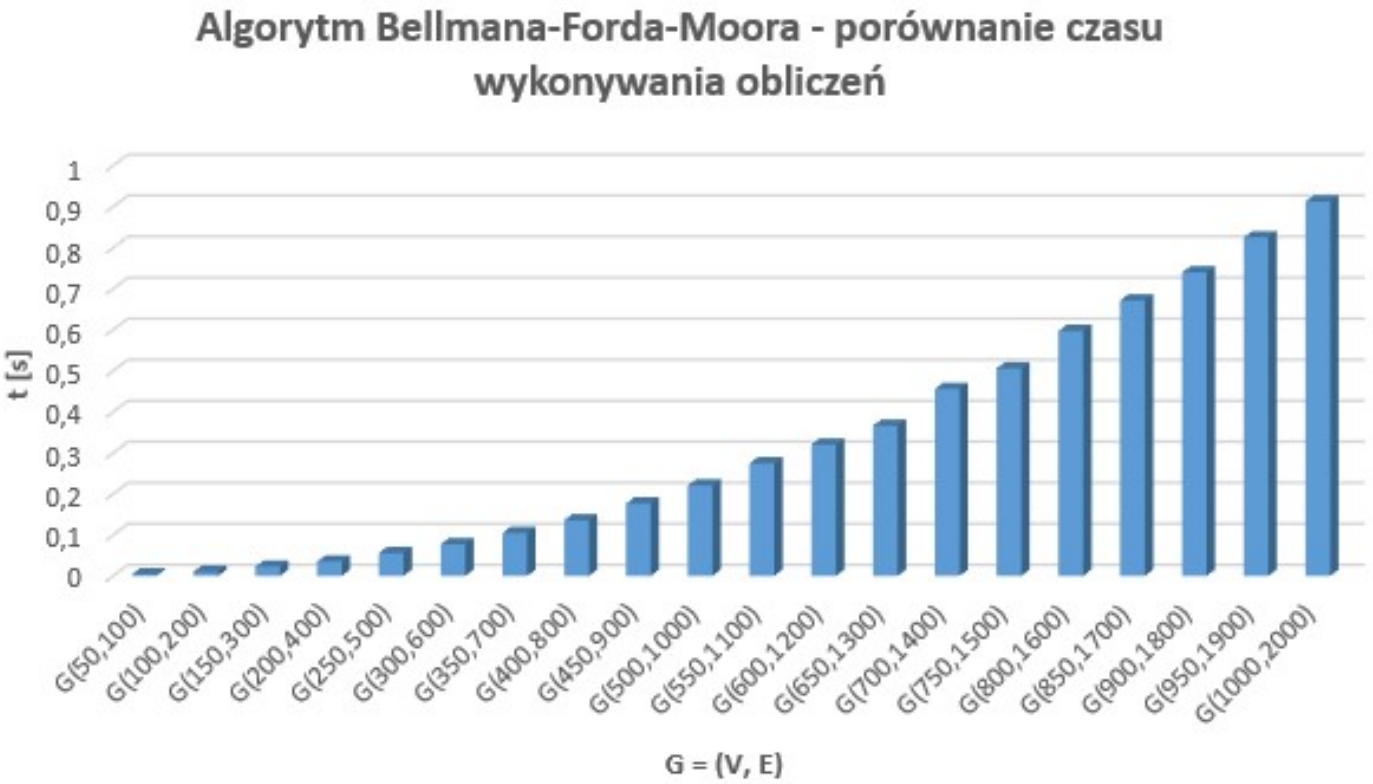
Porównywanie czasu wykonywania obliczeń

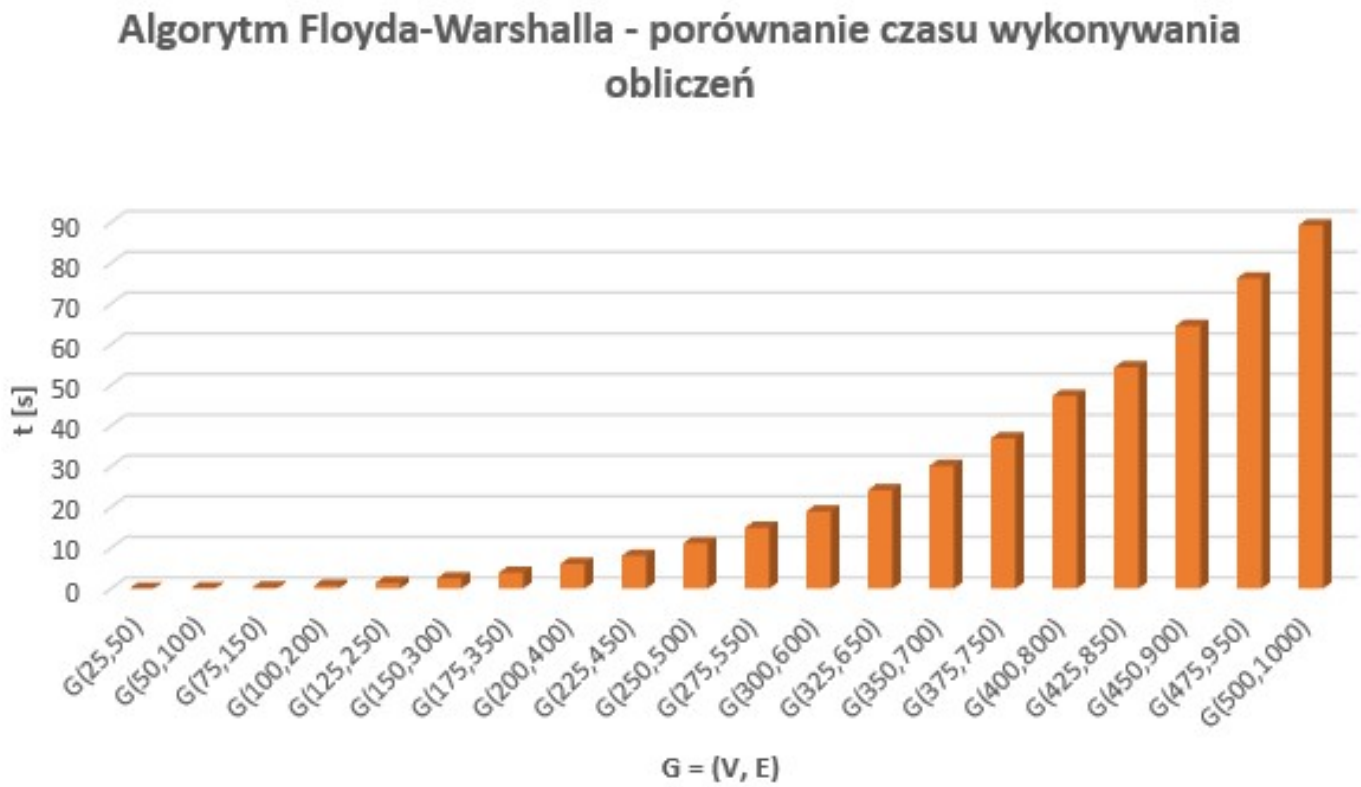
Zajmiemy się teraz badaniem czasu wykonywania obliczeń dla różnych grafów $G = (V, E)$, będących pod wpływem działania naszych algorytmów optymalizacyjnych:

1. Algorytm Dijkstry,
2. Algorytm Bellmana-Forda,
3. Algorytm Floyda-Warshalla.

Przyjęliśmy założenie, że liczba krawędzi musi być dwukrotnie większa od liczby wierzchołków, a czas mierzyć będziemy w sekundach.


Algorytm Djikstry - porównanie czasu wykonywania obliczeń
t [s]
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
G(50,100)
G(100,200)
G(150,300)
G(200,400)
G(250,500)
G(300,600)
G(350,700)
G(400,800)
G(450,900)
G(500,1000)
G(550,1100)
G(600,1200)
G(650,1300)
G(700,1400)
G(750,1500)
G(800,1600)
G(850,1700)
G(900,1800)
G(950,1900)
G(1000,2000)
G = (V, E)


Algorytm Bellmana-Forda-Moora - porównanie czasu wykonywania obliczeń
t [s]
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
G(50,100)
G(100,200)
G(150,300)
G(200,400)
G(250,500)
G(300,600)
G(350,700)
G(400,800)
G(450,900)
G(500,1000)
G(550,1100)
G(600,1200)
G(650,1300)
G(700,1400)
G(750,1500)
G(800,1600)
G(850,1700)
G(900,1800)
G(950,1900)
G(1000,2000)
G = (V, E)


Algorytm Floyda-Warshalla - porównanie czasu wykonywania obliczeń
t [s]
90
80
70
60
50
40
30
20
10
0
G(25,50)
G(50,100)
G(75,150)
G(100,200)
G(125,250)
G(150,300)
G(175,350)
G(200,400)
G(225,450)
G(250,500)
G(275,550)
G(300,600)
G(325,650)
G(350,700)
G(375,750)
G(400,800)
G(425,850)
G(450,900)
G(475,950)
G(500,1000)
G = (V, E)

Podsumowanie

Wnioski

Algorytmy Dijkstry, Bellmana-Forda-Moora oraz Floyda-Warshalla są skutecznymi narzędziami do rozwiązywania problemów najkrótszych dróg w grafach ważonych, różniącymi się swoją złożonością obliczeniową oraz zastosowaniami.

- Algorytm Dijkstry jest najbardziej efektywny dla grafów o dodatnich wagach krawędzi i jednym wierzchołku źródłowym.
- Algorytm Bellmana-Forda-Moora może obsługiwać grafy z ujemnymi wagami krawędzi, ale ma złożoność czasową $O(V \cdot E)$, co czyni go mniej wydajnym dla dużych grafów.
- Algorytm Floyda-Warshalla znajduje najkrótsze ścieżki między wszystkimi parami wierzchołków w grafie, ale jego złożoność obliczeniowa wynosi $O(V^3)$, co sprawia, że może być niepraktyczny dla bardzo dużych grafów.