ALGORYTM DIJKSTRY - Dokumentacja

# Krótki opis algorytmu Dijkstry

Algorytm Dijkstry, opracowany przez holenderskiego informatyka Edsgera Dijkstrę w 1956 roku, służy do znajdowania najkrótszych ścieżek od jednego wierzchołka źródłowego do wszystkich pozostałych wierzchołków w grafie ważonym o nieujemnych wagach krawędzi.  
  
Algorytm w każdym kroku wybiera wierzchołek o najmniejszej dotychczas znanej odległości od źródła i aktualizuje odległości do jego sąsiadów.

**Kluczowe cechy algorytmu:**

* Działa dla grafów skierowanych i nieskierowanych
* Wymaga nieujemnych wag krawędzi
* Gwarantuje znalezienie optymalnych rozwiązań
* Złożoność czasowa: O(V²) w podstawowej implementacji

# UWAGA - Uruchamianie aplikacji

Przy pierwszym uruchomieniu pliku **.exe** system Windows może wyświetlić ostrzeżenie zabezpieczeń. Aby uruchomić aplikację:

* 1. Kliknij "Więcej informacji" (More info)
* 2. Następnie kliknij "Uruchom mimo to" (Run anyway)

# Instrukcja obsługi aplikacji

## Krok 1: Ustawienie parametrów grafu

* W polu "Liczba wierzchołków" wprowadź liczbę wierzchołków (zalecane: 5-15)
* Wybierz typ grafu: "Skierowany" lub "Nieskierowany"
* Kliknij przycisk "Generuj graf"

## Krok 2: Generowanie grafu

* Aplikacja automatycznie wygeneruje losowy graf z połączeniami
* Graf zostanie wyświetlony w dolnej części okna
* W górnej części pojawią się informacje o wygenerowanym grafie

## Krok 3: Uruchomienie algorytmu

* W polu "Wierzchołek startowy" wprowadź numer wierzchołka (0 do n-1)
* Kliknij przycisk "Uruchom algorytm"
* Wyniki pojawią się w górnym panelu aplikacji

## Krok 4: Analiza wyników

* Przejrzyj tabelę najkrótszych ścieżek
* Przeanalizuj szczegółowe kroki algorytmu
* Sprawdź podsumowanie z analizą złożoności

# Szczegółowy opis działania algorytmu Dijkstry

Algorytm Dijkstry opiera się na strategii zachłannej i składa się z trzech głównych faz:  
  
1. INICJALIZACJA:  
 • Dla wierzchołka startowego ustaw odległość = 0  
 • Dla wszystkich pozostałych wierzchołków ustaw odległość = ∞  
 • Utwórz zbiór nieodwiedzonych wierzchołków  
  
2. GŁÓWNA PĘTLA:  
 • Wybierz nieodwiedzony wierzchołek o najmniejszej odległości  
 • Oznacz go jako odwiedzony  
 • Dla każdego sąsiada sprawdź czy przez aktualny wierzchołek można dotrzeć taniej  
 • Jeśli tak, zaktualizuj odległość i poprzednika  
  
3. ZAKOŃCZENIE:  
 • Algorytm kończy się gdy wszystkie osiągalne wierzchołki zostały odwiedzone  
 • Otrzymujemy najkrótsze odległości i ścieżki

# Przykład działania algorytmu

Rozważmy nieskierowany graf o 5 wierzchołkach z następującymi połączeniami:

Obraz zawierający tekst, linia, diagram

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Graf składa się z 5 wierzchołków: A(0), B(1), C(2), D(3), E(4)  
Krawędzie z wagami:  
• A ↔ B (waga: 4) • A ↔ C (waga: 2) • B ↔ C (waga: 1)  
• B ↔ D (waga: 5) • C ↔ D (waga: 8) • C ↔ E (waga: 10) • D ↔ E (waga: 2)  
  
Wierzchołek startowy: A (wierzchołek 0)

## Tabela kroków algorytmu Dijkstry

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr kroku** | **Opis operacji** | **Kolejka nieodwiedzonych** | **Odległości [A,B,C,D,E]** | **Aktualizacje** |
| 0 | Inicjalizacja | {A,B,C,D,E} | [0,∞,∞,∞,∞] | start=A |
| 1 | Wybrano A | {B,C,D,E} | [0,4,2,∞,∞] | B:∞→4, C:∞→2 |
| 2 | Wybrano C (odl=2) | {B,D,E} | [0,3,2,10,12] | B via C: 4→3,  D via C: ∞→10, E via C: ∞→12 |
| 3 | Wybrano B (odl=3) | {D,E} | [0,3,2,8,12] | D via B: 10→8 |
| 4 | Wybrano D (odl=8) | {E} | [0,3,2,8,10] | E via D: 12→10 |
| 5 | Wybrano E (odl=10) | {} | [0,3,2,8,10] | Zakończenie |

## Wyniki końcowe

|  |  |  |
| --- | --- | --- |
| **Wierzchołek docelowy** | **Najkrótsza odległość** | **Najkrótsza ścieżka** |
| A | 0 | A |
| B | 3 | A → C → B |
| C | 2 | A → C |
| D | 8 | A → C → B → D |
| E | 10 | A → C → B → D → E |

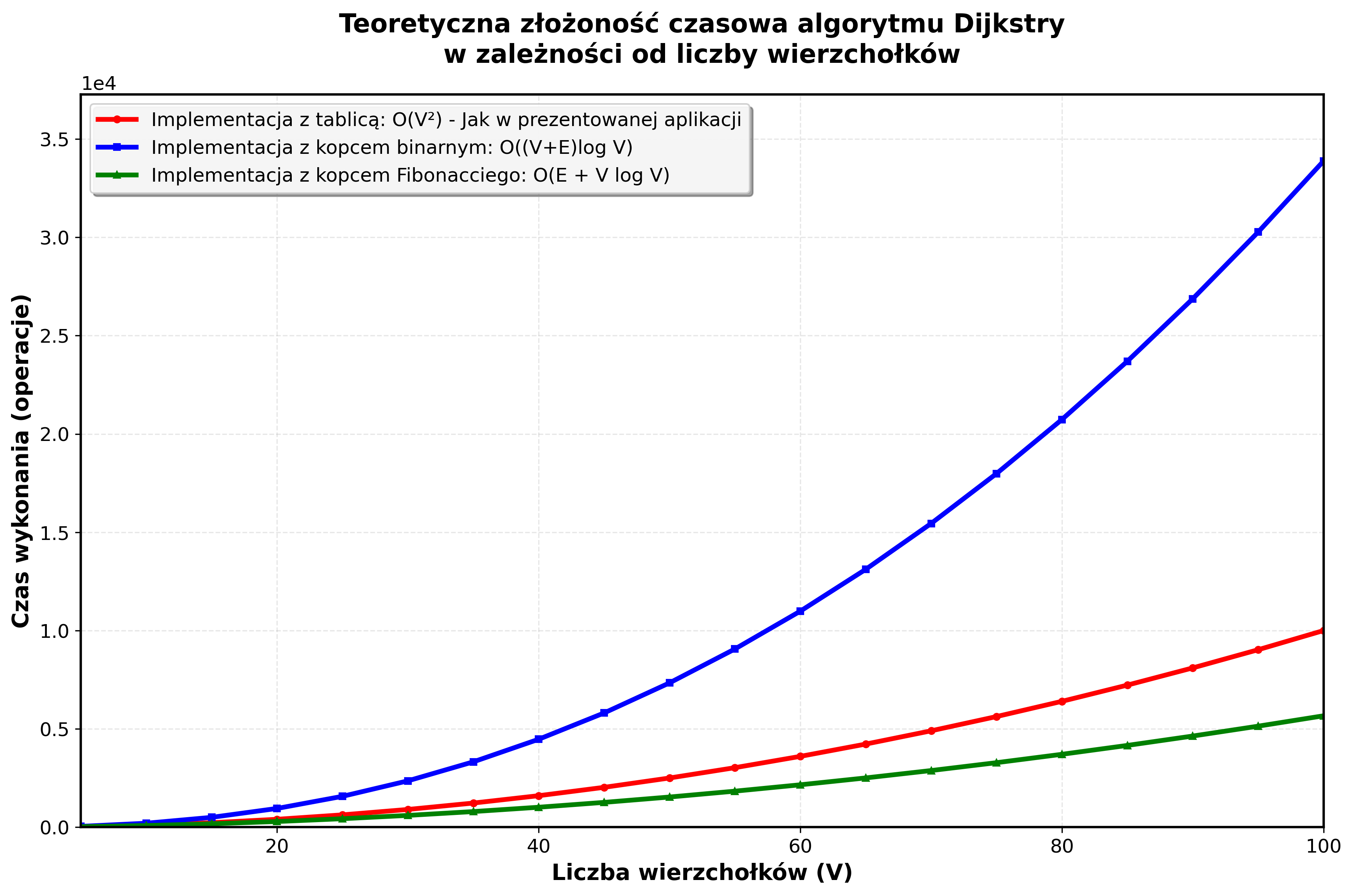
# Analiza złożoności algorytmu Dijkstry

Złożoność czasowa algorytmu Dijkstry zależy od implementacji:  
  
• Nasza implementacja (z tablicą): O(V²)  
• Z kopcem binarnym: O((V+E)log V)   
• Z kopcem Fibonacciego: O(E + V log V)  
**Złożoność pamięciowa: O(V) - dla tablic odległości i poprzedników**

## Obliczenie dla podanego przykładu

Dane grafu:  
• Liczba wierzchołków (V): 5  
• Liczba krawędzi (E): 7  
  
Złożoność dla naszego przykładu:  
• Nasza implementacja: O(V²) = O(5²) = O(25)  
• Z kopcem binarnym: O((V+E)log V) = O((5+7)log 5) ≈ O(28)  
• Z kopcem Fibonacciego: O(E + V log V) = O(7 + 5×2.32) ≈ O(19)  
**Liczba operacji wykonanych w naszym przykładzie: około 29 operacji podstawowych**

## Wykres złożoności czasowej



## Linki do materiałów: kod w pythonie oraz całe repozytorium w Github.

<https://colab.research.google.com/drive/19i49eOIMC0r1lZ5Bfzepc4YMgYlEm_rX?usp=sharing>

<https://github.com/sebastian-c87/Dijkstry-App>