# Floyd-Warshall

## Svi najkraći putevi u grafu

*Luka Horvat*

*17. 1. 2016.*

# Problem

- zadan je usmjeren, otežan graf
- dan je broj vrhova i lista bridova s pripadajućim težinama
- kako odrediti duljinu najkraćeg puta između svih parova vrhova?

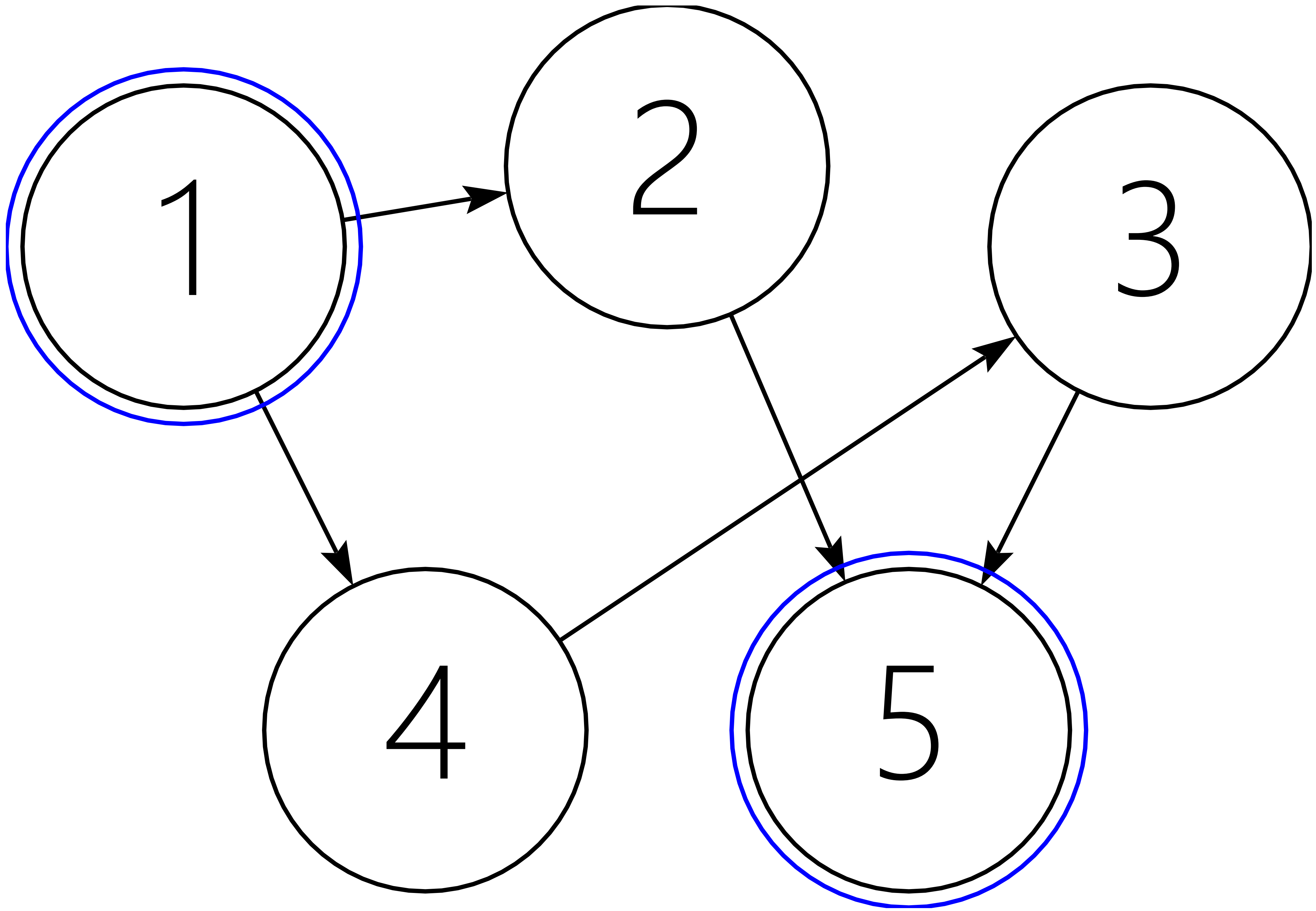
# Rekurzivno rješenje

- ukoliko znamo najkraće duljine puteva između svih parova vrhova, ali samo onih puteva koji prolaze kroz vrhove `1` do `k`, možemo li naći najkraće duljine puteva koji mogu prolaziti i kroz vrh `k + 1`?
- definiramo `dist(i, j, k)` kao duljinu najkraćeg puta od vrha `i` do `j` koji može prolaziti samo vrhovima od `1` do `k`
- ako uključimo `i` vrh `k + 1`, najkraći put postaje minimum od
  - `dist(i, j, k)`
  - `dist(i, k + 1, k)` + `dist(k + 1, j, k)`

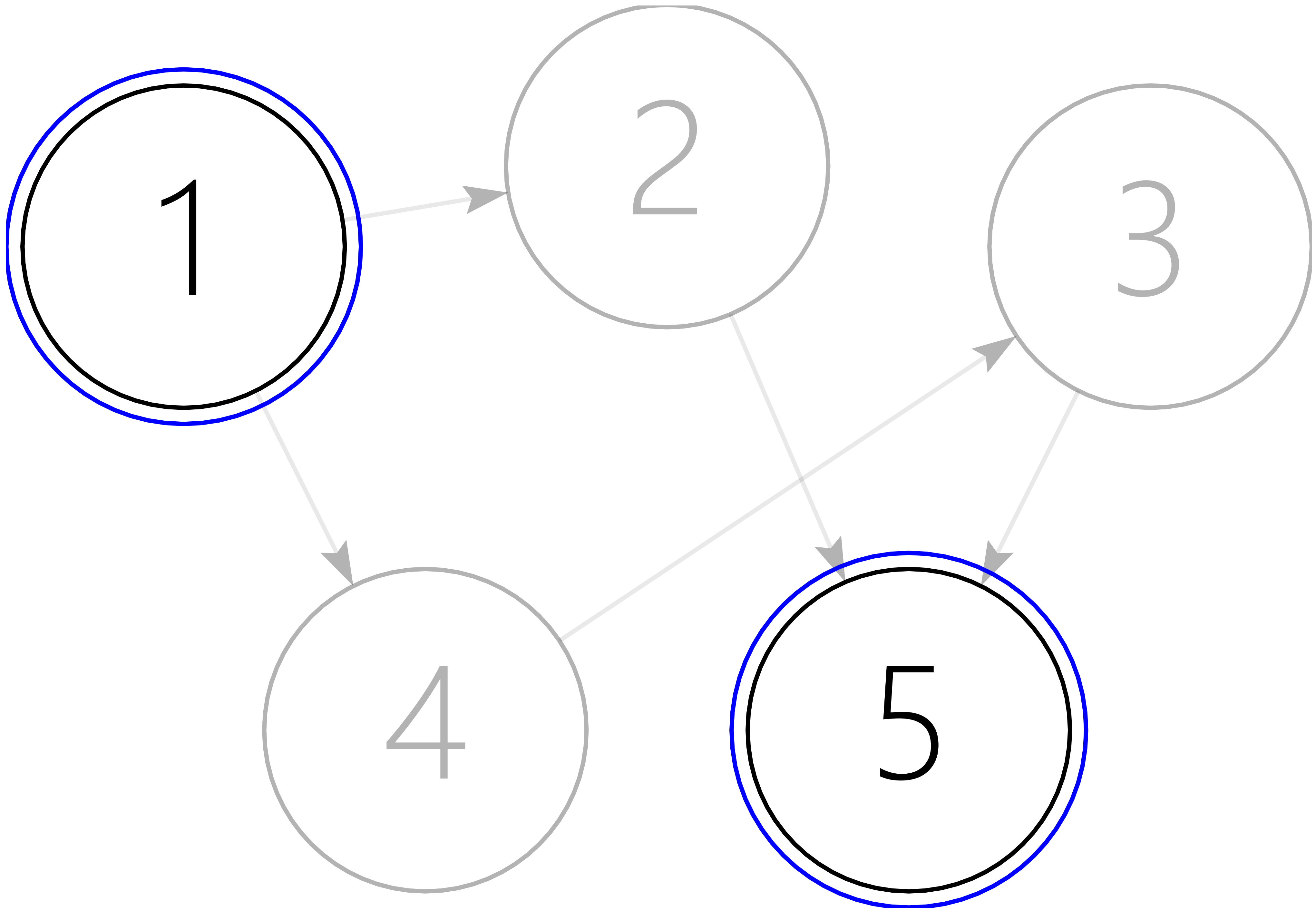
# Zašto?

- budući da znamo sve najkraće puteve koji prolaze kroz vrhove `1` do `k` znamo i duljine puteva od `i` do `k + 1`, te od `k + 1` do `j`
- ako je njihov zbroj manji od trenutno najkraće duljine od `i` do `j` onda je bolje odabrati put koji prvo ide do `k + 1`, a tek onda do `j`
- taj put koristi samo vrhove od `1` do `k`, i još dodatno vrh `k + 1`
- konačna rekurzivna definicija: `dist(i, j, k) = min (dist(i, j, k - 1), dist(i, k, k - 1) + dist(k, j, k - 1))`

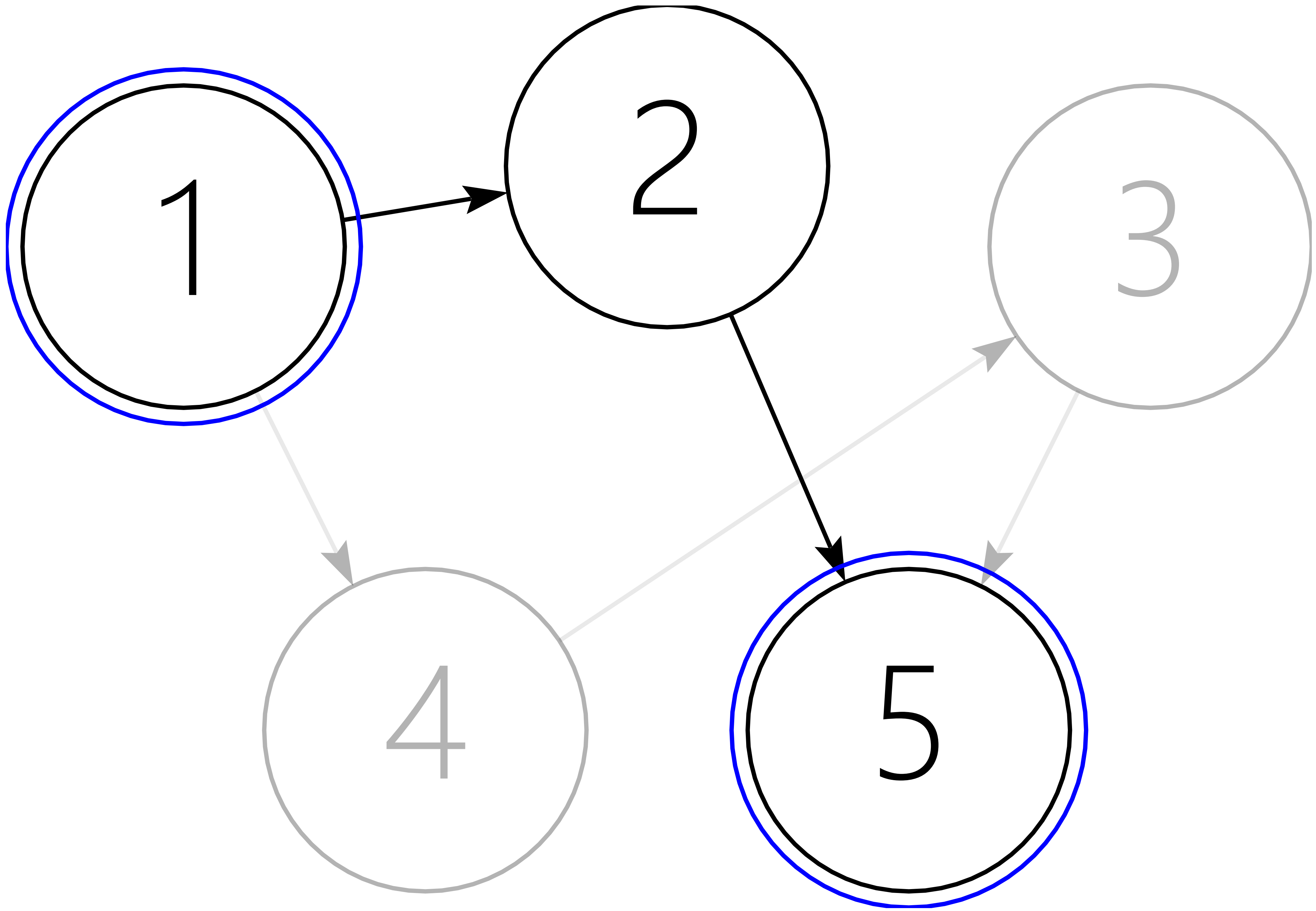
# Primjer



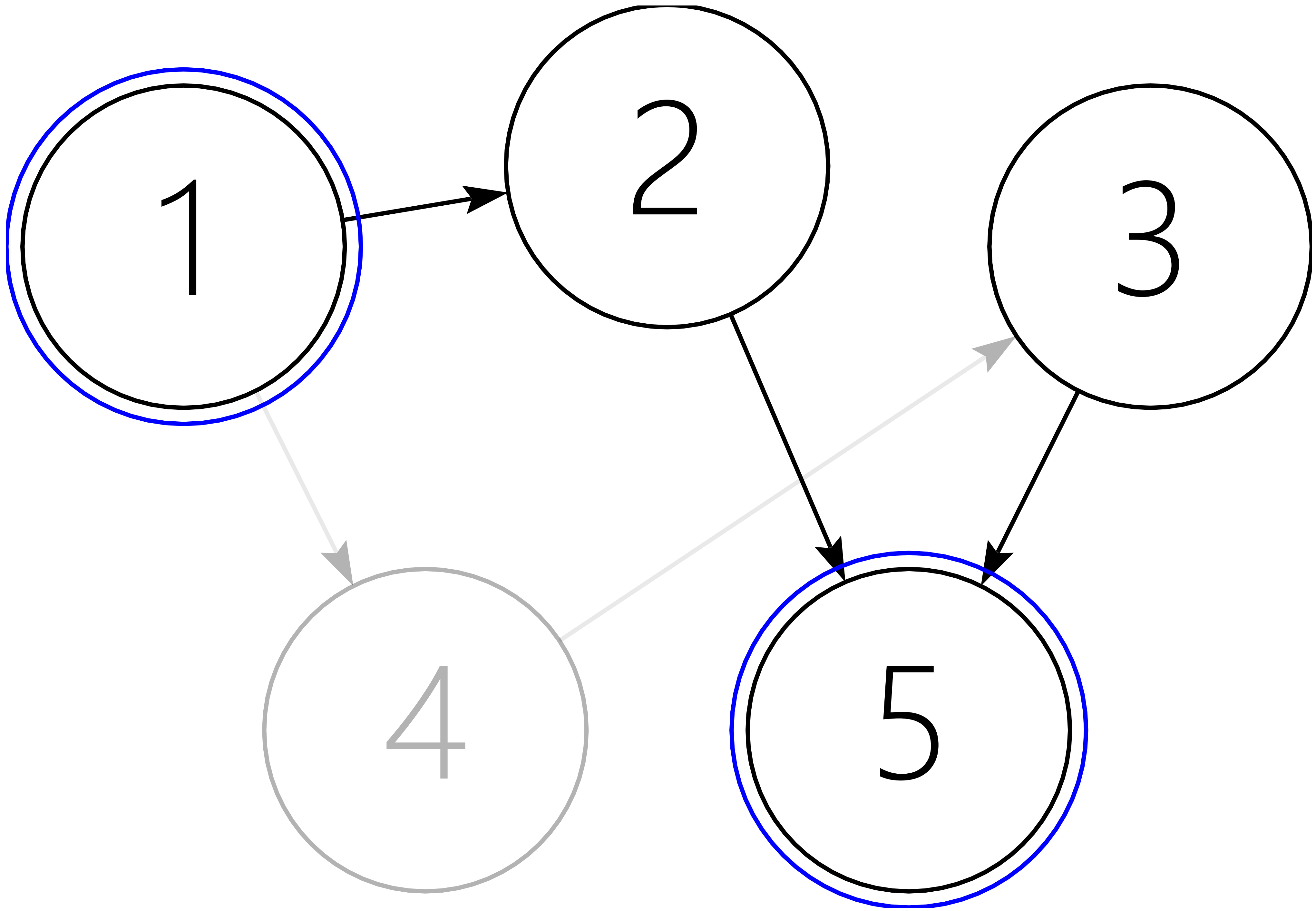
# Primjer



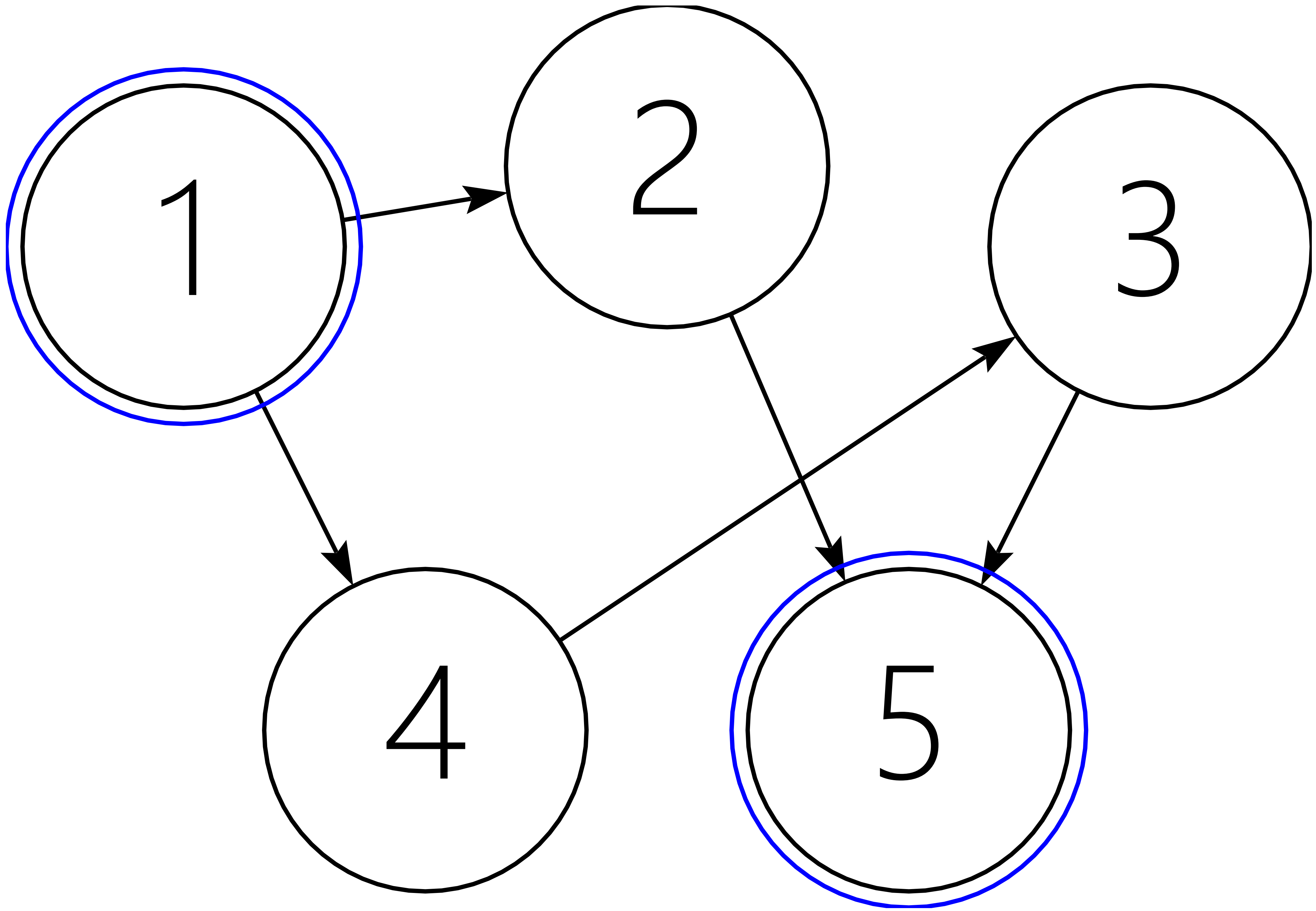
# Primjer



# Primjer



# Primjer



# Dinamičko rješenje

- ako promatamo povećanje parametra `k` kao jednu iteraciju algoritma možemo primjetiti da svaka iteracija ovisi isključivo o rezultatima prethodne
- vođeni tom idejom možemo optimizirati algoritam
- ako spremamo rezultate poziva funkcije `dist`, ne trebamo koristiti trodimenzionalnu matricu već samo dvodimenzionalnu
- štoviše, dovoljno je koristiti samo jednu dvodimenzionalnu matricu i unutar iteracije čitati i pisati po istoj operaciji

# Pseudo kod

```
dist ← matrica |V| × |V| s elementima postavljenim na ∞
za svaki vrh v
    dist[v][v] ← 0
za svaki brid (u, v)
    dist[u][v] ← duljina brida (u,v)
za k od 1 to |V|
   za i od 1 to |V|
       za j od 1 to |V|
           dist[i][j] ← min(dist[i][j],
                            dist[i][k] + dist[k][j])
```

# Jedna matrica

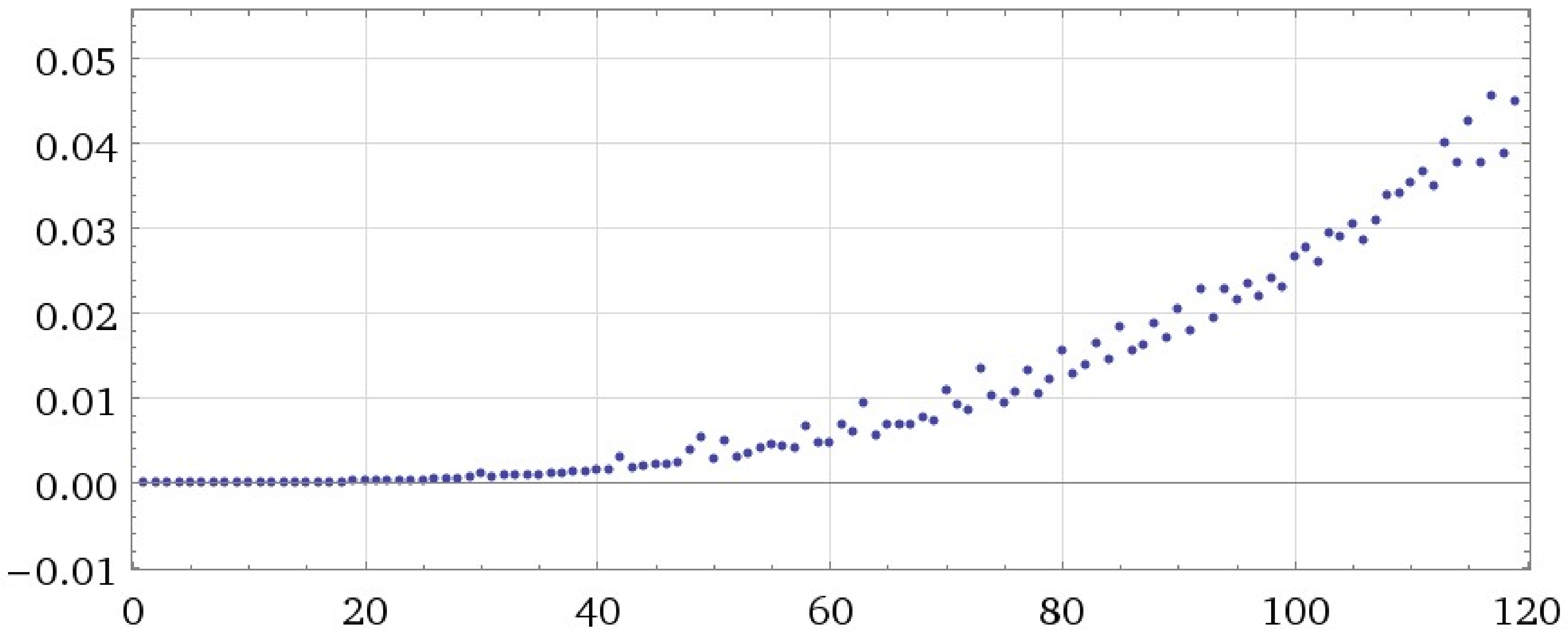
- kako opravdati korištenje samo jedne matrice?
- tokom jedne iteracije pristupamo starim rezultatima u `k`-tom redu i/ili `k`-tom stupcu
- budući da istovremeno pišemo po toj matrici, bitno je da ti elementi ostanu nepromijenjeni
- ako je `i = k`, naša relacija postaje `dist(i, j, k) = min(dist(i, j, k - 1), dist(i, i, k - 1) + dist(i, j, k - 1)) = dist(i, j, k - 1)`
- analogno ako je `j = k`, element matrice ostaje nepromijenjen

# Složenost

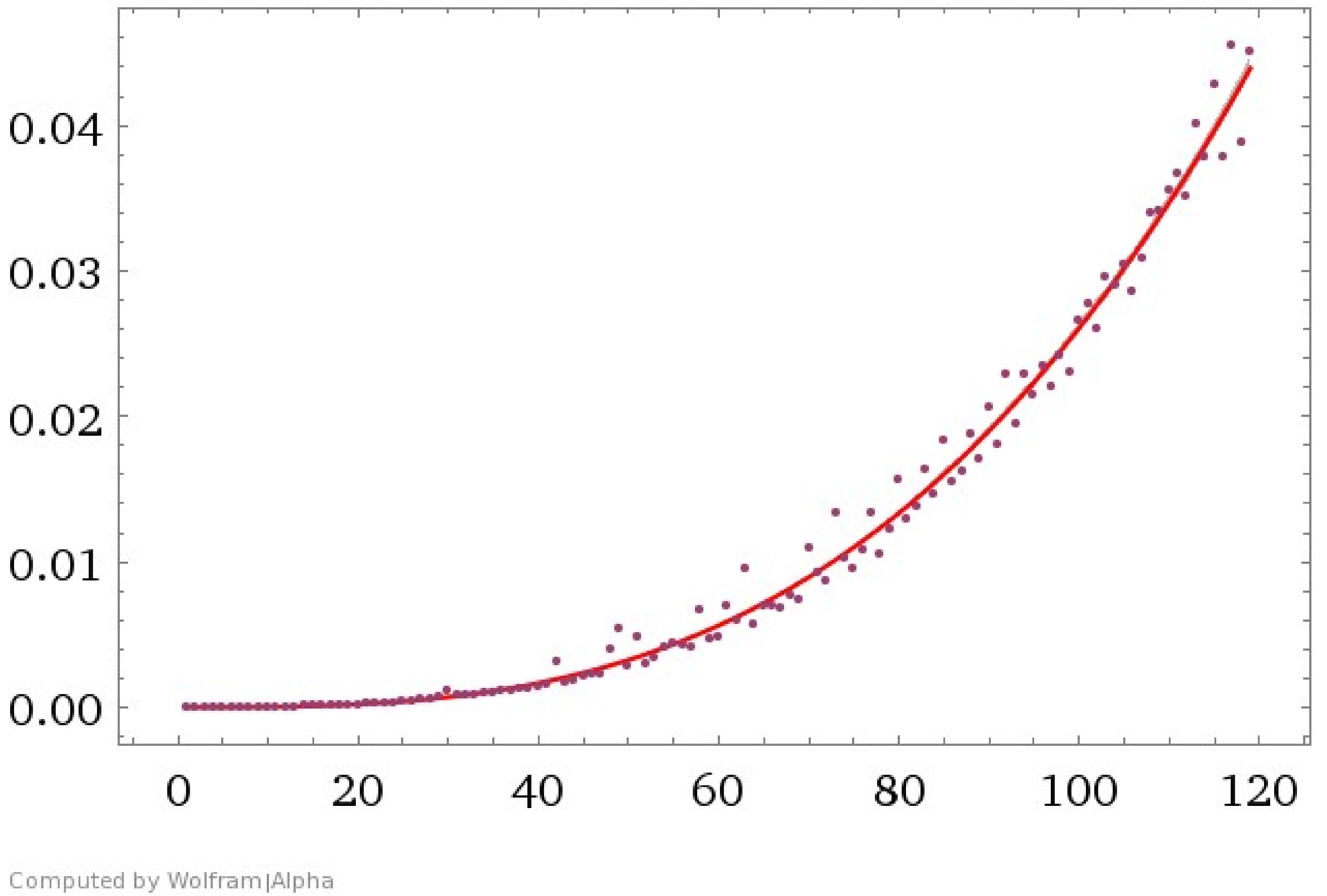
- jedina memorija koju koristimo je matrica veličine $|V|^2$
- algoritam se odvija u tri ugnježdene petlje s fiksnim granicama
- svaka petlja ima $|V|$ iteracija
- zaključak: prostorna složenost je $O(|V|^2)$, a vremenska $O(|V|^3)$

# Testiranje

- algoritam je testiran na grafovima od 1 do 120 vrhova
- bridovi su slučajno generirani
- mjereno je vrijeme izvršavanja algoritma bez vremena za generiranje grafova



- WolframAlpha pronalazi kubičnu krivulju kao najbolju aproksimaciju
- $R^2$=0.991



# Reference

- Algorithms Design Techniques and Analysis - M. H. Alsuwaiyel - 1999