# Algorithmic Strategies 2024/25

# Week 9 – Graph Algorithms

· U C ·

Universidade de Coimbra

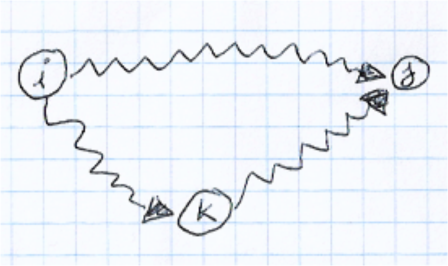
# Floyd-Warshall Algorithm

- Floyd-Warshall Algorithm is a dynamic programming approach that finds the shortest path between every pair of vertices in $O(|V|^3)$
- It can detect negative cycles
- It finds the shortest path with negative weights if there is no negative cycle
- It can also be used to solve reachability questions: whether vertex $i$ can reach vertex $j$.

## Floyd-Warshall Algorithm

Let $D(i, j, k)$ be the solution to the subproblem of finding the shortest path from $i$ to $j$ using the vertices from $\{1, \dots, k\}$ as intermediate vertices.

$$D(i,j,k) = \begin{cases} w(i,j) & \text{if } k = 0 \\ \min \begin{cases} D(i,j,k-1) \\ D(i,k,k-1) + D(k,j,k-1) \end{cases} & \text{otherwise} \end{cases}$$



## Floyd-Warshall Algorithm

**Function** $FW(G, D)$

**for** $i = 1$ to $|V|$ **do**
  $D[i, i, 0] = 0$
**for** each arc $(i, j) \in A$ **do**
  $D[i, j, 0] = w(i, j)$
**for** $k = 1$ to $|V|$ **do**
  **for** $i = 1$ to $|V|$ **do**
    **for** $j = 1$ to $|V|$ **do**
      $D[i, j, k] = \min(D[i, j, k-1], D[i, k, k-1] + D[k, j, k-1])$
**return** $D$

The cells of matrix $D$ are initialized to $\infty$.

## Floyd-Warshall Algorithm

**Function** $FW(G, D)$

  **for** $i = 1$ to $|V|$ **do**
    $D[i,i] = 0$
  **for** each arc $(i,j) \in A$ **do**
    $D[i,j] = w(i,j)$
  **for** $k = 1$ to $|V|$ **do**
    **for** $i = 1$ to $|V|$ **do**
      **for** $j = 1$ to $|V|$ **do**
        $D[i,j] = \min(D[i,j], D[i,k] + D[k,j])$
  **return** $D$

Only a two-dimensional matrix is required

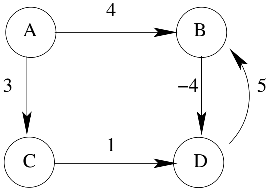
## Floyd-Warshall Algorithm

**Function** $FW(G, D)$

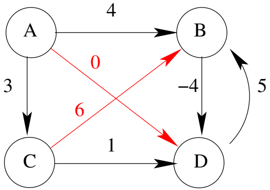
  **for** $i = 1$ to $|V|$ **do**
    $D[i, i] = 0$
  **for** each arc $(i, j) \in A$ **do**
    $D[i, j] = w(i, j)$
  **for** $k = 1$ to $|V|$ **do**
    **for** $i = 1$ to $|V|$ **do**
      **for** $j = 1$ to $|V|$ **do**
        $D[i, j] = \min(D[i, j], D[i, k] + D[k, j])$
  **for** $i = 1$ to $|V|$ **do**
    **if** $D[i, i] < 0$ **then** {check for negative cycle}
      **return** negative cycle found
  **return** $D$

# Floyd-Warshall Algorithm

Find all-pairs shortest path



Find all-pairs shortest path



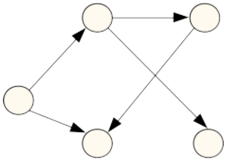

| $i$ | $j$ | $k$ | $D[i,j]$ |
|---|---|---|---|
| $A$ | $D$ | $B$ | 0 |
| $A$ | $D$ | $C$ | 4 |
| $C$ | $B$ | $D$ | 6 |

# Floyd-Warshall Algorithm

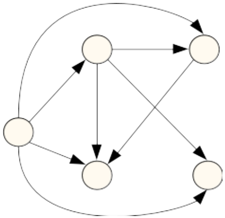
The transitive closure of a graph $G = (V, A)$ is a graph $G^* = (V, A^*)$, where

$$A^* = \{(i,j) : \text{ there is a path from vertex } i \text{ to vertex } j \text{ in G } \}$$

Given a transitive closure, it is possible to know, in constant time, whether one vertice is reachable from another.



A graph



Its transitive closure

## Floyd-Warshall Algorithm for transitive closure

---

**Function** $FW(G, D)$

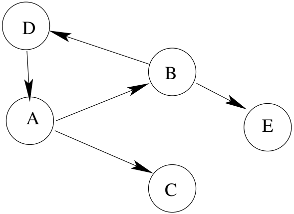
  **for** each arc $(i, j) \in A$ **do**
    $D[i, j] =$ true
  **for** $k = 1$ to $|V|$ **do**
    **for** $i = 1$ to $|V|$ **do**
      **for** $j = 1$ to $|V|$ **do**
        $D[i, j] = D[i, j] \vee (D[i, k] \wedge D[k, j])$
  **return** $D$

---

The cells of matrix $D$ are initialized to false.

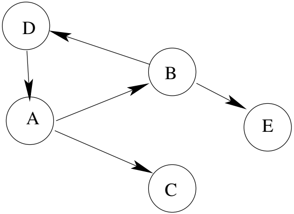
# Floyd-Warshall Algorithm

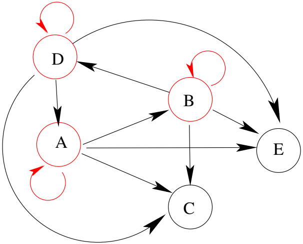
Compilers routinely create so-called "call graphs" during the compilation process. A call graph represents which functions in a program call each other, and may be used to identify functions that are never called, for example. Call graphs may also be used to identify which functions in a program originate recursion. Your task is to write a program that identifies the recursive functions in a given call graph, i.e., functions that call themselves, either directly or through other functions.



# Floyd-Warshall Algorithm

Compilers routinely create so-called "call graphs" during the compilation process. A call graph represents which functions in a program call each other, and may be used to identify functions that are never called, for example. Call graphs may also be used to identify which functions in a program originate recursion. Your task is to write a program that identifies the recursive functions in a given call graph, i.e., functions that call themselves, either directly or through other functions.




Check the main diagonal of the adjacency matrix of the transitive closure