Floyd's Algorithm: Shortest Path Problem

Project 1

Daniel Romero - 2023059668
Adrián Zamora - 2023083307

Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Semester II 2025

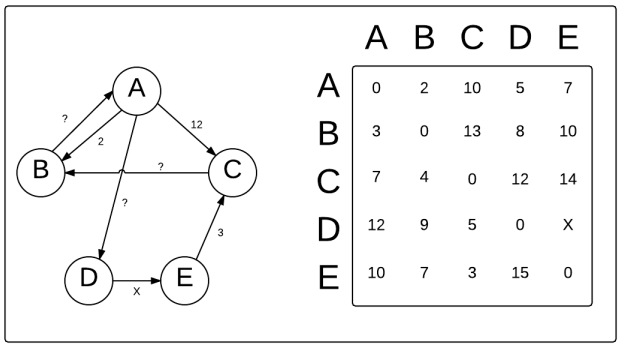
September 12, 2025

Robert W. Floyd

- American computer scientist (1936 - 2001)
- Studied physics at the University of Chicago, B.A. at age **19**
- No formal CS degree $\rightarrow$ self-taught programming and algorithms
- Worked as a math teacher, then in computing $\rightarrow$ professor at Stanford
- Published foundational papers in computational theory
- Collaborated with **Donald Knuth** on "The Art of Computer Programming"
- Created cycle detection and **shortest paths** algorithms
- Received the **Turing Award (1978)**

Floyd Algorithm (Floyd–Warshall Algorithm)

- The Floyd Algorithm computes the **shortest paths between all pairs of nodes** in a graph with edge weights
- Works for both directed and undirected graphs
- **Time Complexity:** $O(n^3)$
  - For each new node considered as an intermediate step, an entire $n \times n$ table is updated
- **Space Complexity:** $O(n^2)$
  - All calculations are performed within the same distance table
- Based on the principle of **dynamic programming**
- It has applications in network routing and navigation systems



	A	B	C	D	E
A	0	2	10	5	7
B	3	0	13	8	10
C	7	4	0	12	14
D	12	9	5	0	X
E	10	7	3	15	0

Floyd Algorithm Overview

- There are two tables: **D** and **P**.
- **D table:** stores distances between any two nodes.
  - $D[i][i] = 0$ (distance from a node to itself).
  - If edge (i, j) exists, then $D[i][j] =$ weight of that edge, otherwise $D[i][j] = \infty$.
- **P table:** stores path reconstruction information.
  - P table is initialized with **0** on every cell (This means that there is a direct path between the two nodes)
- **Algorithm process:**
  - For each node $k = 1$ to n (considered as an intermediate node):
  - For each pair of nodes (i, j), check if going through k is shorter:

$$D(k)[i][j] = \min\{D(k-1)[i][j],\ D(k-1)[i][k] + D(k-1)[k][j]\}$$

  - Update $P[i][j]$ with the value k if there was a change in $D[i][j]$. (meaning we go through k to get from i to j)
- After all iterations:
  - D contains shortest distances.
  - P contains the information to reconstruct the shortest paths.

Graph

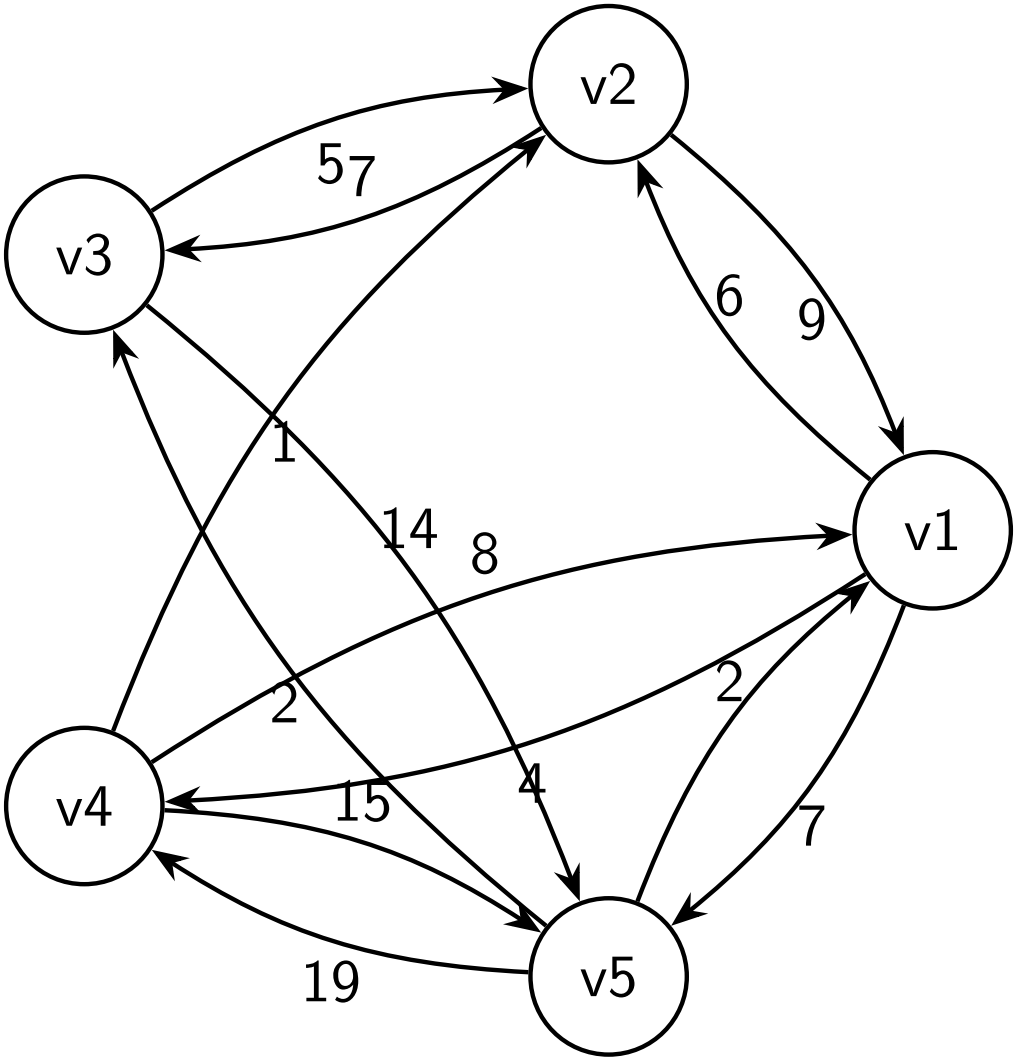


Table D(0)

	v1	v2	v3	v4	v5
v1	0	6	∞	4	7
v2	9	0	7	∞	∞
v3	∞	5	0	∞	14
v4	8	1	∞	0	15
v5	2	∞	2	19	0

Table D(1)

	v1	v2	v3	v4	v5
v1	0	6	∞	4	7
v2	9	0	7	13	16
v3	∞	5	0	∞	14
v4	8	1	∞	0	15
v5	2	8	2	6	0

Table P

	v1	v2	v3	v4	v5
v1	0	0	0	0	0
v2	0	0	0	1	1
v3	0	0	0	0	0
v4	0	0	0	0	0
v5	0	1	0	1	0

Table D(2)

	v1	v2	v3	v4	v5
v1	0	6	13	4	7
v2	9	0	7	13	16
v3	14	5	0	18	14
v4	8	1	8	0	15
v5	2	8	2	6	0

Table P

	v1	v2	v3	v4	v5
v1	0	0	2	0	0
v2	0	0	0	1	1
v3	2	0	0	2	0
v4	0	0	2	0	0
v5	0	1	0	1	0

Table D(3)

	v1	v2	v3	v4	v5
v1	0	6	13	4	7
v2	9	0	7	13	16
v3	14	5	0	18	14
v4	8	1	8	0	15
v5	2	7	2	6	0

Table P

	v1	v2	v3	v4	v5
v1	0	0	2	0	0
v2	0	0	0	1	1
v3	2	0	0	2	0
v4	0	0	2	0	0
v5	0	3	0	1	0

Table D(4)

	v1	v2	v3	v4	v5
v1	0	5	12	4	7
v2	9	0	7	13	16
v3	14	5	0	18	14
v4	8	1	8	0	15
v5	2	7	2	6	0

Table P

	v1	v2	v3	v4	v5
v1	0	4	4	0	0
v2	0	0	0	1	1
v3	2	0	0	2	0
v4	0	0	2	0	0
v5	0	3	0	1	0

Table D(5)

	v1	v2	v3	v4	v5
v1	0	5	9	4	7
v2	9	0	7	13	16
v3	14	5	0	18	14
v4	8	1	8	0	15
v5	2	7	2	6	0

Table P

	v1	v2	v3	v4	v5
v1	0	4	5	0	0
v2	0	0	0	1	1
v3	2	0	0	2	0
v4	0	0	2	0	0
v5	0	3	0	1	0

Shortest Paths from v1

- to v2 (5): v1 $\rightarrow$ v4 $\rightarrow$ v2
- to v3 (9): v1 $\rightarrow$ v5 $\rightarrow$ v3
- to v4 (4): v1 $\rightarrow$ v4
- to v5 (7): v1 $\rightarrow$ v5

Shortest Paths from v2

- to v1 (9): v2 $\rightarrow$ v1
- to v3 (7): v2 $\rightarrow$ v3
- to v4 (13): v2 $\rightarrow$ v1 $\rightarrow$ v4
- to v5 (16): v2 $\rightarrow$ v1 $\rightarrow$ v5

Shortest Paths from v3

- to v1 (14): v3 $\rightarrow$ v2 $\rightarrow$ v1
- to v2 (5): v3 $\rightarrow$ v2
- to v4 (18): v3 $\rightarrow$ v2 $\rightarrow$ v1 $\rightarrow$ v4
- to v5 (14): v3 $\rightarrow$ v5

Shortest Paths from v4

- to v1 (8): v4 $\rightarrow$ v1
- to v2 (1): v4 $\rightarrow$ v2
- to v3 (8): v4 $\rightarrow$ v2 $\rightarrow$ v3
- to v5 (15): v4 $\rightarrow$ v5

Shortest Paths from v5

- to v1 (2): v5 $\rightarrow$ v1
- to v2 (7): v5 $\rightarrow$ v3 $\rightarrow$ v2
- to v3 (2): v5 $\rightarrow$ v3
- to v4 (6): v5 $\rightarrow$ v1 $\rightarrow$ v4