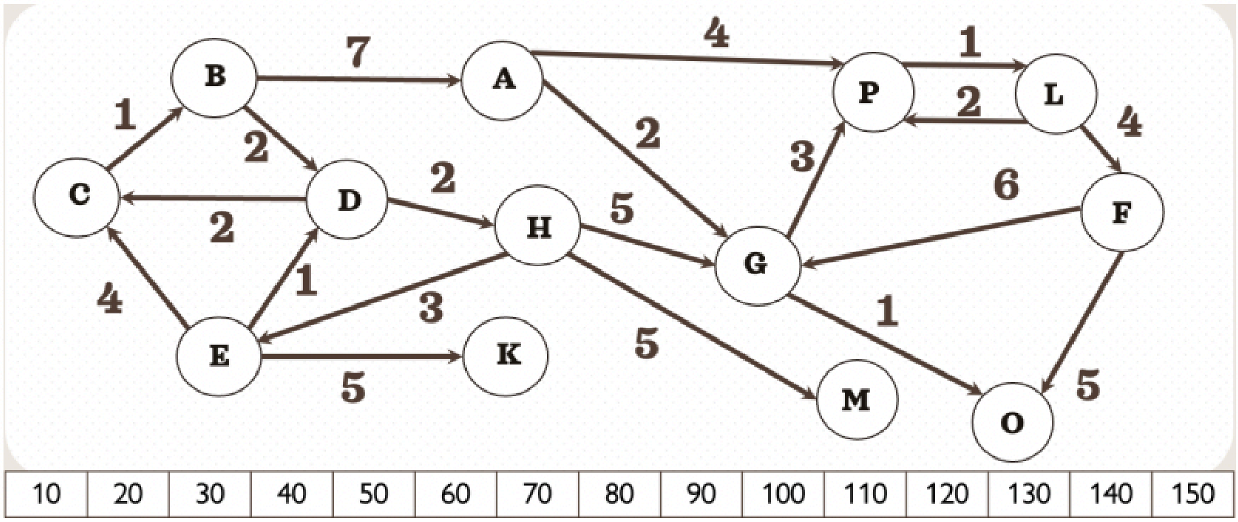
Amalia Karaman Graphs Text



1.Dijkstra from Node E:

Graph nodes: A, B, C, D, E, F, G, H, K, L, M, O, P

Start at E → E = 0, all others = ∞

- Visit E (E = 0) → D = 1, C = 4, K = 5
- Visit D (1) → B = 3, C = 3
- Visit C (3) → no update
- Visit B (3) → A = 10
- Visit K (5) → H = 8
- Visit H (8) → G = 13
- Visit A (10) → P = 14
- Visit G (13) → M = 14, P = 16 (ignored), F = 19
- Visit M (14) → O = 15
- Visit P (14) → L = 16
- Visit O (15) → F = 20 (ignored)
- Visit L (16) → F = 20
- Visit F (19) → no update

Final shortest distances from E:

E = 0 D = 1 C = 3 B = 3 K = 5 H = 8 A = 10 G = 13 P = 14 M = 14 O = 15 L = 16 F = 19

2. A*: Heuristic used: h(n) = |x(n) - x(goal)| from given coordinates

Start at E (x = 40), goal = F (x = 140)

Step-by-step:

- E (0 + 100 = 100) → D (1 + 90 = 91), C (4 + 110 = 114), K (5 + 100 = 105)
- D → C (3 + 110 = 113), B (3 + 120 = 123)
- C → no updates
- K → H (8 + 70 = 78)
- H → G (13 + 40 = 53)
- G → F (19 + 0 = 19)

Shortest path: E → K → H → G → F
- total cost = 19

3. Comparison: A* found the shortest path to F faster than Dijkstra because it skipped unrelated branches using heuristics. They shared the cost of 19 but A* reduced the number of total visited nodes.

7. Algorithm Analysis:

QuestionFour: Directed/Undirected
Time: O(n^2), where n is the number of vertices

- each cell in the adjacency matrix is checked for symmetry

Space: O(n^2), storing the full n×n matrix

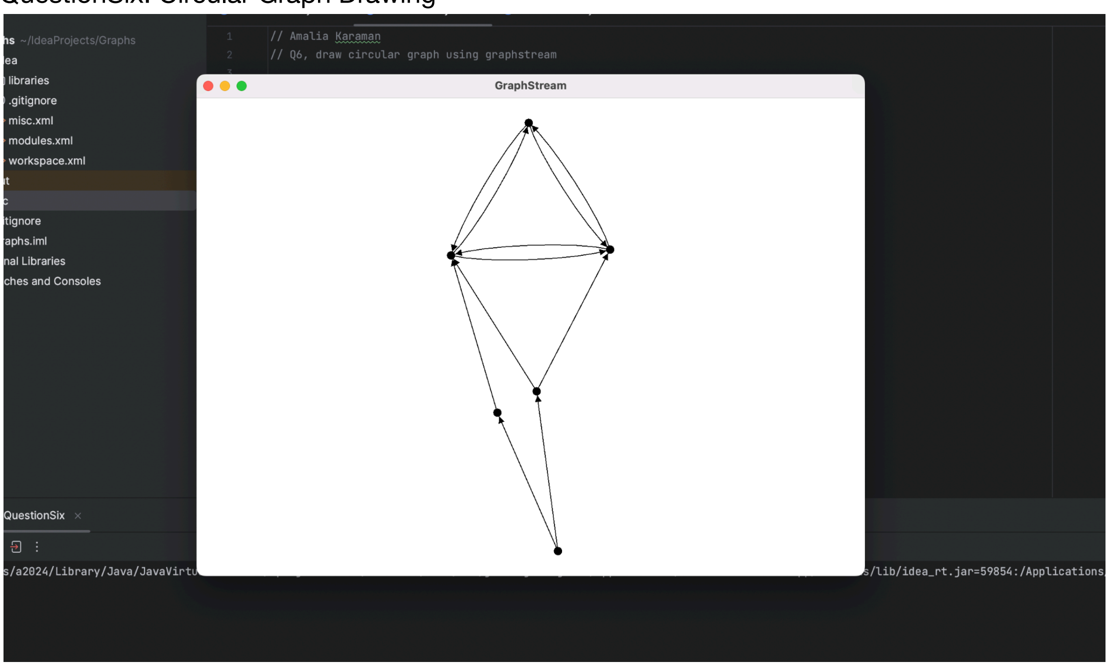
QuestionFive: Paths of Length 7
Time: O(d^7), where d is the average number of neighbors per node

- DFS explores all simple paths of 7 edges

Space: O(V + E), where V is the number of vertices and E is total edges
- adjacency list + recursion depth + visited path

Example Test Case: For a graph with edges A→B, B→C, C→D, D→E, E→F, F→G, G→H, the function should correctly print [A, B, C, D, E, F, G, H] if called with u = A and w = H.

QuestionSix: Circular Graph Drawing



Time: O(n), where n is the number of input pairs

- Parsed and created two edges per vertex

Space: O(n + e), for vertices and edges in memory, GraphStream also adds visuals