

1. **What is the adjacency matrix of the weighted graph G = (V,E) shown in Figure 1.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I |
| A | 0 | 22 | 9 | 12 | 0 | 0 | 0 | 0 | 0 |
| B | 22 | 0 | 35 | 0 | 0 | 36 | 0 | 34 | 0 |
| C | 9 | 35 | 0 | 4 | 65 | 42 | 0 | 0 | 0 |
| D | 12 | 0 | 4 | 0 | 33 | 0 | 0 | 0 | 30 |
| E | 0 | 0 | 65 | 33 | 0 | 18 | 23 | 0 | 0 |
| F | 0 | 36 | 42 | 0 | 18 | 0 | 39 | 24 | 0 |
| G | 0 | 0 | 0 | 0 | 23 | 39 | 0 | 25 | 21 |
| H | 0 | 34 | 0 | 0 | 0 | 24 | 25 | 0 | 19 |
| I | 0 | 0 | 0 | 30 | 0 | 0 | 21 | 19 | 0 |

1. **Find the shortest path from A to all other vertices using Dijkstra’s algorithm (Slide 12). (Figure 1)**

Start from A

Dis[A] = 0

Path[A] = {}

Dis[B] = dis[A] + wt(A, B) = 0 + 22 = 22

Dis[C] = dis[A] + wt(A, C) = 0 + 9 = 9 => this is the minimum

Dis[D] = dis[A] + wt(A, D) = 0 + 12 = 12

Path[C] = path[A] U {(A, C)} = {(A, C)}

Start from C

Previous

Dis[B] = dis[A] + wt(A, B) = 0 + 22 = 22

Dis[D] = dis[A] + wt(A, D) = 0 + 12 = 12 => This is the minimum

Dis[B] = dis[C] + wt(C, B) = 9 + 35 = 44

Dis[D] = dis[C] + wt(C, D) = 9 + 4 = 13

Dis[E] = dis[C] + wt(C, E) = 9 + 65 = 74

Dis[F] = dis[C] + wt(C, F) = 9 + 42 = 51

Path[D] = path[A] U {(A, D)} = {(A, D)}

Start from D

Previous

Dis[B] = dis[A] + wt(A, B) = 0 + 22 = 22 => This is the minimum

Dis[E] = dis[C] + wt(C, E) = 9 + 65 = 74

Dis[F] = dis[C] + wt(C, F) = 9 + 42 = 51

Dis[E] = dis[D] + wt(D, E) = 12 + 33 = 45

Dis[I] = dis[D] + wt(D, I) = 12 + 30 = 42

Path[B] = path[A] U {(A, B)} = {(A, B)}

Start from B

Previous

Dis[F] = dis[C] + wt(C, F) = 9 + 42 = 51

Dis[E] = dis[D] + wt(D, E) = 12 + 33 = 45

Dis[I] = dis[D] + wt(D, I) = 12 + 30 = 42 => this is the minimum

Dis[F] = dis[B] + wt(B, F) = 22 + 36 = 58

Dis[H] = dis[B] + wt(B, H) = 22 + 34 = 56

Path[I] = path[D] U {(D, I)} = {(A, D)} U {(D, I)} = {(A, D), (D, I)}

Start from I

Previous

Dis[F] = dis[C] + wt(C, F) = 9 + 42 = 51

Dis[E] = dis[D] + wt(D, E) = 12 + 33 = 45 => This is the minimum

Dis[H] = dis[B] + wt(B, H) = 22 + 34 = 56

Dis[G] = dis[I] + wt(I, G) = 42 + 21 = 63

Dis[H] = dis[I] + wt(I, H) = 42 + 19 = 61

Path[E] = path[D] U {(D, E)} = {(A, D)} U {(D, E)} = {(A, D), (D, E)}

Start from E

Previous

Dis[F] = dis[C] + wt(C, F) = 9 + 42 = 51 => this is minimum

Dis[H] = dis[B] + wt(B, H) = 22 + 34 = 56

Dis[G] = dis[I] + wt(I, G) = 42 + 21 = 63

Dis[F] = dis[E] + wt(E, F) = 45 + 18 = 63

Dis[G] = dis[E] + wt(E, G) = 45 + 23 = 68

Path[F] = path[C] U {(C, F)} = {(A,C)} U {(C, F)} = {(A, C), (C, F)}

Start from F

Previous

Dis[H] = dis[B] + wt(B, H) = 22 + 34 = 56 => This is minimum

Dis[G] = dis[I] + wt(I, G) = 42 + 21 = 63

Dis[G] = dis[F] + wt(F, G) = 51 + 39 = 90

Dis[H] = dis[F] + wt(F, H) = 51 + 24 = 75

Path[H] = Path[B] U {(B, H)} = {(A, B)} U {(B, H)} = {(A, B), (B, H)}

Start from H

Previous

Dis[G] = dis[I] + wt(I, G) = 42 + 21 = 63 => This is minimum

Dis[G] = dis[H] + wt(H, G) = 56 + 25 = 81

Path[G] = path[I] U {(I, G)} = {(A, D), (D, I)} U {(I, G)} = {(A, D), (D, I), (I, G)}

1. **What is the time complexity?**

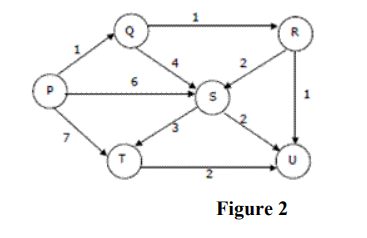
O(mlogn)

1. **Find a minimum spanning tree using Kruskal’s Algorithm (Figure 1)**

|  |  |
| --- | --- |
| Edges (Sorted Order) | Spanning Tree |
| (C,D) – 4 ✓  (A, C) – 9 ✓  (A, D) – 12 ✘  (E, F) – 18 ✓  (H, I) – 19 ✓  (G, I) – 21 ✓  (A, B) – 22 ✓  (E, G) – 23 ✓  (F, H) – 24 ✘  (H, G) – 25 ✘  (D, I) – 30 ✓  (B, H) – 34  (B, C) – 35  (B, F) – 36  (F, G) – 39  (C, F) – 42  (C, E) - 65 | A B C D E F G H I  If (cluster(C ) == cluster(D)) no  A B (C, D) E F G H I  (A, C, D) B E F G H I  (A, C, D) B (E, F) G H I  (A, C, D) B (E, F) G (H, I)  (A, C, D) B (E, F) (G, H, I)  (A, B, C, D) (E, F) (G, H, I)  (A, B, C, D) (E, F, G, H, I)  (A, B, C, D, E, F, G, H, I)   * Minimum Spanning Tree   Sum of all weights = 146 |

1. **What is the time complexity?**

O(mlogn)



1. **What is the adjacency matrix of the weighted directed Acyclic graph G = (V,E) shown in Figure 2.**
2. **Find the shortest path from P to U. (Figure 2). (Use the algorithm starting at slide 33).**
3. **What is the time complexity?**
4. **Can you use Dijkstra’s algorithm (Slide 12) to find the shortest path from P to U? (Figure 2).**
5. **If “Yes”, find the shortest path from P to U using Dijkstra’s algorithm (Slide 12) (Figure 2)**