CSCI 2270 – CS 2: Data Structures

University of Colorado
Boulder

Topics

- Dijkstra's algorithm

Dijkstra's Algorithm

- The breadth-first search algorithms found the shortest distance in an unweighted graph, where the shortest distance is the path that traverses the fewest number of edges.
- What is the shortest path to X?
- Greedy algorithm

Shortest Path

- **Weighted Edges**
  - Nonnegative real number assigned to the edges connecting to vertices
- **Weighted graphs**
  - When a graph uses the weight to represent the distance between two places
- **Weight of the path *P***
  - Given *G* as a weighted graph with vertices *u* and *v* in *G* and *P* as a path in *G* from *u* to *v*
    - ***Sum of the weights of all the edges on the path***
- **Shortest path: path with the smallest weight**
  - In a weighted graph, the path with the shortest distance between two vertices is the path with the lowest cumulative edge weight, which makes the calculation a bit more complicated than in an unweighted graph. The path with the lowest weight won't necessarily be the path that traverses the fewest number of edges.

Example 1

Example 9: Find the shortest path in the graph in Figure 25 between the Start and End vertices.

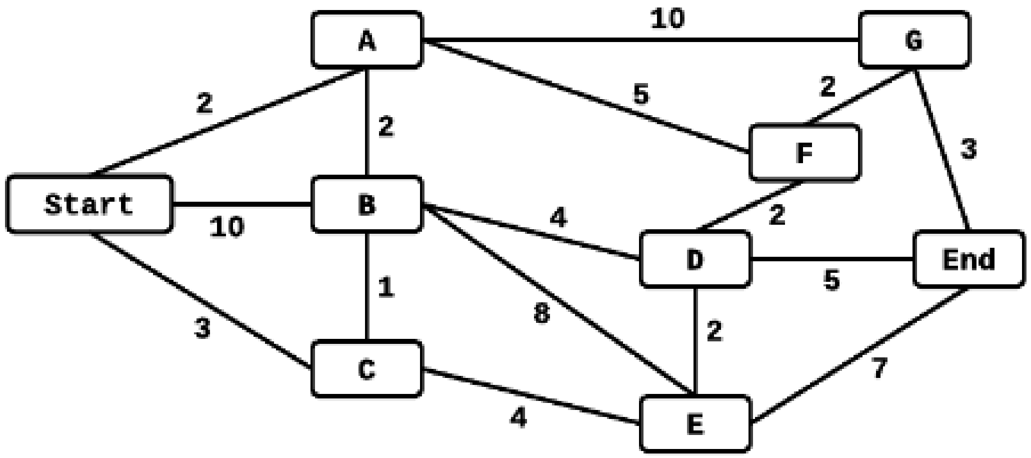


Figure 25. Find the shortest path between the Start and End vertices in this weighted graph.

Example 1 - Undirected

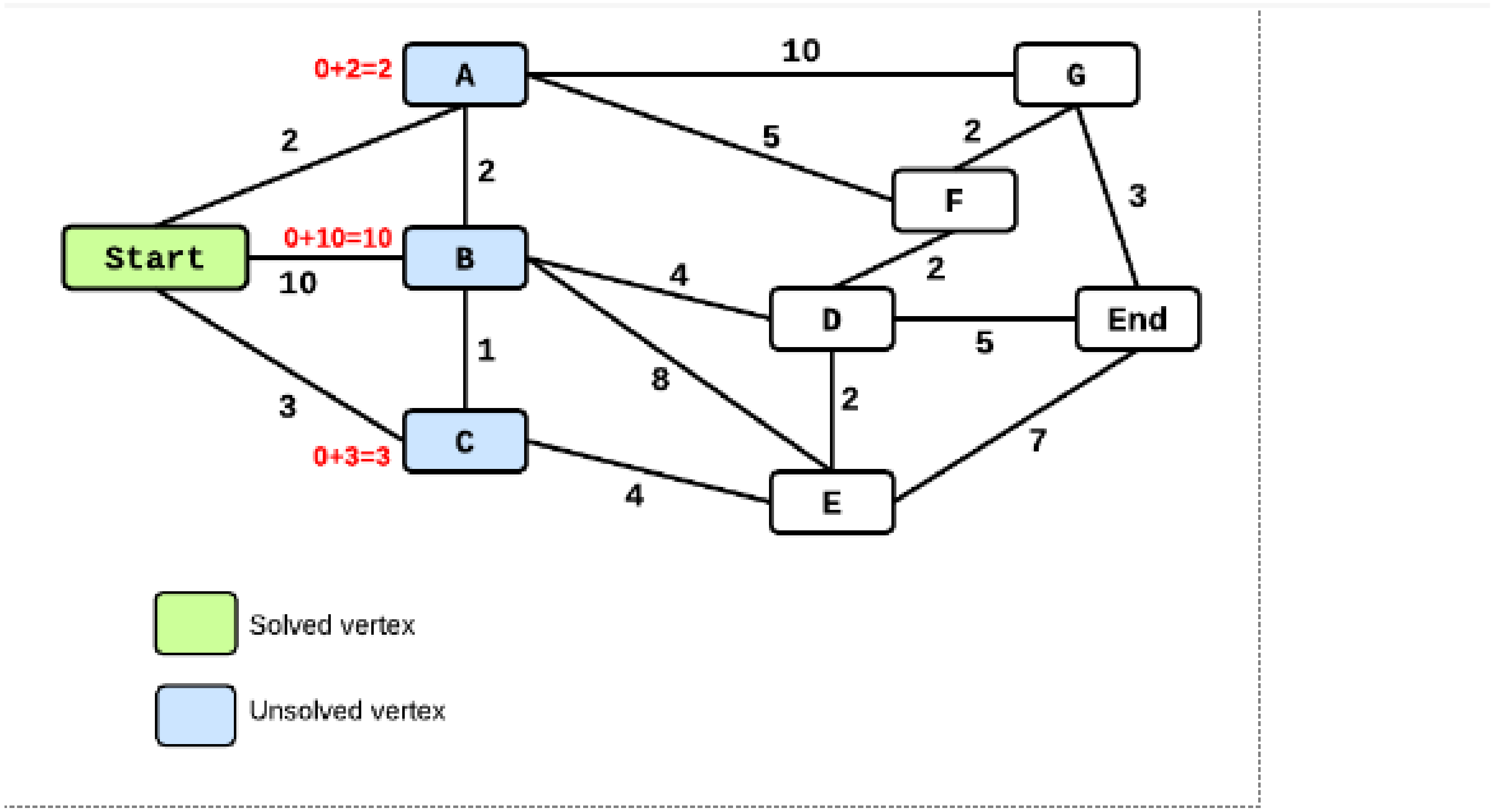
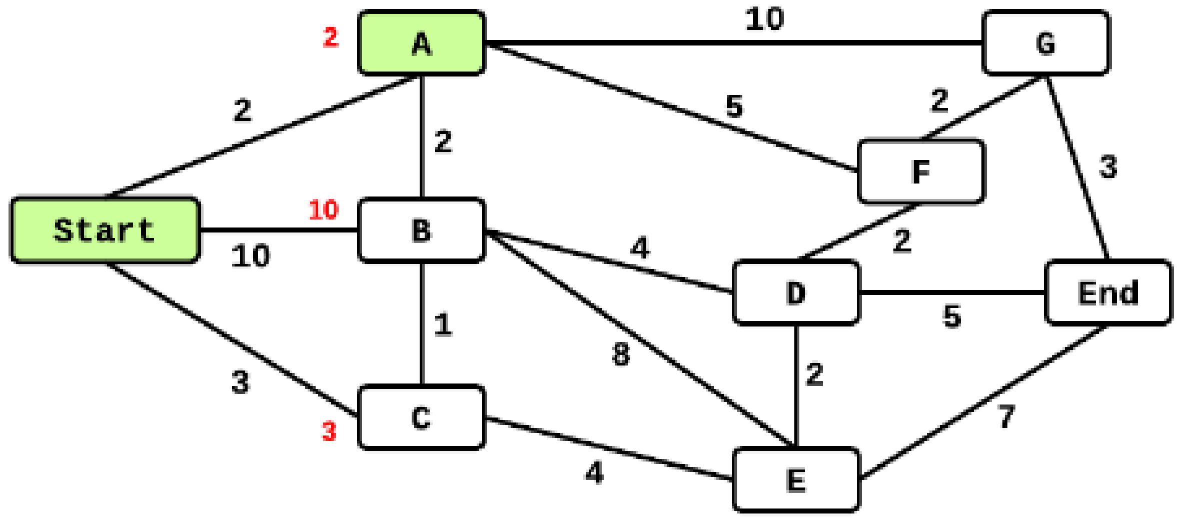


Figure 26. From the Start vertex, identify the unsolved vertices adjacent to Start and calculate the distance to each vertex. Those vertices are A, B, and C.

Example 1 - Undirected

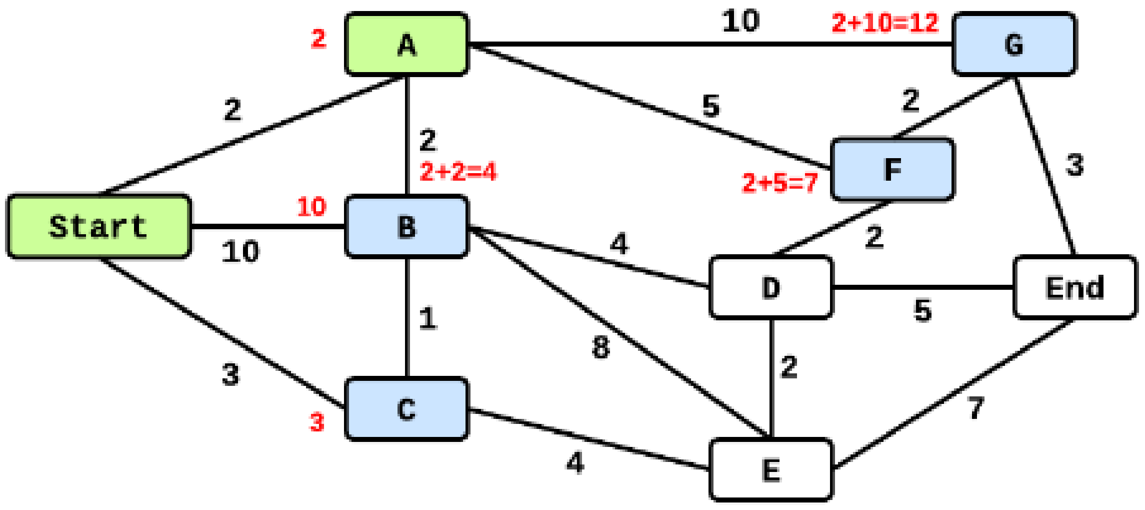


🟩 Solved vertex

🟦 Unsolved vertex

Figure 27. The vertex closest to Start is A with a distance of 2. The vertex A is now marked solved with its parent as Start and won't be solved again through another path.

Example 1 - Undirected

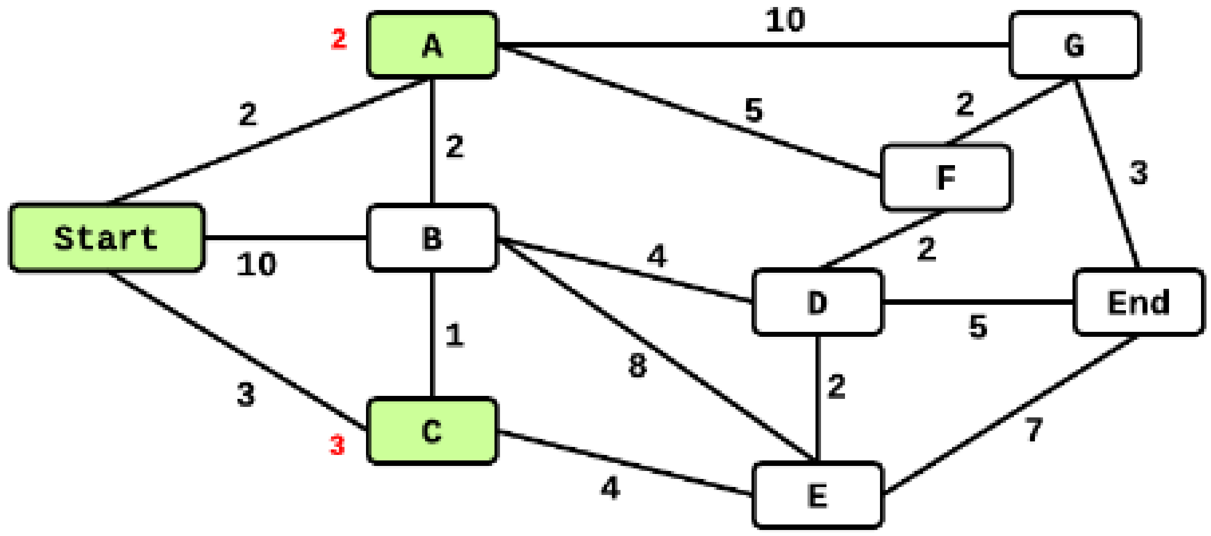


🟩 Solved vertex

🟦 Unsolved vertex

Figure 28. The vertices Start and A are both solved. The unsolved adjacent vertices to Start and A are B, C, F, and G.

Example 1 - Undirected

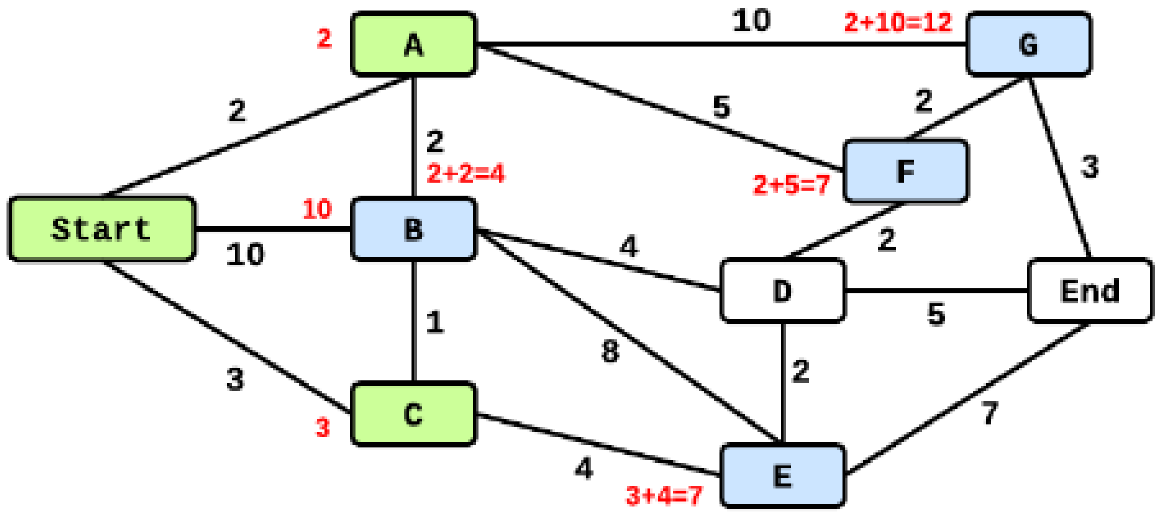


🟩 Solved vertex

🟦 Unsolved vertex

Figure 29. The vertices Start, A, and C are now solved.

Example 1 - Undirected

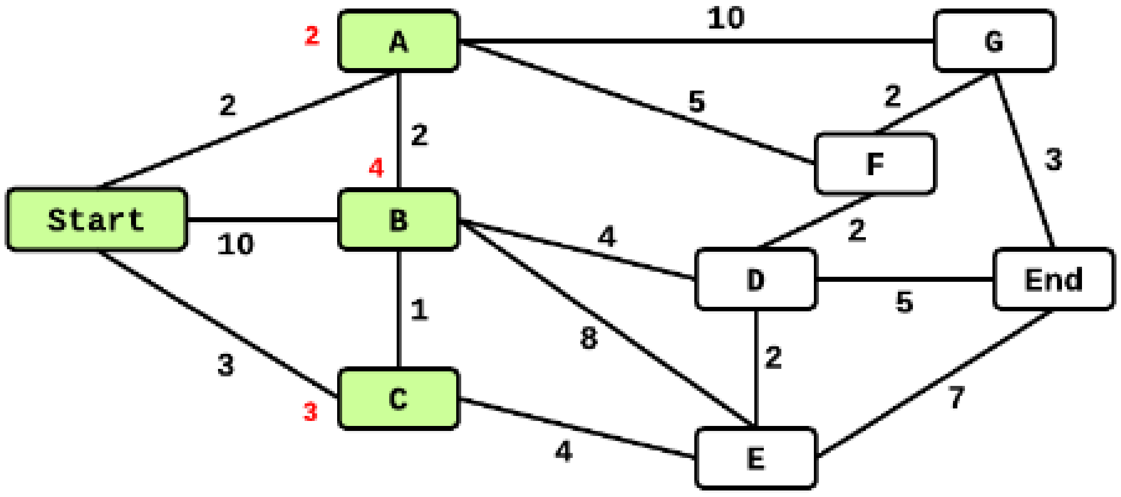


🟩 Solved vertex

🟦 Unsolved vertex

Figure 30. The solved vertices are Start, A, and C. The vertices adjacent to the solved vertices are B, E, F, and G.

Example 1



🟩 Solved vertex

🟦 Unsolved vertex

Figure 31. The vertices Start, A, B, and C are now solved. There are no unsolved vertice adjacent to the Start vertex.

Example 1 - Undirected

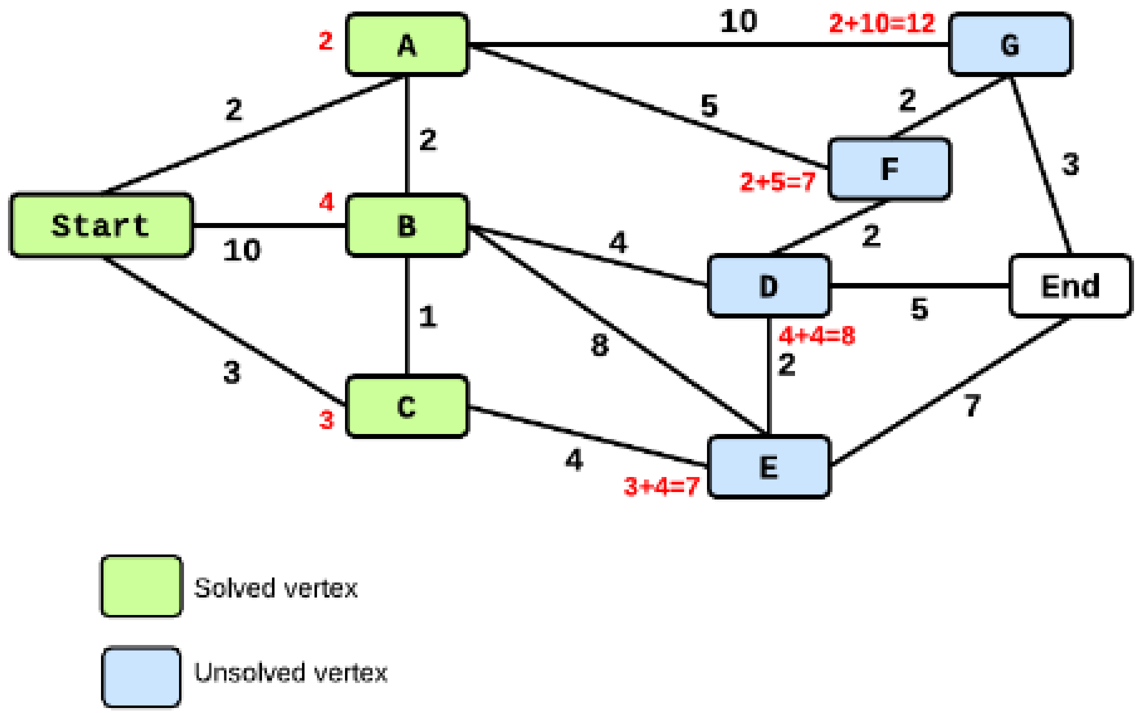


Figure 32. The solved vertices are Start, A, B, and C. The unsolved vertices adjacent to the solved vertices are D, E, F, and G.

Example 1 - Undirected

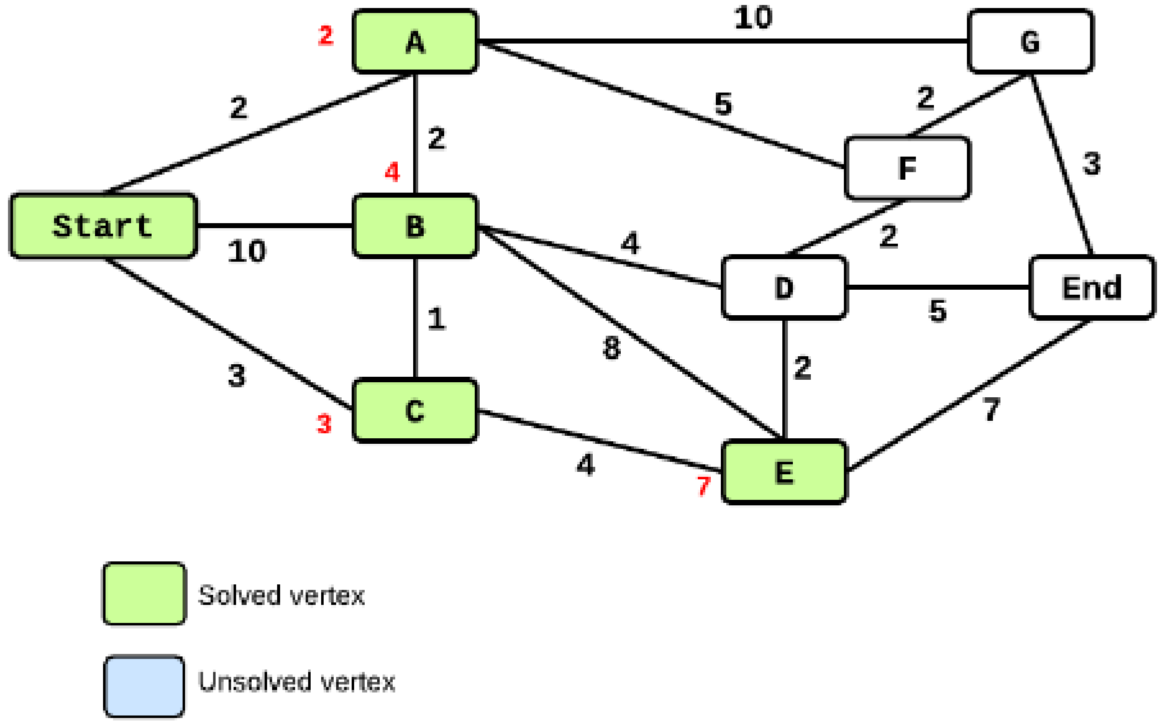
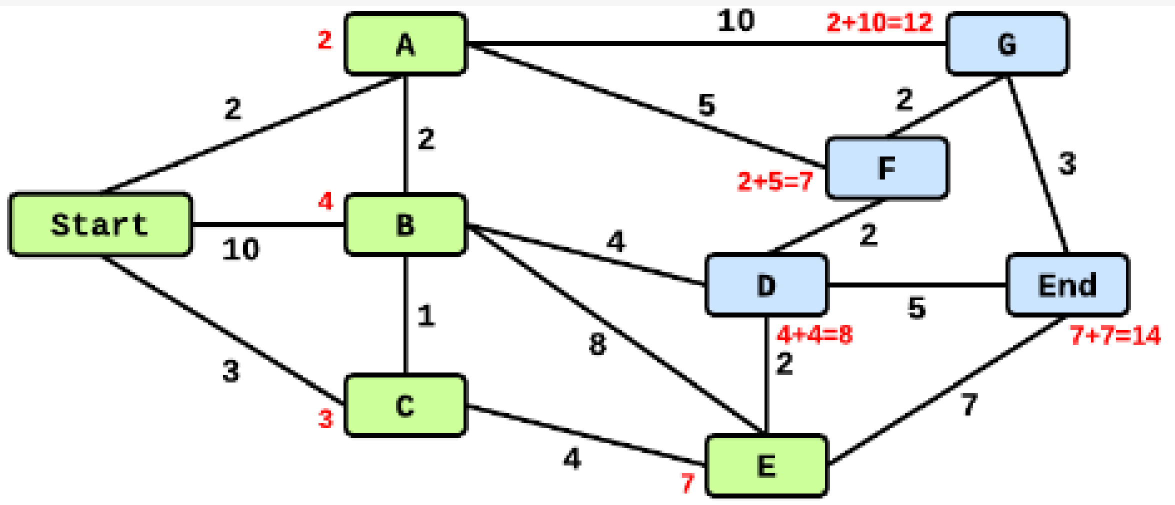


Figure 33. The vertex E is marked solved with a distance of 7. The solved vertices are Start, A, B, C, and E.

University of Colorado **Boulder**

Example 1

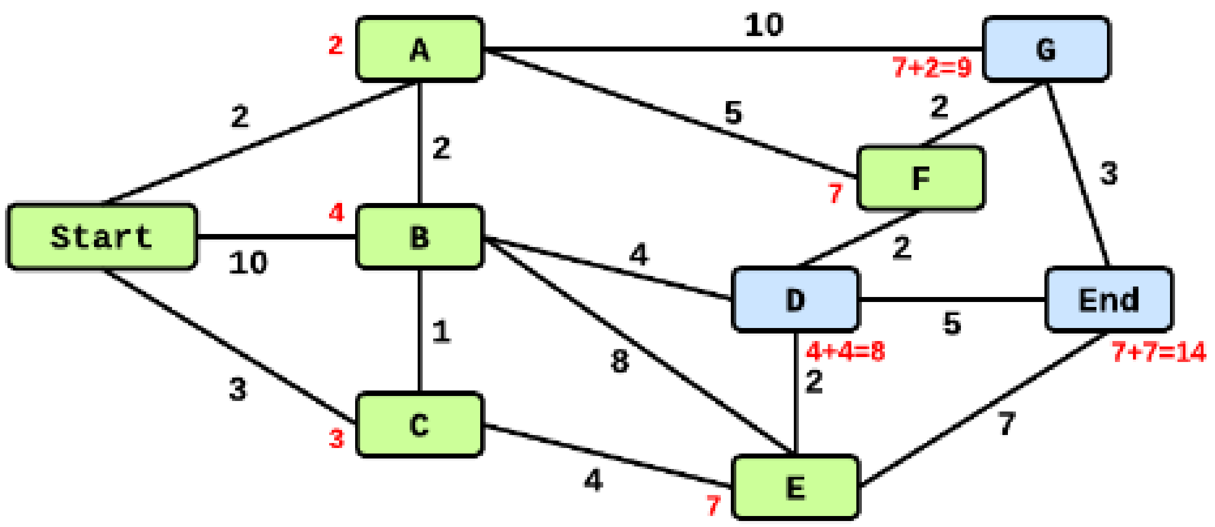


🟩 Solved vertex

🟦 Unsolved vertex

Figure 34. The solved vertices are Start, A, B, C, and E. The unsolved vertices are D, F, G, and End. The shortest distance is to F with a cost of 7.

Example 1 - Undirected

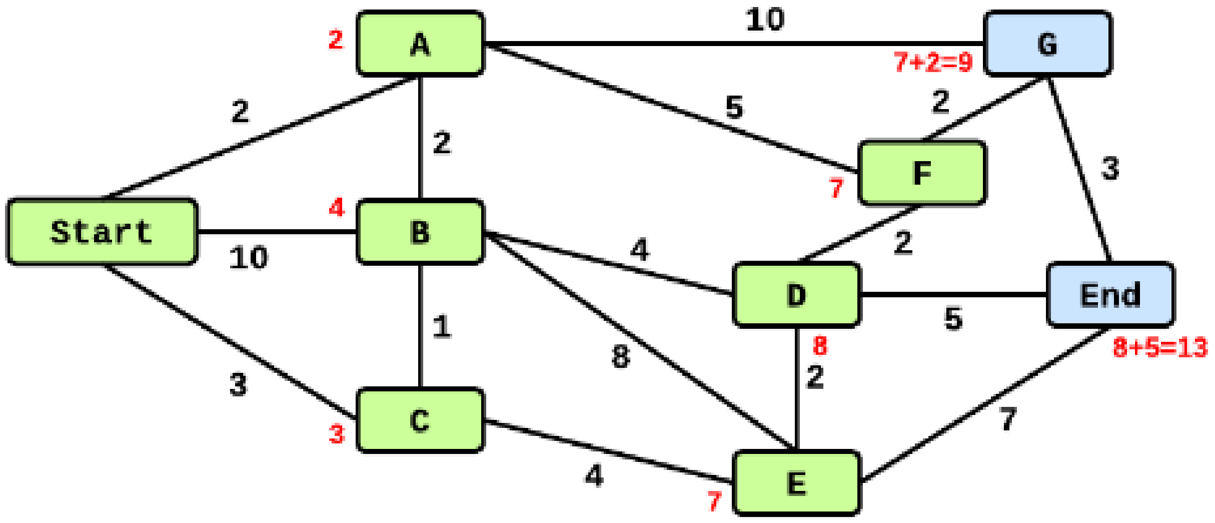


🟩 Solved vertex

🟦 Unsolved vertex

Figure 35. The vertex F is now solved. The unsolved vertices are D, G, and End with costs of 8, 9, and 14 respectively.

Example 1 - Undirected



🟩 Solved vertex

🟦 Unsolved vertex

Figure 36. The only unsolved vertices are G with a shortest distance of 9 and End with a shortest distance of 13.

Example 1

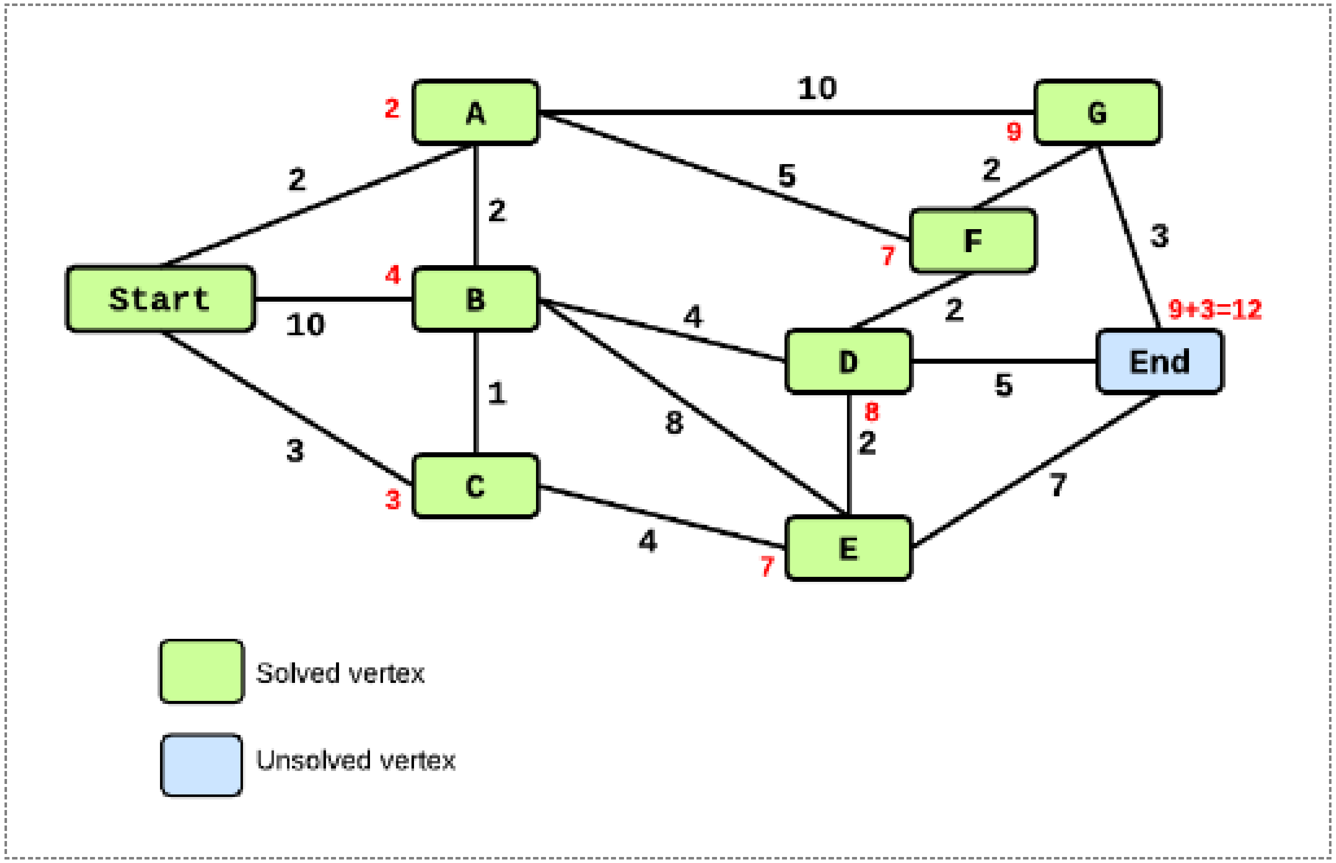
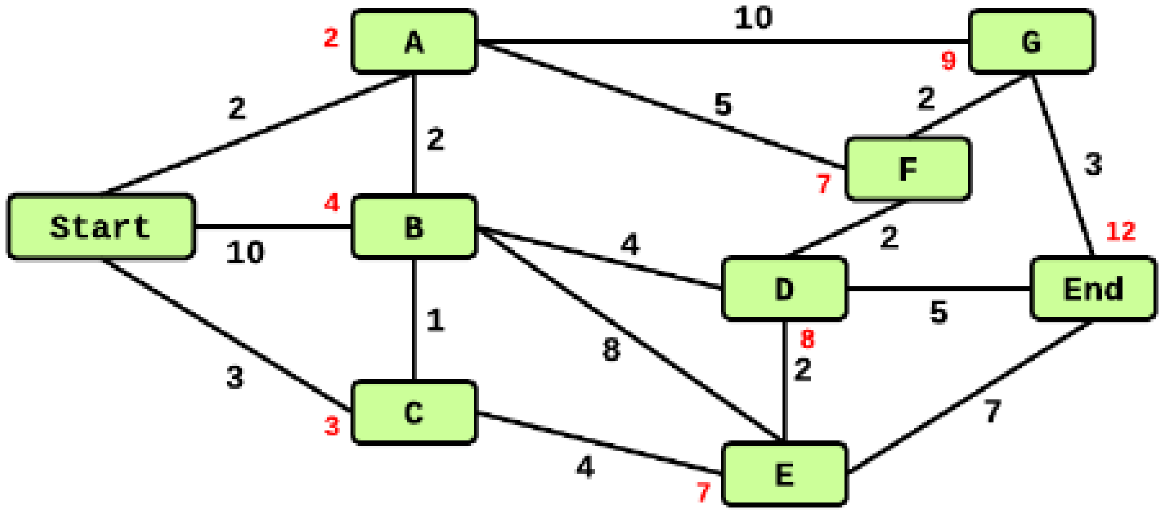


Figure 37. The vertex G is marked as solved with a distance of 9. The only unsolved vertex is End with a distance of 12.

Example 1 - Undirected



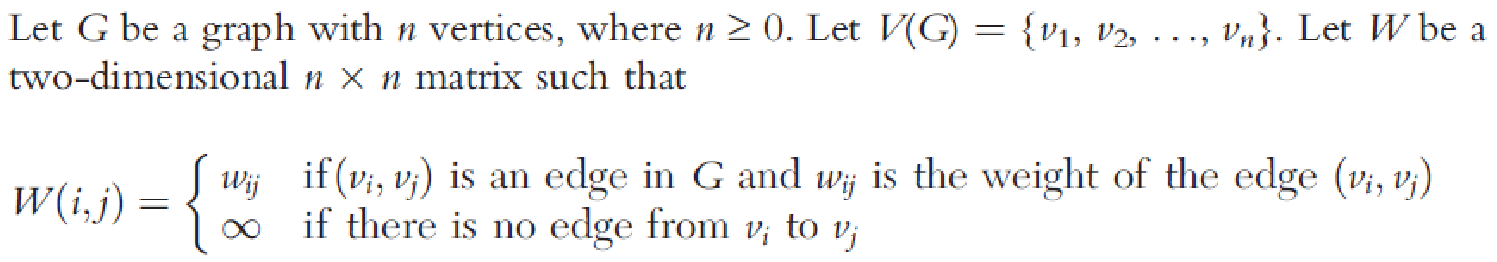
🟩 Solved vertex

🟦 Unsolved vertex

Figure 38. The shortest distance from Start to End is 12 following the path Start-A-F-G-End.

Example 2 - Directed

- Shortest path algorithm (Greedy Algorithm)



Let G be a graph with n vertices, where $n \geq 0$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. Let W be a two-dimensional $n \times n$ matrix such that

$$W(i,j) = \begin{cases} w_{ij} & \text{if } (v_i, v_j) \text{ is an edge in } G \text{ and } w_{ij} \text{ is the weight of the edge } (v_i, v_j) \\ \infty & \text{if there is no edge from } v_i \text{ to } v_j \end{cases}$$

Example 2 - Directed

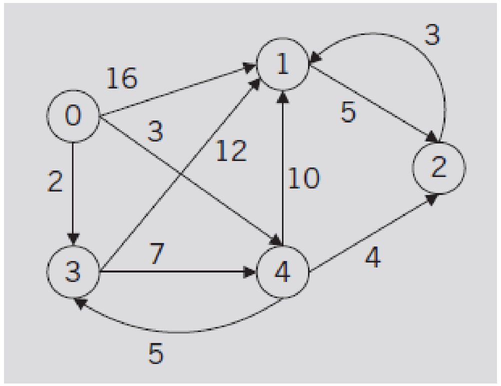
- General algorithm
  - Initialize array `smallestWeight`

    `smallestWeight[u] = weights[vertex, u]`
  - Set `smallestWeight[vertex] = zero`
  - Find vertex *v* closest to vertex where shortest path is not determined
  - Mark *v* as the (next) vertex for which the smallest weight is found

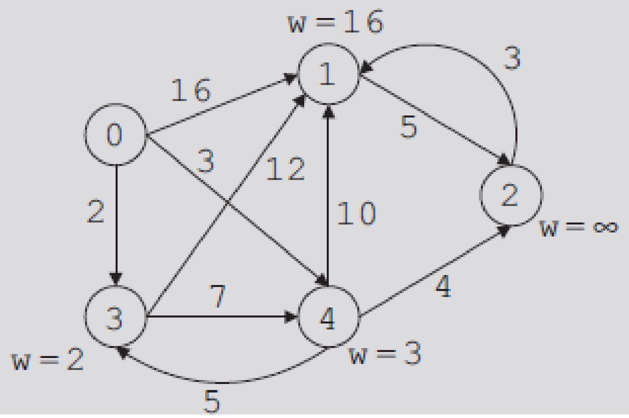
Example 2 - Directed

- General algorithm (cont'd.)
  - For each vertex *w* in *G*, such that the shortest path from vertex to w has not been determined and an edge (*v*, *w*) exists
    - ***If weight of the path to w via v smaller than its current weight***
    - ***Update weight of w to the weight of v + weight of edge (v, w)***

Example 2 - Directed



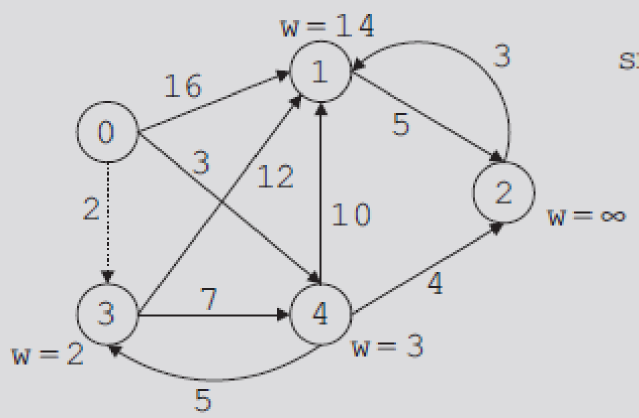
Weighted graph *G*



	[0]	[1]	[2]	[3]	[4]
smallestWeight	0	16	∞	2	3

	[0]	[1]	[2]	[3]	[4]
weightFound	T	F	F	F	F

Example 2 - Directed

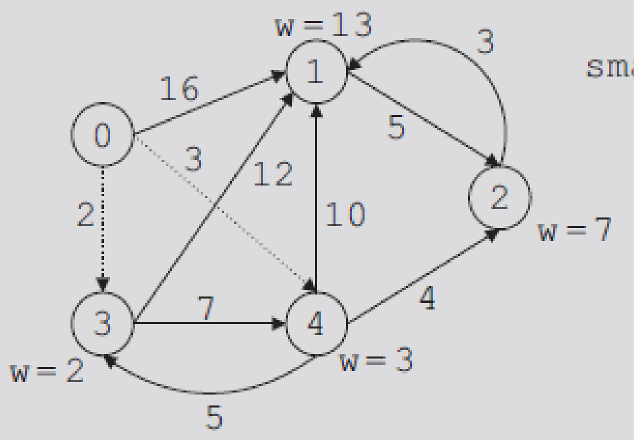


smallestWeight

[0]	[1]	[2]	[3]	[4]
0	14	∞	2	3

weightFound

[0]	[1]	[2]	[3]	[4]
T	F	F	T	F



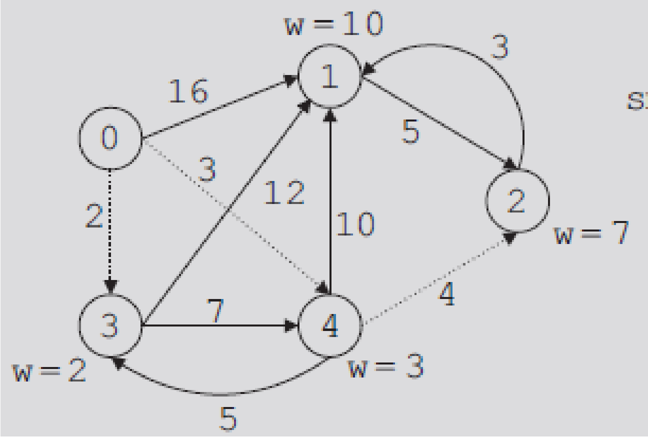
smallestWeight

[0]	[1]	[2]	[3]	[4]
0	13	7	2	3

weightFound

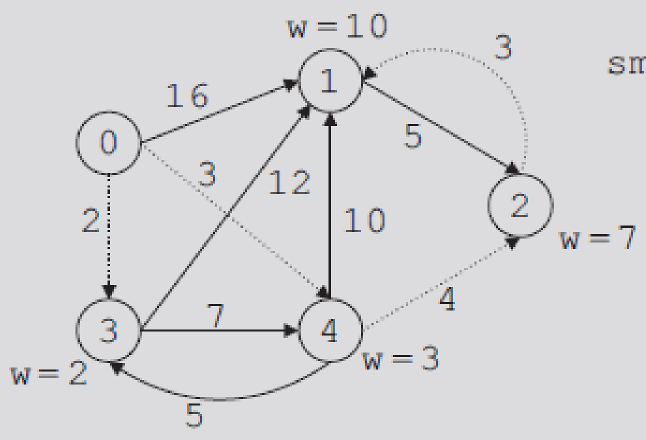
[0]	[1]	[2]	[3]	[4]
T	F	F	T	T

Example 2 - Directed



	[0]	[1]	[2]	[3]	[4]
smallestWeight	0	10	7	2	3

	[0]	[1]	[2]	[3]	[4]
weightFound	T	F	T	T	T



	[0]	[1]	[2]	[3]	[4]
smallestWeight	0	10	7	2	3

	[0]	[1]	[2]	[3]	[4]
weightFound	T	T	T	T	T

Questions?