**Depth First Search/Traversal (DFS) Algorithm:**

Set starting vertex of the graph as visited and push it onto the stack

While the stack is not empty

Peek at the top vertex on the stack [Action 1]

If there is an unvisited connection for that vertex

Mark that unvisited connection as visited and push that vertex onto the stack [Action 2A]

Else

Pop the top vertex off the stack [Action 2B]

Example 1 – use G as starting verte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action (1)** | **Action (2)** | **Stack** | **Unvisited Vertices** | **Visited Vertices** |
| Get vertex G | Push G | G | A, B, C, D, E, F | G |
| Peek at G | Push B | G, B | A, C, D, E, F | G, B |
| Peek at B | Push A | G, B, A | C, D, E, F | G, B, A |
| Peek at A | Push D | G, B, A, D | C, E, F | G, B, A, D |
| Peek at D | Push C | G, B, A, D, C | E, F | G, B, A, D, C |
| Peek at C | Push F | G, B, A, D, C, F | E | G, B, A, D, C, F |
| Peek at F | Pop F | G, B, A, D, C | E | G, B, A, D, C, F |
| Peek at C | Pop C | G, B, A, D | E | G, B, A, D, C, F |
| Peek at D | Push E | G, B, A, D, E | - | G, B, A, D, C, F, E |
| Peek at E | Pop E | G, B, A, D | - | G, B, A, D, C, F, E |
| Peek at D | Pop D | G, B, A | - | G, B, A, D, C, F, E |
| Peek at A | Pop A | G, B | - | G, B, A, D, C, F, E |
| Peek at B | Pop B | G | - | G, B, A, D, C, F, E |
| Peek at G | Pop G | Ø | - | G, B, A, D, C, F, E |

Example 2 – Use E as starting vertex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action (1)** | **Action (2)** | **Stack** | **Unvisited Vertices** | **Visited Vertices** |
| Get E | Push E | E | A, B, C, D, F , G | E |
| Peek at E | Push D | E, D | A, B, C, F,G | E, D |
| Peek at D | Push A | E, D, A | B, C, F, G | E, D, A |
| Peek at A | Push B | E, D, A, B | CFG | E, D, A, B |
| Peek at B | Push C | E, D, A, B, C | FG | E, D, A, B, C |
| Peek at C | Push F | E, D, A, B, C, F | G | E, D, A, B, C, F |
| Peek at F | Push G | E, D, A, B, C, F, G | - | E, D, A, B, C, F, G |
| Peek at G | Pop G | E, D, A, B, C, F | - | E, D, A, B, C, F, G |
| Peek at F | Pop F | E, D, A, B, C | - | E, D, A, B, C, F, G |
| Peek at C | Pop C | E, D, A, B | - | E, D, A, B, C, F, G |
| Peek at B | Pop B | E, D, A | - | E, D, A, B, C, F, G |
| Peek at A | Pop A | E, D | - | E, D, A, B, C, F, G |
| Peek at D | Pop D | E | - | E, D, A, B, C, F, G |
| Peek at E | Pop E | 0 | - | E, D, A, B, C, F, G |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Example 3 – Use A as starting vertex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action (1)** | **Action (2)** | **Stack** | **Unvisited Vertices** | **Visited Vertices** |
| Get A | Push A | A | B, C, D, E, F | A |
| Peek at A | Push B | A, B | C, D, E, F | A, B |
| Peek at B | Push E | A, B, E | C, D, F | A, B, E |
| Peek at E | Pop E | A, B | C, D, F | A, B, E |
| Peek at B | Push F | A, B, F | C, D | A, B, E, F |
| Peek at F | Push C | A, B, F, C | D | A, B, E, F, C |
| Peek at C | Pop C | A, B, F | D | A, B, E, F, C |
| Peek at F | Pop F | A, B | D | A, B, E, F, C |
| Peak at B | Pop B | A | D | A, B, E, F, C |
| Peak at A | Push D | A, D | - | A, B, E, F, C, D |
| Peek at D | Pop D | A | - | A, B, E, F, C, D |
| Peek at A | Pop A | 0 | - | A, B, E, F, C, D |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Breadth First Search/Traversal (DFS) Algorithm:**

Set starting vertex as visited and enqueue it into the queue

While the queue is not empty

Dequeue the next vertex

While there is an unvisited connected vertex of the dequeued vertex

Mark unvisited vertex as visited and enqueue it into the queue

Example 1 – use F as starting vertex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dequeue** | **Enqueue** | **Queue** | **Unvisited Vertices** | **Visited Vertices** |
| - | F | F | A, B, C, D, E, G | F |
| F | C, E, G | C, E, G | A, B, D | F, C, E, G |
| C | B, D | E, G, B, D | A | F, C, E, G, B, D |
| E | - | G, B, D | A | F, C, E, G, B, D |
| G | - | B, D | A | F, C, E, G, B, D |
| B | A | D, A | Ø | F, C, E, G, B, D, A |
| D | - | A | Ø | F, C, E, G, B, D, A |
| A | - | Ø | Ø | F, C, E, G, B, D, A |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Example 2 – Use D as starting vertex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dequeue** | **Enqueue** | **Queue** | **Unvisited Vertices** | **Visited Vertices** |
| - | D | D | A, B, C, E, F, G | D |
| D | A, C, E | A, C, E | B, F, G | D, A, C, E |
| A | B | C, E, B | F, G | D, A, C, E, B |
| C | F, G | E, B, F, G | - | D, A, C, E, B, F, G |
| E | - | B, F, G | - | D, A, C, E, B, F, G |
| B | - | F, G | - | D, A, C, E, B, F, G |
| F | - | G | - | D, A, C, E, B, F, G |
| G | - | 0 | - | D, A, C, E, B, F, G |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Example 3 – Use F as starting vertex:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dequeue** | **Enqueue** | **Queue** | **Unvisited Vertices** | **Visited Vertices** |
| - | F | F | A, B, C, D, E | F |
| F | B, C | B, C | A, D, E | F, B, C |
| B | A, E | C, A, E | D | F, B, C, A, E |
| C | - | A, E | D | F, B, C, A, E |
| A | D | E, D | - | F, B, C, A, E, D |
| E | - | D | - | F, B, C, A, E, D |
| D | - | 0 | - | F, B, C, A, E, D |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Dijkstra’s Shortest Path Algorithm:**

While some discovered vertices are still not in the completed set

Select a vertex based on cheapest cost

Discover connecting vertices

Evaluate vertices not in completed set based on cost

If the cost is less than previous cost

Update path and cost

Use path/cost data to find optimal path from vertex to vertex

Example 1 – A is the initial vertex:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selected Vertex |  | Destination Vertex | | | | | | |
|  | A | B | C | D | E | F | G |
| Initial | **0-A** | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| A (start vertex) |  | 0 + 3 = 3  **3-A** | ∞ | 0 + 4 = 4  4-A | ∞ | ∞ | ∞ |
| B |  |  | 3 + 7 = 10  10-B | **4-A** | ∞ | ∞ | 3 + 4 = 7  7-B |
| D |  |  | 4 + 5 = 9  9-D |  | 4 + 1 = 5  **5-D** | ∞ | 7-B |
| E |  |  | 9-D |  |  | 5 + 2 = 7  **7-E** | 7-B |
| F |  |  | 7 + 1 = 8  8-F |  |  |  | ~~7 + 5 = 12~~  **7-B** |
| G |  |  | **8-F** |  |  |  |  |
| C |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Shortest Path From A | B | C | D | E | F | G |
| Cost | 3 | 8 | 4 | 5 | 7 | 7 |
| Through vertex | A | F | A | D | E | B |
| Path | A-B | A-D-E-F-C | A-D | A-D-E | A-D-E-F | A-B-G |

Example 2 – A is the initial vertex:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selected Vertex |  | Destination Vertex | | | | | | |
|  | A | B | C | D | E | F | G |
| Initial | **0** | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| A (start vertex) |  | 0 + 7 =7  **7-A** | inf | 0+8 =8  8-A | inf | inf | inf |
| B |  |  | 7 + 9=16  16-B | **8-A** | inf | inf | 7+1 =8  8-B |
| D |  |  | 8+3=11  11-D |  | 8+6=14  14-D | inf | **8-B** |
| G |  |  | 11-D |  | 14-D | 8+2=10  **10-G** |  |
| F |  |  | **11-D** |  | 10+2= 12  12-F |  |  |
| C |  |  |  |  | **12-F** |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Shortest Path From A | B | C | D | E | F | G |
| Cost | 7 | 11 | 8 | 12 | 10 | 8 |
| Through Vertex | A | D | A | F | G | B |
| Path | A-B | A-D-C | A-D | A-B-G-F-E | A-B-G-F | A-B-G |

Example 3 – G is the initial vertex:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selected Vertex |  | Destination Vertex | | | | | | |
|  | A | B | C | D | E | F | G |
| Initial | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | **G-0** |
| G (start vertex) |  | 0+3=3  **3-G** | inf | inf | inf | 0+8=8  8-G |  |
| B | 3+9=12  12-B |  | 3+1=4  **4-B** | inf | inf | 8-G |  |
| C | 12-B |  |  | 4+5=9  9-C | inf | 4+4=8  **8-C** |  |
| F | 12-B |  |  | **9-C** | 8+9=17  17-F |  |  |
| D | **12-B** |  |  |  | 17-F |  |  |
| A |  |  |  |  | **17-F** |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Shortest Path From G | A | B | C | D | E | F |
| Cost | 12 | 3 | 4 | 9 | 17 | 8 |
| Through Vertex | B | G | B | C | F | C |
| Path | G-B-A | G-B | G-B-C | G-B-C-D | G-F-E | G-F |

Prim’s Minimum Spanning Tree (mathematician Vojtěch Jarník discovered algorithm a generation earlier)

High level algorithm:

Pick a vertex to go into a discovered set of vertices

While there are vertices that are undiscovered

Find the cheapest available edge that discovers a new vertex

Place that vertex into the discovered set

Place that edge into the minimum spanning tree

(When all the vertices have been discovered, a minimum spanning tree is completed.)

The algorithm can be accomplished easily with a sorted list of edges.

Example 1 – Start at vertex E:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Sorted Edges | T1 | T2 | T3 | T4 | T5 | T6 | | Tree: E0, F1, C2, G3, B4, D5, A6 | | | | | | | 1BG | - | - | - | Add | // | // | | 1CF | - | Add | // |  | // | // | | 2EF | Add |  | // |  | // | // | | 2FG |  |  | Add |  | // | // | | 3CD |  |  |  |  | ADD | // | | 6DE |  |  |  |  |  | // | | 7AB |  |  |  |  |  | Add | | 8AD |  |  |  |  |  |  | | 9BC |  |  |  |  |  |  | |  |

Minimum Spanning Tree: 3CD, 1CF, 2EF, 2FG, 1BG, 7AB

Sum: 16

Prim’s Algorithm - Example 2 (trace execution, sum, and draw):

Pick A:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Sorted Edges | T1 | T2 | T3 | T4 | T5 | T6 | | Tree: A,B,D,E,F,C,G | | | | | | | 1FC | - | - | - | - | Add | // | | 1DE | - | - | Add | // |  | // | | 2EF | - | - |  | Add |  | // | | 3BA | Add | // |  |  |  | // | | 4BG |  | // |  |  |  | Add | | 4AD |  | Add |  |  |  |  | | 5CD |  |  |  |  |  |  | | 5GF |  |  |  |  |  |  | | 7CB |  |  |  |  |  |  | |
| Minimum spanning tree: 3BA, 4AD, 1DE, 2EF, 1FC, 4BG  Sum: 3+4+4+2+1+1 = 15 | |

Prim’s Algorithm - Example 3 (trace execution, sum, and draw):

Pick A ; GF = FG = 8

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Sorted Edges | T1 | T2 | T3 | T4 | T5 | T6 | | Tree: A, D, C, B, G, F, E | | | | | | | 1CB | - | - | Add | // | // | // | | 3BG | - | - |  | Add | // | // | | 4CF | - | - |  |  | Add | // | | 5DC | - | Add |  |  |  | // | | 6AD | Add |  |  |  |  | // | | 8DE |  |  |  |  |  | Add | | 8FG |  |  |  |  |  |  | | 9EF |  |  |  |  |  |  | | 9AB |  |  |  |  |  |  | |
| Minimum spanning tree: 6AD, 5DC, 1CB, 3BG, 4CF, 8AD  Sum: 6+5+1+3+4+8 = 27 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Sorted Edges | Same  Set? | Action | Result | |  |  |  | { A }, { B }, { C }, { D }, { E }, { F },  { G } | | 1BG | No | Add and  Merge | { A }, { B, G }, { C }, { D }, { E }, { F } | | 1CE | No | Add and  Merge | { A }, { B, G }, { C, E }, { D }, { F } | | 1CF | No | Add and  Merge | { A }, { B, G }, { C, E, F }, { D } | | 1EF | Yes | Skip | { A }, { B, G }, { C, E, F }, { D } | | 2AB | No | Add and  Merge | { A, B, G }, { C, E, F }, { D } | | 2DE | No | Add and  Merge | { A, B, G }, { C, D, E, F } | | 2FG | No | Add and  Merge | { A, B, C, D, E, F, G } | | 4BC |  |  |  | | … |  |  |  | | **Start with all vertices as individual disjoint sets. Go through sorted edges, merging disjoint sets whenever a new edge to the tree is found until all the vertices are joined.**  Sum: 9 |
| NOTE: A&M = Add and Merge |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kruskal’s Algorithm Example 2 Graph: | |  |  |  |  | | --- | --- | --- | --- | | Sorted Edges | Same  Set? | Action | Result | |  |  |  | { A }, { B }, { C }, { D },  { E }, { F }, { G } | | 1DC | No | A&M | { A }, { B }, { C, D }, { E }, { F }, { G } | | 2AB | No | A&M | { A, B }, { C, D }, { E }, { F }, { G } | | 2AC | No | A&M | { A, B, C, D }, { E }, { F }, { G } | | 3AD | Yes | Skip | { A, B, C, D }, { E }, { F }, { G } | | 4FG | No | A&M | { A, B, C, D }, { E }, { F, G } | | 4CB | Yes | Skip | { A, B, C, D }, { E }, { F, G } | | 5BG | No | A&M | { A, B, C, D, F, G }, { E } | | 6CF | Yes | Skip | { A, B, C, D, F, G }, { E } | | 7CE | No | A&M | { A, B, C, D, E, F, G } |   Sum: 1+2+2+5+4+7 = 21 |
| Example 2 Minimum Spanning Tree | Example 3 Minimum Spanning Tree |
| Kruskal’s Algorithm Example 3 Graph | |  |  |  |  | | --- | --- | --- | --- | | Sorted Edges | Same  Set? | Action | Result | |  |  |  | { A }, { B }, { C }, { D },  { E }, { F }, { G } | | 1BG | No | A&M | { A }, { B, G }, { C }, { D }, { E }, { F } | | 1BC | No | A&M | { A }, { B, C, G }, { D }, { E }, { F } | | 2AB | No | A&M | { A, B, C, G }, { D }, { E }, { F } | | 2AC | Yes | Skip | { A, B, C, G }, { D }, { E }, { F } | | 2GC | Yes | Skip | { A, B, C, G }, { D }, { E }, { F } | | 2CF | No | A&M | { A, B, C, F, G }, { D }, { E } | | 3GF | Yes | Skip | { A, B, C, F, G }, { D }, { E } | | 5AD | No | A&M | { A, B, C, D, F, G }, { E } | | 7CE | No | A&M | { A, B, C, D, E, F, G } |   Sum: 1+1+2+2+5+7 = 18 |

Some videos that might be enlightening:

Dijkstra’s Shortest Path Algorithm:

<https://www.youtube.com/watch?v=0nVYi3o161A>

Prim’s Minimum Spanning Tree Algorithm (for graphs)

<https://www.youtube.com/watch?v=xtOljl0muG4>

Prim’s Algorithm with heap

<https://www.youtube.com/watch?v=EqiyCv7bqzY>

Kruskal’s Minimum Spanning Tree Algorithm

<https://www.youtube.com/watch?v=5xosHRdxqHA> (our focus is on the theory rather than code now)

( Prim: <https://www.youtube.com/watch?v=z1L3rMzG1_A> )

<https://www.youtube.com/watch?v=NWPIJRW2cQo>

Depth-First Search and Breadth-First Search algorithms:

<https://www.teachengineering.org/collection/uno_/lessons/uno_connection/uno_connection_lesson01_instructions_v2_tedl_dwc.pdf>

Dijkstra’s Shortest Path Algorithm:

<https://www.youtube.com/watch?v=0nVYi3o161A>

Prim’s Minimum Spanning Tree Algorithm (for graphs)

<https://www.youtube.com/watch?v=xtOljl0muG4>

Kruskal’s Minimum Spanning Tree Algorithm

<https://www.youtube.com/watch?v=5xosHRdxqHA> (our focus is on the theory rather than code now)