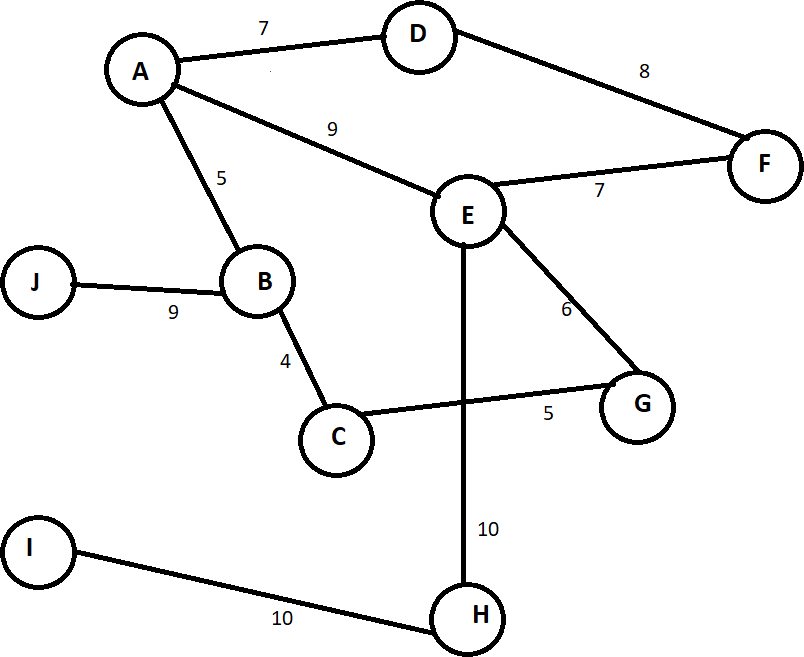
**CSE 214**

**Recitation 10 (Graph) Student Version**

1. **[15 minutes] Graph Representations Graph G:**



1. **Write the adjacency matrix representation of the weights of the edges of Graph G, mark**

-1 for no connecting edge.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Graph G** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** |
| **A** | - |  |  |  |  |  |  |  |  |  |
| **B** | 5 | - |  |  |  |  |  |  |  |  |
| **C** |  | 4 | - |  |  |  |  |  |  |  |
| **D** | 7 | - | - | - |  |  |  |  |  |  |
| **E** | 9 | - | - | - | - |  |  |  |  |  |
| **F** | - | - | - | 8 | 7 | - |  |  |  |  |
| **G** | - | - | 5 | - | 6 | - | - |  |  |  |
| **H** | - | - | - | - | 10 | - | - | - | 10 |  |
| **I** | - | - | - | - | - | - | - | 10 | - |  |
| **J** | - | 9 | - | - | - | - | - | - | - | - |

1. **Construct the edge list for Graph G.**

**A: (B, 5)****(D, 7)****(E, 9) B: (A, 5)****(C, 4)****(J, 9)**

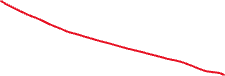
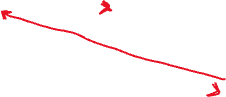
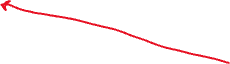
**C: (B, 4)****(G, 5)** **D: (A, 7)****(F, 8)**

**E: (A, 9)****(F, 7)****(G, 6)****(H, 10)** **F: (D, 8)****(E, 7)**

**G: (C, 5)****(E, 6)** **H: (E, 10)****(I, 10)**

**I: (H, 10)** **J: (B, 9)**

1. **Given the following adjacency matrix, draw the corresponding unweighted directed graph.**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Graph C** | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **2** | **0** | **0** | **0** | **1** | **0** | **0** |
| **3** | **1** | **0** | **0** | **0** | **0** | **0** |
| **4** | **0** | **0** | **0** | **0** | **1** | **1** |
| **5** | **0** | **0** | **0** | **0** | **0** | **0** |
| **6** | **1** | **0** | **0** | **0** | **1** | **0** |

1. **[5 minutes] True/False**
2. **Given a graph with n vertices and m edges, the space complexity of an adjacency matrix is** 𝑂(𝑛𝑚).

False, an adjacency matrix has n rows and n columns, so space complexity would be O(n^2)

1. **For an undirected graph, the adjacency matrix is asymmetrical.**

False, if directed then asymmetrical.

False, since connections are undirected, if A is connected to B, then B must be connected to A, and thus the adjacency matrix will be symmetrical.

1. **It is faster to add or remove an edge on an adjacency matrix compared to an edge list.**

True, we can check in O(1) time if the edge exists using the adjacency matrix.

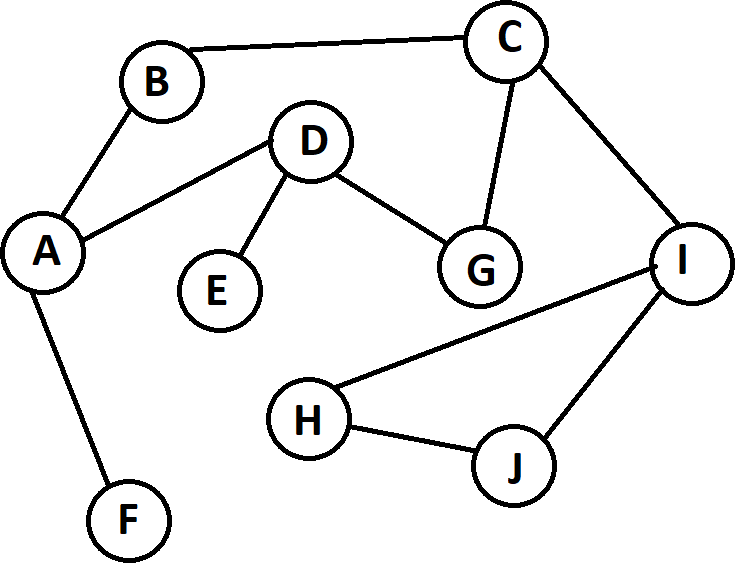
1. **When representing a graph, an adjacency matrix is always more memory efficient than edge lists.**

**False, an edge list is the more memory efficient representation when the adjacency matrix is sparse.**

1. **Dijkstra’s shortest-path algorithm finds the path with the least number of edges from a source to every other vertex of a graph.**

**False, the algorithm finds the path of minimum total weight from a source node to every other node in the graph.**

1. **[15 minutes] Graph Traversal Graph T:**



1. **Perform breadth-first traversal on Graph T, begin with vertex A and vertex C.**

Begin with A: {A, B, D, F, C, E, G, I, H, J}

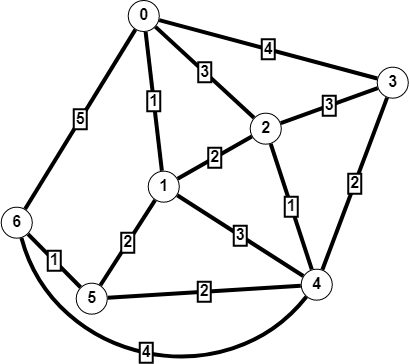
Begin with C: {C, B, G, I, A, D, H, J, F, E}

1. **Perform depth-first traversal on Graph T, begin with vertex D and vertex I.**

**Begin with D: {D, A, B, C, G, I, H, J, F, E}**

**Begin with I: {I, C, B, A, D, E, G, F, H, J}**

1. **[10 minutes] Dijkstra’s Shortest Path Algorithm given Graph D:**



1. **Simulate Dijkstra’s shortest path algorithm on Graph D with source vertex 6.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S** | **V-S** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
|  | {0,1,2,3,4,5,6} |  |  |  |  |  |  | 0 |
| {6} | {0,1,2,3,4,5} | 5 |  |  |  | 4 | 1 | - |
| {5,6} | {0,1,2,3,4} | 5 | 3 |  |  | 3 | - |  |
| {5,6,1} | {0,2,3,4} | 4 | - | 5 |  | 3 |  |  |
| {5,6,1,4} | {0,2,3} | 4 |  | 4 | 5 | - |  |  |
| {5,6,1,4,0} | {2,3} | - |  | 4 | 5 |  |  |  |
| {5,6,1,4,0,2} | {3} |  |  | - | 5 |  |  |  |

1. **What is the shortest path from vertex 6 to vertex 3, and what is the total weight of that path?**

Path is {6, 5, 4, 3} and total weight is 1+2+2 = 5

1. **[10 minutes] Graph Traversal Application**

Design an algorithm that returns true if there exists a path between two vertices in the graph.

**boolean isReachable(Vertex src, Vertex dst){**

**// use breadth-first traversal to find a path from src to dst.**

**Queue<Vertex> bftQueue = new Queue<Vertex>();**

**src.visited = true;**

**bftQueue.enqueue(src);**

**Vertex curr;**

**while(!bftQueue.isEmpty()){**

**curr = bftQueue.dequeue();**

**for(Vertex v : curr.getNeighbors()){**

**if(v.equals(dst)) return true;**

**if(!v.visited) {**

**v.visited = true;**

**bftQueue.enqueue(v);**

**}**

**}**

**}**

**return false;**

**}**