1. Given a Vertex object x, give Java code to enumerate its outgoing edges.

- Since the Vertex class has an edgesFrom function which returns outgoing edges, we can iterate each nodes using for loop. The code shown below

```
// x is an instance of the Vertex object
for(Edge currE : x.edgesFrom()) {
       System.out.println(currE);
       // currE is an Edge from Vertex x
}
```

2. Given a VertexAndDist object x and a new distance d, give Java code to create an updated object with the same vertex as x but with distance d.

- Get a current Vertex from the object x and create another VertextAndDist instance with the Vertex and the new distance d.

```
// x is an instance of VertexAndDist
// VertexAndDist has a vertex and a distance in its class
// Since they are public, we can access them directly.
Vertex currV = x.vertex;
new VertexAndDist(currV, d);
```

3. For simplicity of implementation, we use HashMaps to map vertices to handles/parents and edges to weights. How could you modify the Vertex and/or Edge classes, as well as the maps themselves, to implement the maps using ordinary arrays, with no hashing? Be sure to address both the vertex and edge maps in your answer. (Hint: consider the Vertex's "id" field for inspiration.)

- How could you modify the Vertex and/or Edge classes
  - I will use 'id' as an array index. Since Vertex class already has 'id' variable, so I will make 'id' variable inside the Edge class too. And to access 'id' variable from outside, need to change the access modifier 'private' to 'public' on each classes.
- Implement the maps using ordinary arrays.
  - I will allocate array variables for 'handles', 'parentEdges', 'weights' in the default construct of ShortestPaths class like this. And access each array by Vertex.id or Edge.id
  - For `private HashMap<Vertex, Edge> parentEdges;`

```
// Allocate array for a parentEdges as many as number of Vertices
// The type is Edge class because. It stores Edges for Vertices
Edge [] arrayForParentEdges = new Edge [g.getNumVertices()];

// It can be accessed like this
arrayForParentEdges[Vertex.id];
```

- For `private HashMap<Vertex, Decreaser<VertexAndDist>> handles;`

```
// Allocate array for handles as many as number of Vertices
// The type is Decreaser<VertexAndDist>. It stores Decreaser for Vertices
Decreaser<VertexAndDist> [] arrayForHandles = new Decreaser [g.getNumVertices()];
// It can be accessed like this
arrayForHandles[Vertex.id];
```

- For `private HashMap<Edge, Integer> weights;`

```
// Create and allocate of array for a handles as many as number of Edges
// This array is type of Integer because it will store Integer for Edges
Integer [] arrayForWeights = new Integer [g.getNumEdges()];

// It can be accessed like this
arrayForWeights [Edge.id];
```

4. Suppose we know that our input graph G = (V, E) is dense. What is the asymptotic running time of Dijkstra's algorithm on G in terms of the number of vertices |V |?

- Time complexity of Dijkstra's algorithm using Priority Queue is $O((|E| + |V|) \cdot \log|V|)$
- A dense graph is a graph in which the number of edges is close to the maximal number of edges.
  - So, a dense graph has such relationship $|E| = O(|V|^2)$
- Put $|E| = O(|V|^2)$ into the time complexity equation above and then we get like this
  - $O\big((|V|^2 + |V|) \cdot \log|V|\big) = O(|V|^2 \log|V|)$
- Asymptotic running time is $O(|V|^2 \log|V|)$

5. Now suppose we know that our input graph G = (V, E) is sparse. What is the asymptotic running time in terms of |V |?

- Time complexity of Dijkstra's algorithm using Priority Queue is $O((|E| + |V|) \cdot \log|V|)$
- A sparse graph is a graph with only a few edges.
  - So, a sparse graph has such relationship $|E| = O(|\mathrm{V}|)$
- Put $|E| = O(|\mathrm{V}|)$ into the time complexity equation above and then we get like this
  - $O\big((|V| + |V|) \cdot \log|V|\big) = O(|V| \log|V|)$
- Asymptotic running time is $O(|V| \log|V|)$

6. If the graph is dense, what is the asymptotic complexity of Dijkstra's algorithm using a Fibonacci heap, in terms of |V |?

- Time complexity of Dijkstra's algorithm using Fibonacci heap is $O(|E| + |V| \log|V|)$
- A dense graph is a graph in which the number of edges is close to the maximal number of edges
  - So, a dense graph has such relationship $|E| = O(|V|^2)$
- Put $|E| = O(|V|^2)$ into the time complexity equation above and then we get like this
  - $O(|V|^2 + |V| \log|V|) = O(|V|^2)$
- Asymptotic running time is $O(|V|^2)$

7. If the graph is sparse, what is the asymptotic complexity of Dijkstra's algorithm using a Fibonacci heap, in terms of |V |?

- Time complexity of Dijkstra's algorithm using Fibonacci heap is $O(|E| + |V| \log|V|)$
- A sparse graph is a graph with only a few edges
  - So, a sparse graph has such relationship $|E| = O(|\mathrm{V}|)$
- Put $|E| = O(|\mathrm{V}|)$ into the time complexity equation above and then we get like this
  - $O(|V| + |V| \log|V|) = O(|V| \log|V|)$
- Asymptotic running time is $O(|V| \log|V|)$