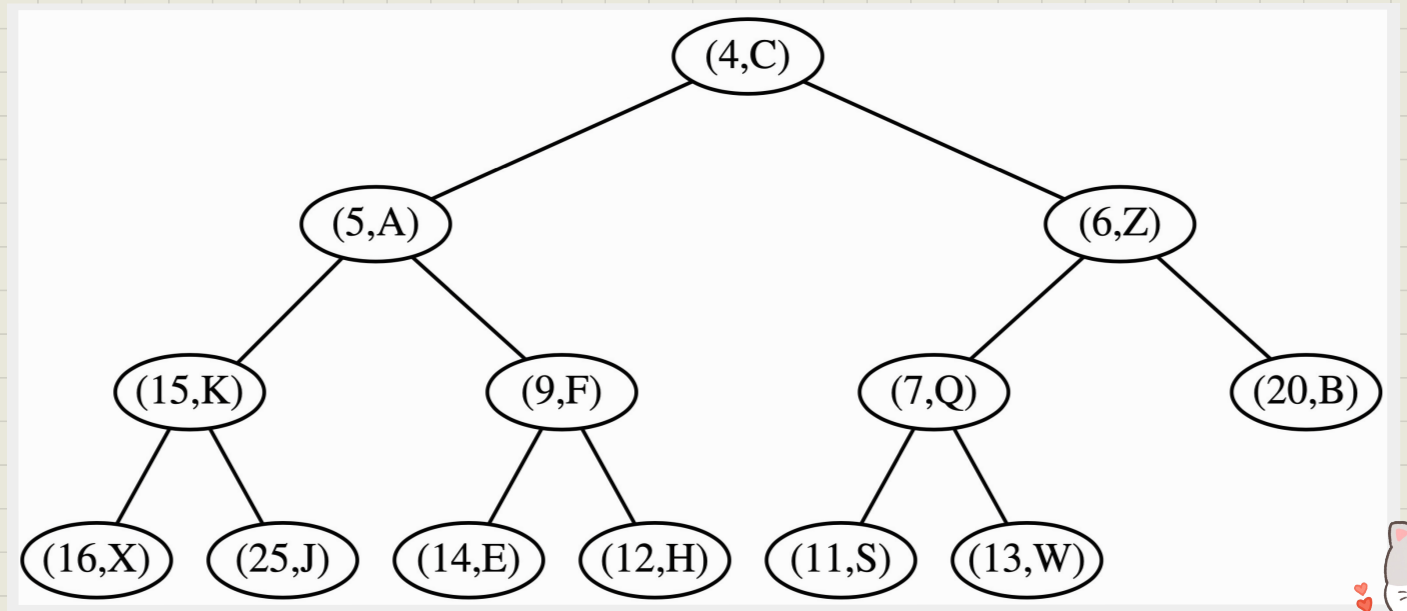
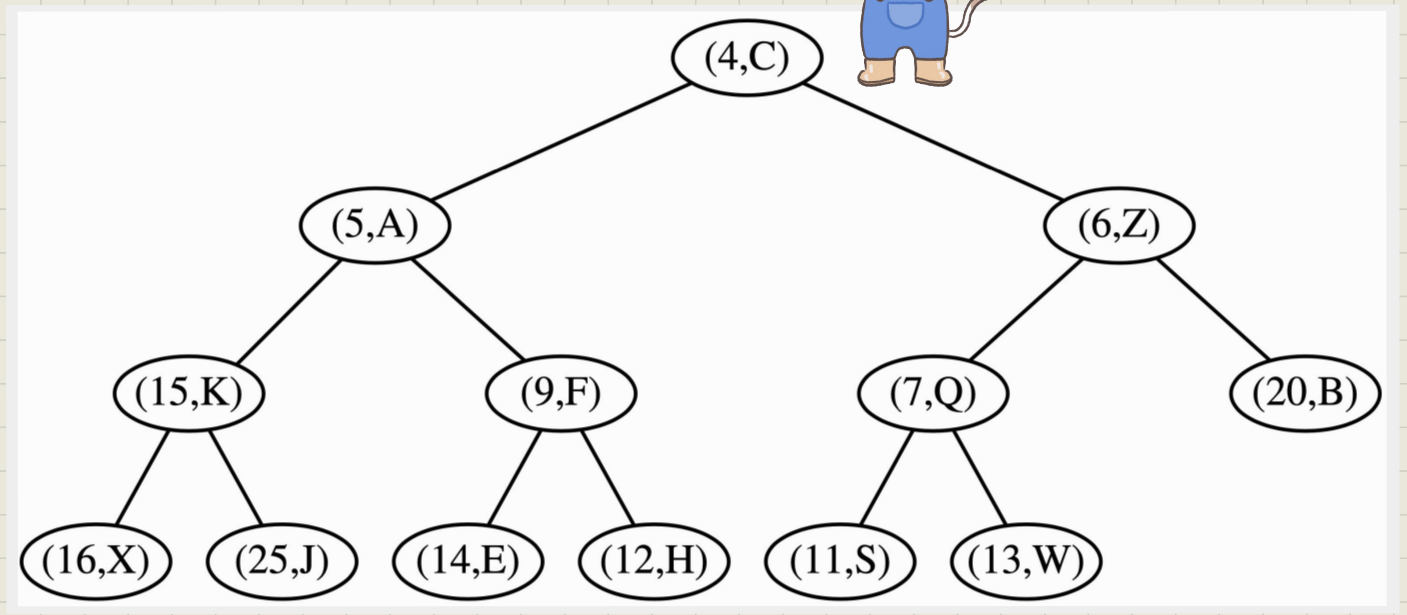
Heap Operations: Insertion

Insert a new entry (2, T)



Heap Operations: Deletion

Delete the root/minimum



Dijkstra's Shortest Path Algorithm: Time Complexity

```
1  ALGORITHM: Dijkstra-Shortest-Path
2    INPUT: Graph G = (V, E); Source Vertex s ∈ V
3    OUTPUT: For t ∈ V (t ≠ s),
4      • D(t) := d(s, t)
5      • Shortest Path: ⟨s, ..., a(a(t)), a(t), t⟩
6  PROCEDURE:
7    D(s) = 0
8    for(t ∈ (V ∖ {s})): D(t) := ∞
9    for(v ∈ V): a(v) := nil
10   for(v ∈ V): Q.insert(v) -- Q is a PQ keyed by D
11   while ( ¬Q.isEmpty() ):
12     u := Q.min()
13     for(v adjacent to u):
14       if(v ∈ Q ∧ D(u) + w(u, v) < D(v)):
15         D(v) := D(u) + w(u, v)
16         a(v) := u
17       else:
18         skip
19     Q.removeMin()
```

Graphs in Java: Adjacency Matrix Strategy (1)

```
class AdjacencyMatrixGraph<V, E> implements Graph<V, E> {
  private DoublyLinkedList<AdjacencyMatrixVertex<V>> vertices;
  private DoublyLinkedList<EdgeListEdge<E, V>> edges;
  private boolean isDirected;

  private EdgeListEdge<E, V>[][] matrix;

  /* initialize an empty graph */
  AdjacencyMatrixGraph(boolean isDirected) {
    this.vertices = new DoublyLinkedList<>();
    this.edges = new DoublyLinkedList<>();
    this.isDirected = isDirected;
  }
```

```
public class Vertex<V> {
  private V element;
  public Vertex(V element) { this.element = element; }
  /* setter and getter for element */
}
```

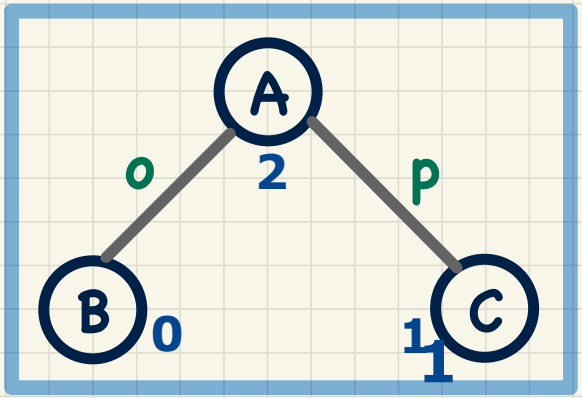
```
public class Edge<E, V> {
  private E element;
  private Vertex<V> origin;
  private Vertex<V> dest;
  public Edge(E element) { this.element = element; }
  /* setters and getters for element, origin, and destination */
}
```

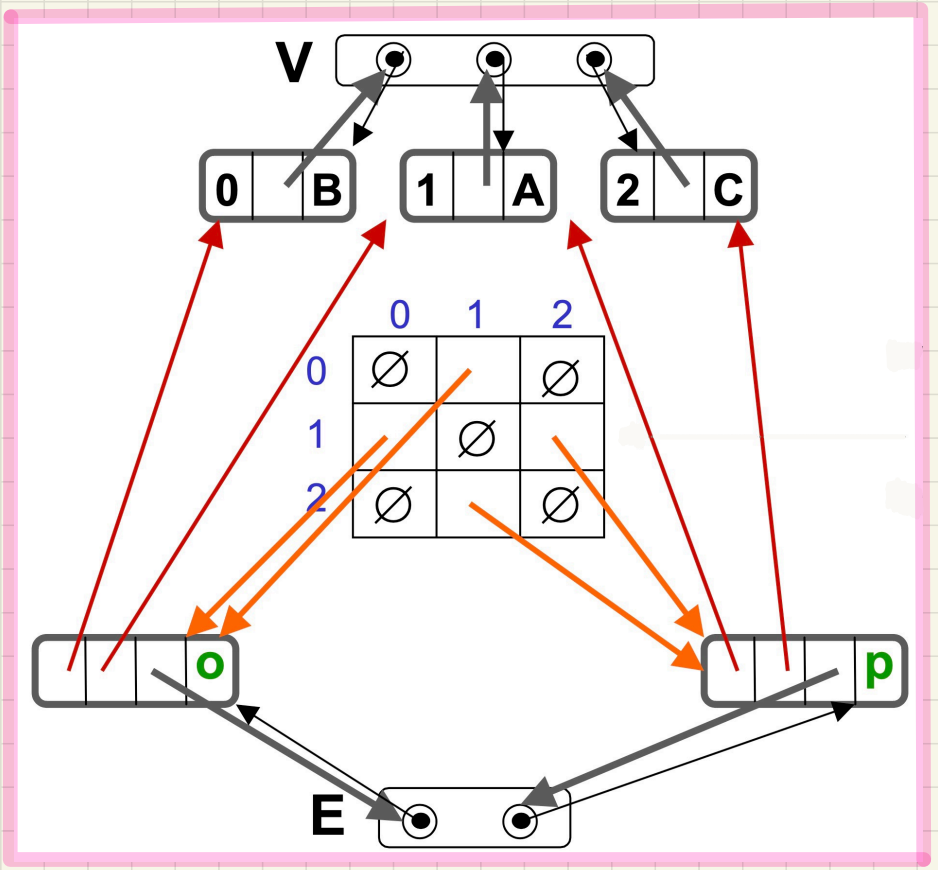
```
public class EdgeListVertex<V> extends Vertex<V> {
  public DLNode<Vertex<V>> vertextListPosition;
  /* setter and getter for vertexListPosition */
}
```

```
public class EdgeListEdge<E, V> extends Edge<E, V> {
  public DLNode<Edge<E, V>> edgeListPosition;
  /* setter and getter for edgeListPosition */
}
```

```
class AdjacencyMatrixVertex<V> extends EdgeListVertex<V> {
  private int index;
  /* getter and setter for index */
}
```

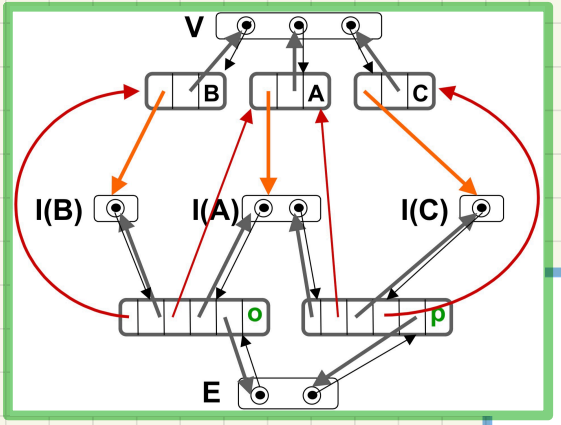
Graphs in Java: Adjacency Matrix Strategy (2)

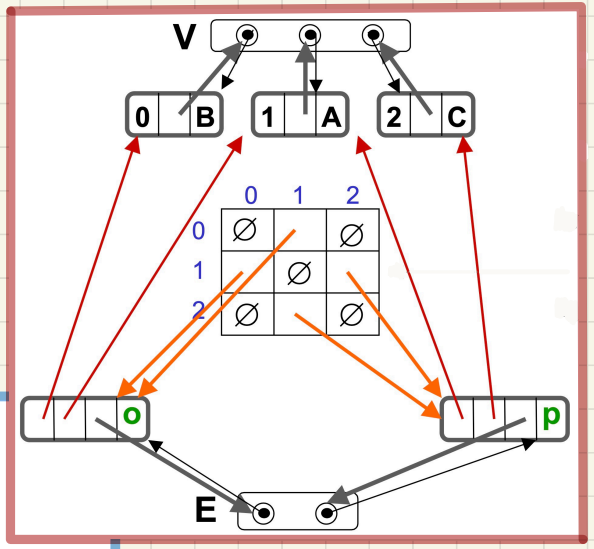


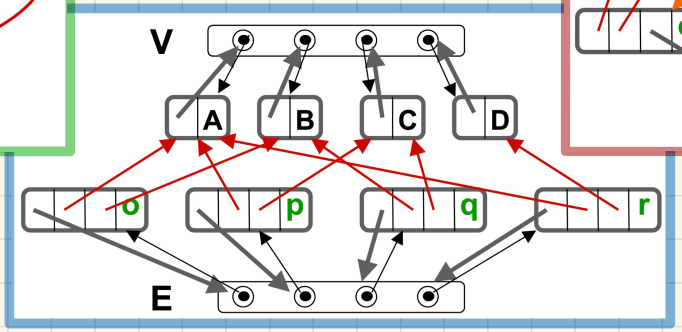


Graphs in Java: Strategies

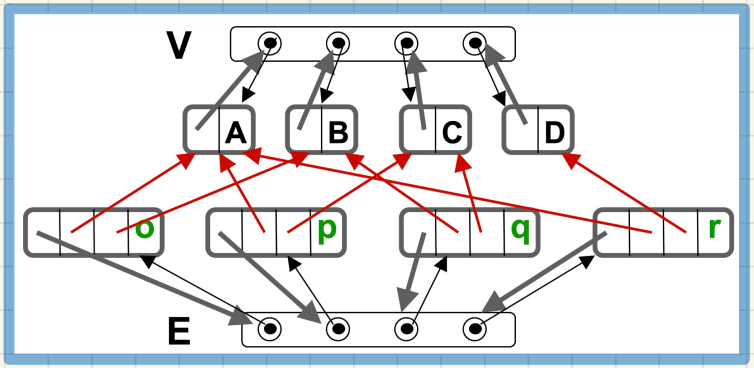
	Vertex	Edge	Graph
Adjacency List	incidentEdges	originIncidentListPos	isDirected vertices edges
		destIncidentListPos	
Edge List	vertexListPosition	edgeListPosition	
Adjacency Matrix	index		matrix



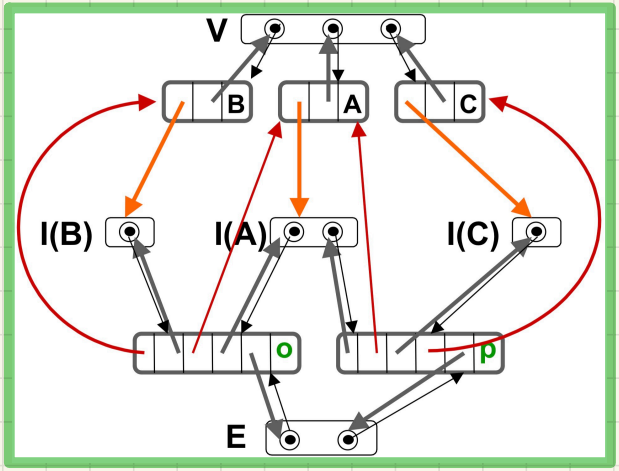


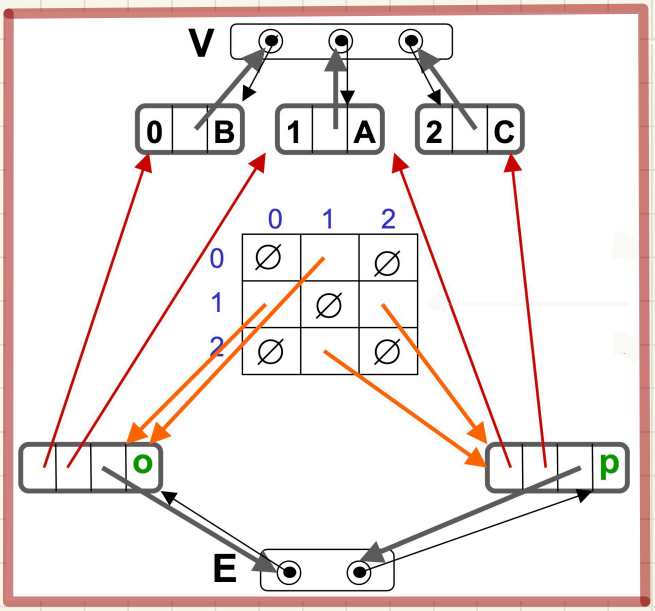


Graphs in Java: Time Complexities (1)

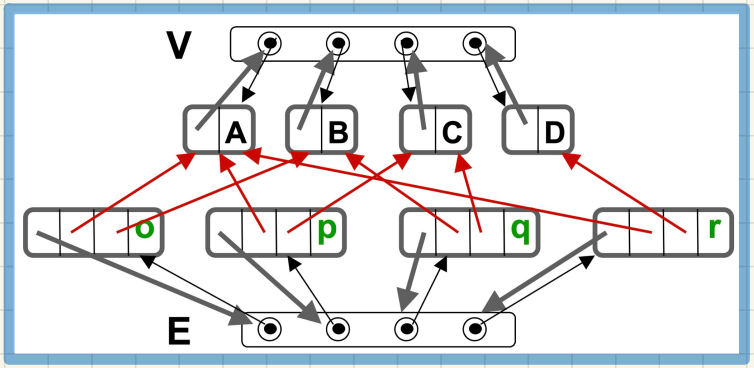


numVertices(), numEdges()

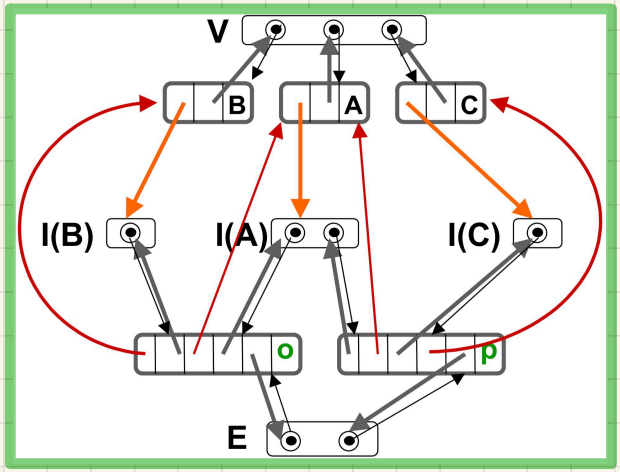


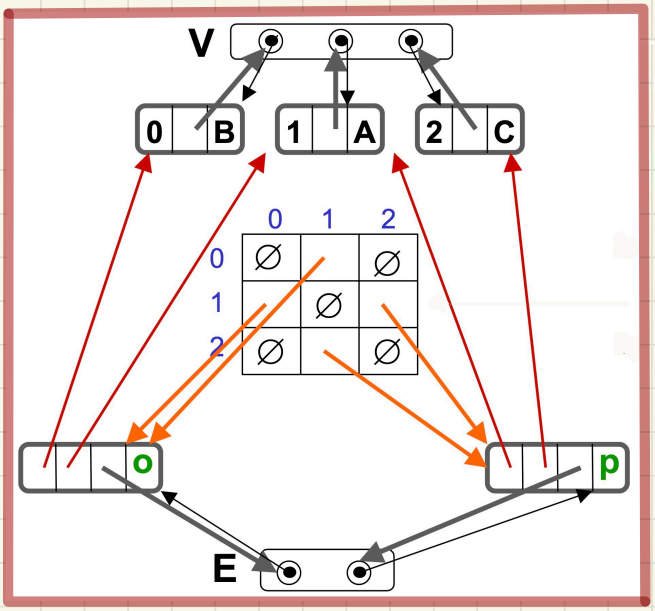


Graphs in Java: Time Complexities (2)

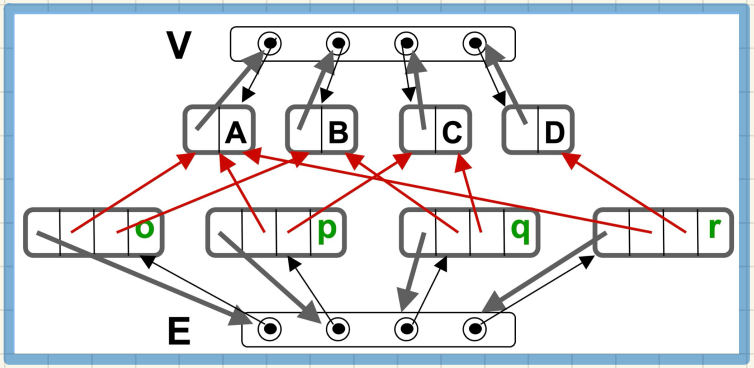


vertices(), edges()

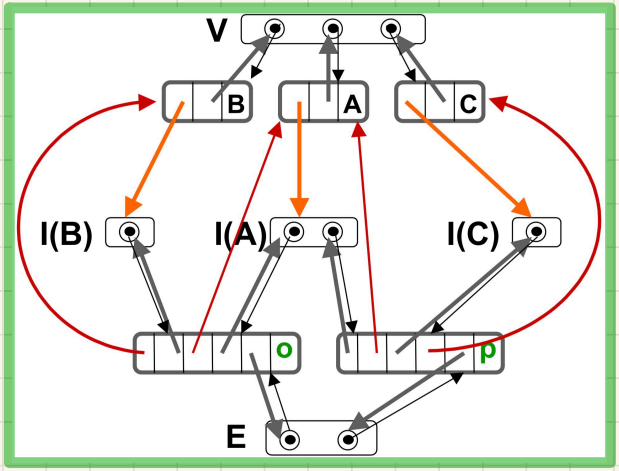


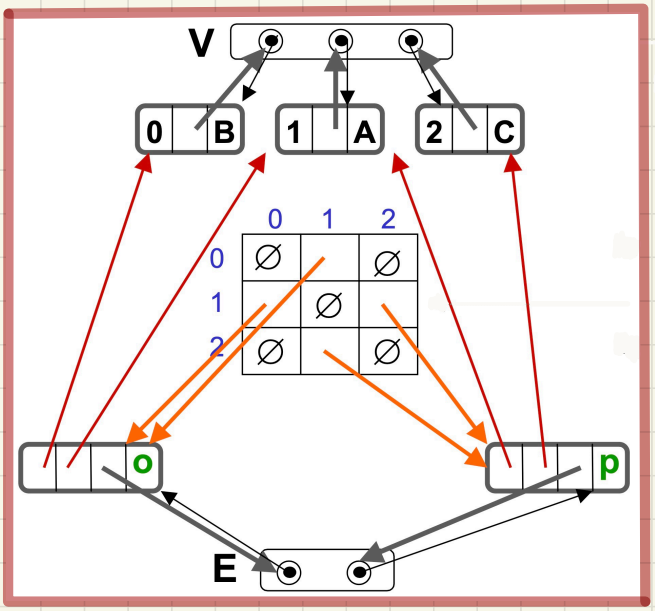


Graphs in Java: Time Complexities (3)

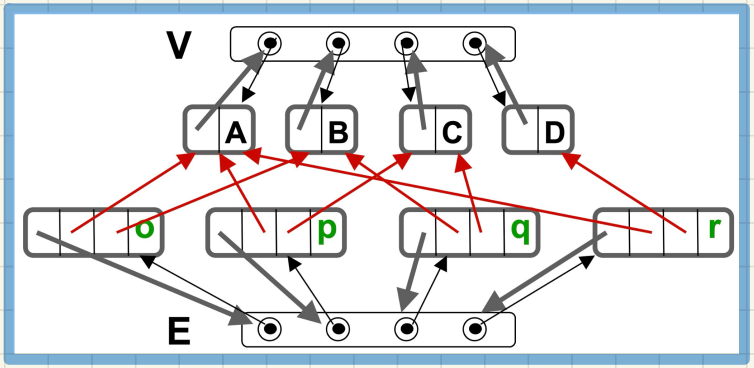


getEdge(u, v)

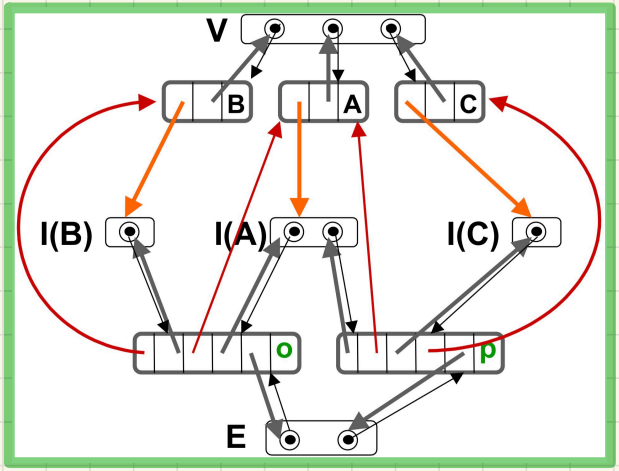


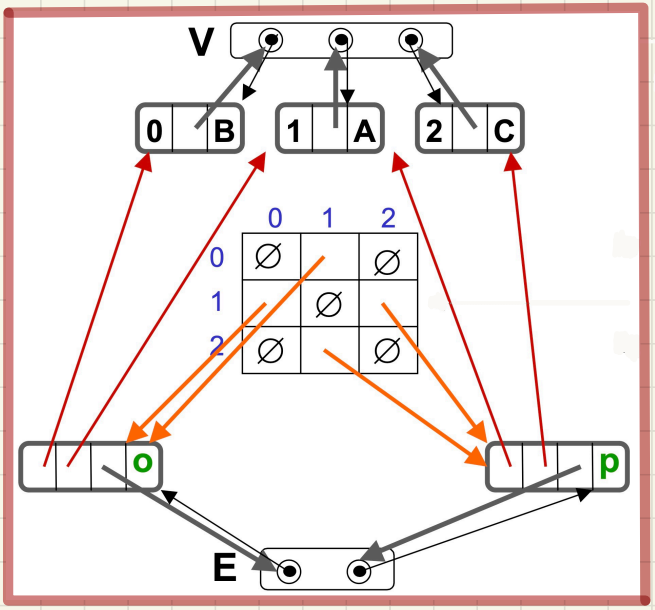


Graphs in Java: Time Complexities (4)

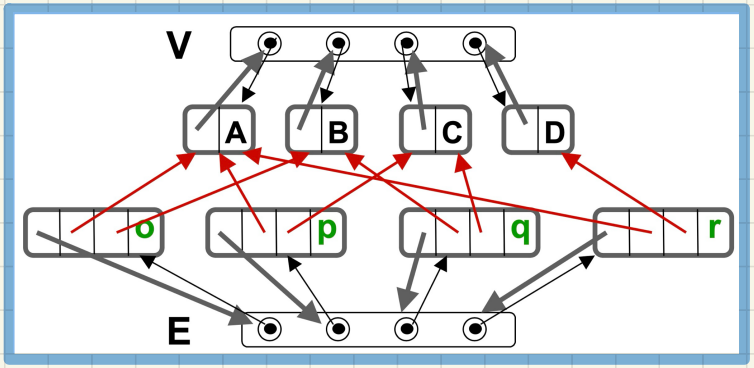


outDegree(u), inDegree(u)
inEdges(v), outEdges(v)

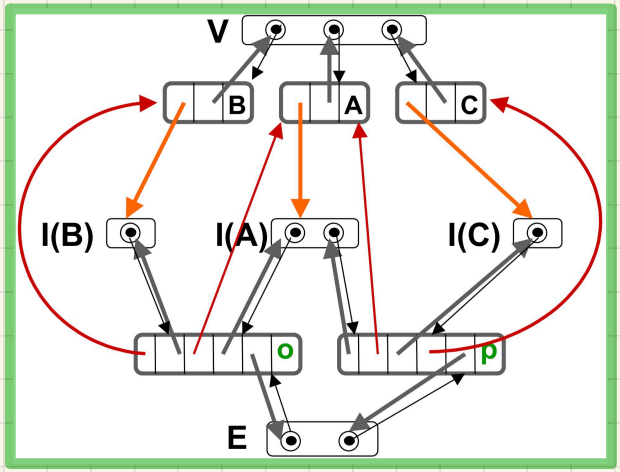


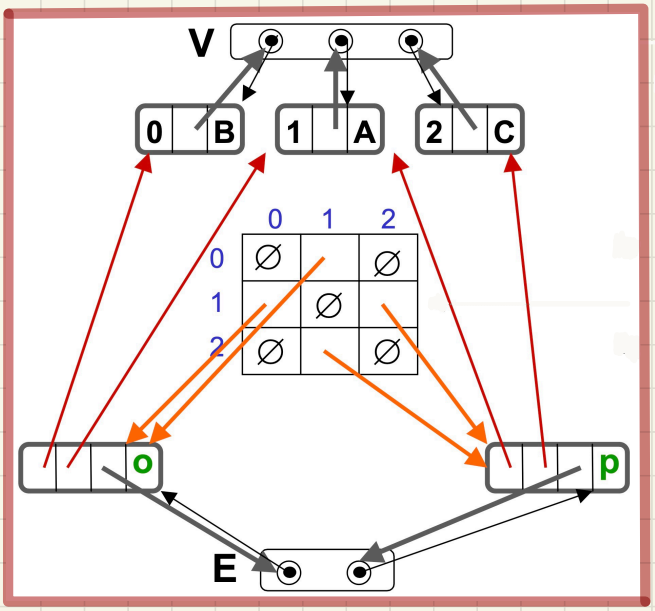


Graphs in Java: Time Complexities (5)

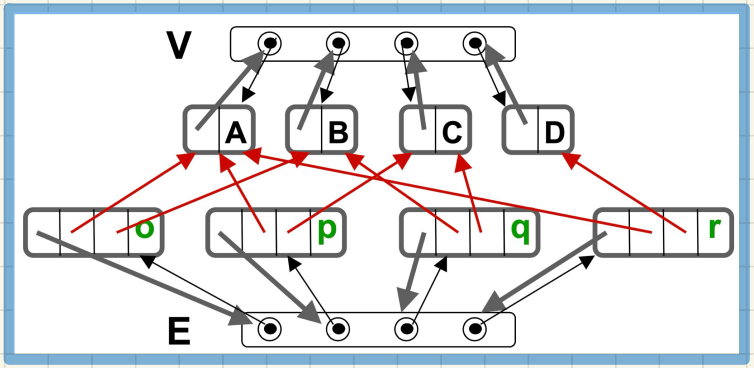


insertVertex(x)

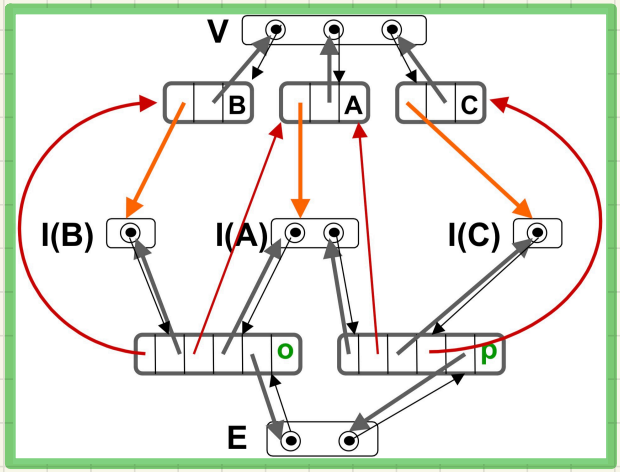


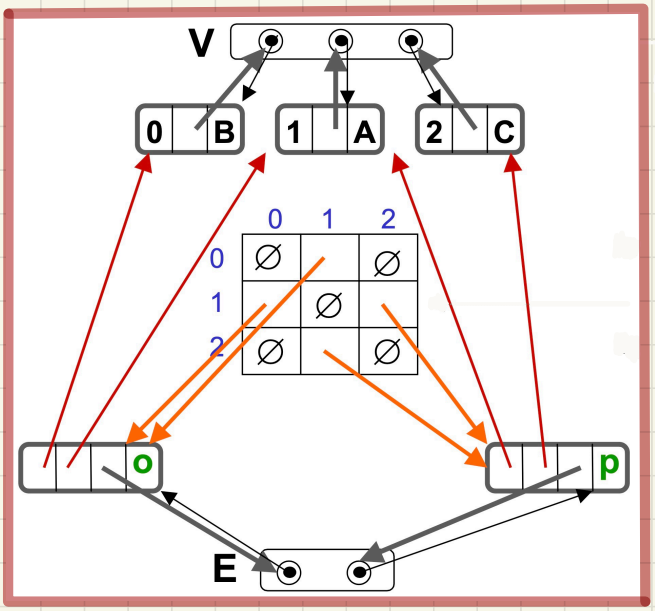


Graphs in Java: Time Complexities (6)

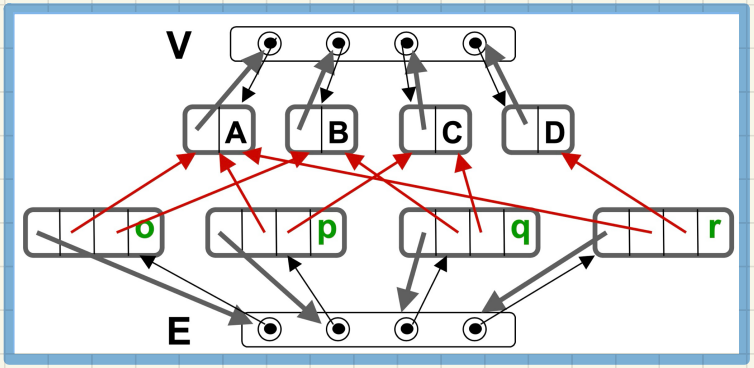


removeVertex(v)





Graphs in Java: Time Complexities (7)



insertEdge(u, v, x),
removeEdge(e)