INSTITUTO TECNOLÓGICO DE SONORA

Reporte Complejidad Temporal

Equipo:
Abraham Coronel Bringas / 252233
José Adolfo Ortega Ruiz / 252882
José Ramón Reynaga Núñez / 253020
Luis Rafael Lagarda Encina / 252607

Materia: Análisis de algoritmos

7 de Julio de 2025

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Complejidad Temporal

Grafos:

agregarVertice:

```
public void agregarVertice(Vertice nodo) {
    adyacencias.putIfAbsent(nodo, new ArrayList<>());
}
```

T(n) = 2
O(1)

agregarArista:

```
public void agregarArista(Vertice origen, Vertice destino, double peso) {
    if (!adyacencias.containsKey(origen) || !adyacencias.containsKey(destino)) {
        throw new IllegalArgumentException("Ambos nodos deben existir en el grafo");
    }
    adyacencias.get(origen).add(new Arista(destino, peso));
    adyacencias.get(destino).add(new Arista(origen, peso));
}
```

T(n) = 6
O(1)

getAdyacencias:

```
public Map<Vertice, List<Arista>> getAdyacencias() {
    return adyacencias;
}
```

T(n) = 1
O(1)

getVertices:

```
public Set<Vertice> getVertices() {
    return adyacencias.keySet();
}
```

T(n) = 2
O(1)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

getVecinos:

```java
public List<Arista> getVecinos(Vertice nodo) {
    return adyacencias.getOrDefault(nodo, Collections.emptyList());
}
```

1 + 1 + 1

T(n) = 3
O(1)

imprimirGrafo:

```java
public void imprimirGrafo() {
    for (Vertice nodo : adyacencias.keySet()) {
        System.out.println(nodo + " conectado a:");
        for (Arista arista : adyacencias.get(nodo)) {
            System.out.println("    " + arista);
        }
    }
}
```

n+n+n
n
3n²
n²

T(n) = 4n^2 + 4n
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

AlgoritmosBusqueda:

BFS:

```
public Grafo BFS(Grafo grafo, Vertice origen) {
    //Guarda el orden de visita de cada vertice
    List<Vertice> ordenVisitas = new ArrayList<>();

    //Guarda los vertices ya visitados
    Map<Vertice, Boolean> visitado = new HashMap<>();

    // Grafo que representara el BFS
    Grafo grafoResultante = new Grafo();

    //Pone todos los vertices como no visitados
    for (Vertice v : grafo.getVertices()) {
        visitado.put(v, false);
        // Agregar todos los vértices al grafo resultante
        grafoResultante.agregarVertice(v);
    }

    //Cola para gestionar el orden de visita
    Queue<Vertice> cola = new ArrayDeque<>();
    //Marca y encola el vertice origen
    visitado.put(origen, true);
    cola.offer(origen);

    while (!cola.isEmpty()) {
        //Saca el siguiente vertice de la cola
        Vertice actual = cola.poll();
        ordenVisitas.add(actual);

        //Obtiene todos los vecinos del vertice actual
        for (var arista : grafo.getVecinos(actual)) {
            Vertice vecino = arista.getDestino();
            //Va marcando los vertices que no esten visitados
            if (!visitado.get(vecino)) {
                visitado.put(vecino, true);
                cola.offer(vecino);
                grafoResultante.agregarArista(actual, vecino, arista.getPeso());
            }
        }
    }
    return grafoResultante;
}
```

T(n) = 11n^2 + 5n + 4
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

DFS:

```java
public Grafo DFS(Grafo grafo, Vertice origen) {
    //Guarda el orden de visita de cada vertice
    List<Vertice> ordenVisitas = new ArrayList<>();
    //Guarda los vertices ya visitados
    Map<Vertice, Boolean> visitado = new HashMap<>();
    // Grafo que representara el DFS
    Grafo grafoResultante = new Grafo();

    //Pone todos los vertices como no visitados
    for (Vertice v : grafo.getVertices()) {
        visitado.put(v, false);
        // Agregar todos los vertices al grafo resultante
        grafoResultante.agregarVertice(v);
    }

    //Pila para gestionar el orden de visita
    Deque<Vertice> pila = new ArrayDeque<>();
    // Marcar el origen como visitado y agregarlo a la pila
    visitado.put(origen, true);
    pila.push(origen);
    ordenVisitas.add(origen);
    while (!pila.isEmpty()) {
        Vertice actual = pila.pop();
        // Explorar vecinos del vertice actual
        for (var arista : grafo.getVecinos(actual)) {
            Vertice vecino = arista.getDestino();
            if (!visitado.get(vecino)) {
                visitado.put(vecino, true);
                pila.push(vecino);
                ordenVisitas.add(vecino);
                grafoResultante.agregarArista(actual, vecino, arista.getPeso());
            }
        }
    }
    return grafoResultante;
}
```

T(n) = 10n^2 + 10n + 7
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Dijkstra:

```
public Grafo Dijkstra(Grafo grafo, Vertice origen) {
    //Mapa para almacenar la distancia minima desde el origen a cada vertice
    Map<Vertice, Double> distancias = new HashMap<>(); 1
    //Mapa para ir guardando el vertice previo al camino optimo
    Map<Vertice, Vertice> predecesores = new HashMap<>();1
    //Grafo que representa el Dijkstra
    Grafo arbolCaminos = new Grafo(); 1
    //Cola que ordena los vertices por distancia
    PriorityQueue<NodoDistancia> cola = inicializarEstructuras(grafo, origen, distancias, predecesores);

    for (Vertice v : grafo.getVertices()) {
        arbolCaminos.agregarVertice(v); n
    }

    //Procesa la cola hasta dejarla vacia
    procesarCola(grafo, distancias, predecesores, cola);

    for (Vertice v : predecesores.keySet()) {
        Vertice predecesor = predecesores.get(v);
        if (predecesor != null) {
            // Buscar el peso de la arista en el grafo original
            double peso = obtenerPesoArista(grafo, predecesor, v);
            arbolCaminos.agregarArista(predecesor, v, peso);
        }
    }

    //Retorna los resultados de las distancias y sus predecesores
    return arbolCaminos; 1
}
```

T(n) = 4n^2 + 37n + 4
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

Métodos privados implementados en Dijkstra:

iniciarlizarEstructuras:

```
private PriorityQueue<NodoDistancia> inicializarEstructuras(
        Grafo grafo, Vertice origen,
        Map<Vertice, Double> distancias,
        Map<Vertice, Vertice> predecesores) {

    //Crea la cola de prioridad ordenada por distancia de manera ascendente
    PriorityQueue<NodoDistancia> cola = new PriorityQueue<>(
            Comparator.comparingDouble(n -> n.distancia)
    );

    //Inicializa todos los vertices en infinito
    for (Vertice v : grafo.getVertices()) {
        distancias.put(v, Double.POSITIVE_INFINITY);
        predecesores.put(v, null);
    }

    //Pone las distancias en 0
    distancias.put(origen, 0.0);
    cola.offer(new NodoDistancia(origen, 0.0));

    return cola;
}
```

T(n) = 6n + 7
O(n)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

actualizarDistancias:

```
private void actualizarDistancias(
        Vertice vecino,
        double nuevaDistancia,
        Vertice predecesor,
        Map<Vertice, Double> distancias,
        Map<Vertice, Vertice> predecesores,
        PriorityQueue<NodoDistancia> cola) {

    //Actualiza la distancia minima para el vertice
    distancias.put(vecino, nuevaDistancia);
    //Guarda un nuevo predecesor en el camino optimo
    predecesores.put(vecino, predecesor);
    //Inserta en la cola para el procesamiento
    cola.offer(new NodoDistancia(vecino, nuevaDistancia));
}
```

T(n) = 4
O(1)

procesarVecino:

```
private void procesarVecino(
        Vertice actual,
        Arista arista,
        Map<Vertice, Double> distancias,
        Map<Vertice, Vertice> predecesores,
        PriorityQueue<NodoDistancia> cola) {

    Vertice vecino = arista.getDestino();
    double distanciaActual = distancias.get(actual); //Distancia del vertice actual
    double nuevaDistancia = distanciaActual + arista.getPeso(); //Calcula la nueva distancia
    double distanciaVecino = distancias.get(vecino); //Mejor distancia conocida del vecino

    //Comprueva si se encuentra un mejor camino
    if (nuevaDistancia < distanciaVecino) {
        actualizarDistancias(vecino, nuevaDistancia, actual, distancias, predecesores, cola);
    }
}
```

T(n) = 14
O(1)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

procesarCola:

```
private void procesarCola(
        Grafo grafo,
        Map<Vertice, Double> distancias,
        Map<Vertice, Vertice> predecesores,
        PriorityQueue<NodoDistancia> cola) {

    while (!cola.isEmpty()) {
        //Agarra el vertice con menor distancia acumulada
        NodoDistancia actual = cola.poll();
        Vertice u = actual.nodo;

        //Verifa si encuentra una mejor distancia
        if (actual.distancia > distancias.get(u)) {
            continue;
        }

        //Iterar sobre los vecinos
        for (Arista arista : grafo.getVecinos(u)) {
            procesarVecino(u, arista, distancias, predecesores, cola);
        }
    }
}
```

T(n) = 4n^2 + 11n
O(n^2)

obtenerPesoArista:

```
private double obtenerPesoArista(Grafo grafo, Vertice origen, Vertice destino) {
    for (Arista arista : grafo.getVecinos(origen)) {
        if (arista.getDestino().equals(destino)) {
            return arista.getPeso();
        }
    }
    return 0.0;
}
```

T(n) = 7n + 1
O(n)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Bellman:

```
public static Map<String, Object> BellmanFord(Grafo grafo, Vertice origen, Vertice destino) {
    Map<Vertice, Double> distancias = new HashMap<>();
    Map<Vertice, Vertice> predecesores = new HashMap<>();
    Map<String, Object> resultado = new HashMap<>();

    for (Vertice v : grafo.getVertices()) {
        distancias.put(v, Double.MAX_VALUE);
        predecesores.put(v, null);
    }
    distancias.put(origen, 0.0);

    int numVertices = grafo.getVertices().size();
    for (int i = 1; i < numVertices; i++) {
        for (Vertice u : grafo.getVertices()) {
            for (Arista arista : grafo.getVecinos(u)) {
                Vertice v = arista.getDestino();
                double peso = arista.getPeso();
                if (distancias.get(u) != Double.MAX_VALUE && distancias.get(u) + peso < distancias.get(v)) {
                    distancias.put(v, distancias.get(u) + peso);
                    predecesores.put(v, u);
                }
            }

        }
    }
    for (Vertice u : grafo.getVertices()) {
        for (Arista arista : grafo.getVecinos(u)) {
            Vertice v = arista.getDestino();
            double peso = arista.getPeso();
            if (distancias.get(u) != Double.MAX_VALUE && distancias.get(u) + peso < distancias.get(v)) {
                resultado.put("ciclo negativo", true);
                resultado.put("distancia", Double.NEGATIVE_INFINITY);
                resultado.put("ruta", new java.util.ArrayList<>());
                return resultado;
            }
        }
    }
    resultado.put("ciclo negativo", false);

    List<Vertice> ruta = new java.util.ArrayList<>();
    Vertice paso = destino;
    if (predecesores.get(paso) != null || paso.equals(origen)) {
        while (paso != null) {
            ruta.add(paso);
            paso = predecesores.get(paso);

        }
    }
    Collections.reverse(ruta);
    if (ruta.isEmpty() || !ruta.get(0).equals(origen)) {
        resultado.put("ruta", new java.util.ArrayList<>());//aqui devuelve una ruta vacia
    } else {
        resultado.put("ruta", ruta);
    }

    resultado.put("distancia", distancias.get(destino));

    return resultado;
}

}
```

T(n) = 37n^2 + 14n + 29

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

O(n^2)

MapaPanel

encontrarVerticeCercano:

```
private Vertice encontrarVerticeCercano(Coordinate coord) {
    double minDistancia = Double.MAX_VALUE;
    Vertice verticeCercano = null;
    double umbral = 0.01;

    for (Vertice v : grafo.getVertices()) {
        double distancia = distanciaCoord(coord, new Coordinate(v.getLatitud(), v.getLongitud()));
        if (distancia < umbral && distancia < minDistancia) {
            minDistancia = distancia;
            verticeCercano = v;
        }
    }
    return verticeCercano;
}
```

T(n) = 10n + 4
O(n)

distanciaCoord:

```
private double distanciaCoord(Coordinate c1, Coordinate c2) {
    double lat = c1.getLat() - c2.getLat();
    double lon = c1.getLon() - c2.getLon();
    return Math.sqrt(lat * lat + lon * lon);
}
```

T(n) = 13
O(1)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

actualizarGrafo:

```
public void actualizarGrafo(Grafo g) throws InterruptedException {
    getMapMarkerList().clear();
    lineas.forEach(this::removeMapPolygon);
    lineas.clear();

    for (Vertice nodo : g.getVertices()) {
        Coordinate coord = new Coordinate(nodo.getLatitud(), nodo.getLongitud());
        MapMarkerDot marker = new MapMarkerDot(nodo.getNombre(), coord);
        if (nodo.equals(verticeSeleccionado)) {
            marker.setBackColor(Color.GREEN);
        } else {
            marker.setBackColor(Color.RED);
        }
        marker.setColor(Color.WHITE);
        marker.setName(nodo.getNombre());
        addMapMarker(marker);
    }

    for (Vertice nodo : g.getVertices()) {
        Coordinate coordOrigen = new Coordinate(nodo.getLatitud(), nodo.getLongitud());

        for (Arista arista : g.getVecinos(nodo)) {
            Vertice destino = arista.getDestino();
            Coordinate coordDestino = new Coordinate(destino.getLatitud(), destino.getLongitud());

            List<Coordinate> puntos = new ArrayList<>();
            puntos.add(coordOrigen);
            puntos.add(coordDestino);
            puntos.add(coordDestino);

            MapPolygon linea = new CustomPolyline(puntos, Color.BLUE);
            addMapPolygon(linea);
            lineas.add(linea);

        }
    }
    repaint();
}
```

T(n) = 16n^2 + 21n + 5
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

dibujarGrafo:

```
private void dibujarGrafo() {
    getMapMarkerList().clear();
    lineas.forEach(this::removeMapPolygon);
    lineas.clear();

    for (Vertice nodo : grafo.getVertices()) {
        Coordinate coord = new Coordinate(nodo.getLatitud(), nodo.getLongitud());
        MapMarkerDot marker = new MapMarkerDot(nodo.getNombre(), coord);
        marker.setBackColor(Color.RED);
        marker.setColor(Color.WHITE);
        marker.setName(nodo.getNombre());
        addMapMarker(marker);
    }

    for (Vertice nodo : grafo.getVertices()) {
        Coordinate coordOrigen = new Coordinate(nodo.getLatitud(), nodo.getLongitud());

        for (Arista arista : grafo.getVecinos(nodo)) {
            Vertice destino = arista.getDestino();
            Coordinate coordDestino = new Coordinate(destino.getLatitud(), destino.getLongitud());

            List<Coordinate> puntos = new ArrayList<>();
            puntos.add(coordOrigen);
            puntos.add(coordDestino);
            puntos.add(coordDestino);

            MapPolygon linea = new CustomPolyline(puntos, Color.BLUE);
            addMapPolygon(linea);
            lineas.add(linea);
        }
    }
    repaint();
}
```

T(n) = 15n^2 + 21n + 5
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

AlgoritmosMST:

Métodos de la clase unionFind:

makeSet:

```
public void makeSet(Set<Vertice> vertices) {
    for (Vertice v : vertices) {
        padre.put(v, v);
    }
}
```

T(n) = 3n
O(n)

find:

```
public Vertice find(Vertice v) {
    if (padre.get(v) != v) {
        padre.put(v, find(padre.get(v)));
    }
    return padre.get(v);
}
```

T(n) = T(h−1)+O(1)
O(n)

union:

```
    public void union(Vertice u, Vertice v) {
        Vertice raizU = find(u);
        Vertice raizV = find(v);
        if (!raizU.equals(raizV)) {
            padre.put(raizU, raizV);
        }
    }
}
```

T(n) = Tfind(Hu) + Tfind(Hv) + O(1)
O(n)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Kruskal:

```
public static Grafo aplicarKruskal(Grafo grafoOriginal) {
    Grafo mst = new Grafo();
    double w = 0;

    for (Vertice v : grafoOriginal.getVertices()) {
        mst.agregarVertice(v);
    }

    List<Arista> aristas = new ArrayList<>();
    Set<String> vistas = new HashSet<>();
    for (Vertice origen : grafoOriginal.getVertices()) {
        for (Arista arista : grafoOriginal.getVecinos(origen)) {
            String id = origen.getNombre() + "-" + arista.getDestino().getNombre();
            String idReverso = arista.getDestino().getNombre() + "-" + origen.getNombre();
            if (!vistas.contains(id) && !vistas.contains(idReverso)) {
                aristas.add(new AristaPair(origen, arista));
                vistas.add(id);
            }
        }
    }

    aristas.sort(Comparator.comparingDouble(Arista::getPeso));

    UnionFind uf = new UnionFind();
    uf.makeSet(grafoOriginal.getVertices());

    StringBuilder str = new StringBuilder("");
    str.append("Aristas seleccionadas para el MST:\n");
    for (Arista arista : aristas) {
        Vertice u = ((AristaPair) arista).origen;
        Vertice v = arista.getDestino();
        if (!uf.find(u).equals(uf.find(v))) {
            mst.agregarArista(u, v, arista.getPeso());
            uf.union(u, v);
            str.append("• ").append(u.getNombre())
                    .append(" — ").append(v.getNombre())
                    .append(" (").append(arista.getPeso()).append(" km)\n");
            w+=arista.getPeso();
        }
    }

    str.append("\nPeso total: ").append(w).append(" KM");
    JOptionPane.showMessageDialog(null, str.toString(), str.toString(), JOptionPane.INFORMATION_MESSAGE);

    return mst;
}
```

O(n log n)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Boruvka:

```
public static Grafo aplicarBoruvka(Grafo grafoOriginal) {
    Grafo mst = new Grafo();
    grafoOriginal.getVertices().forEach(mst::agregarVertice);

    UnionFind uf = new UnionFind();
    uf.makeSet(grafoOriginal.getVertices());

    long numComponentes = grafoOriginal.getVertices().size();

    while (numComponentes > 1) {
        Map<Vertice, AristaPair> aristasMasBaratas = new HashMap<>();

        for (Vertice origen : grafoOriginal.getVertices()) {
            for (Arista arista : grafoOriginal.getVecinos(origen)) {
                Vertice destino = arista.getDestino();

                Vertice raizOrigen = uf.find(origen);
                Vertice raizDestino = uf.find(destino);

                if (raizOrigen != null && !raizOrigen.equals(raizDestino)) {

                    AristaPair aristaActualOrigen = aristasMasBaratas.get(raizOrigen);
                    if (aristaActualOrigen == null || arista.getPeso() < aristaActualOrigen.getPeso()) {
                        aristasMasBaratas.put(raizOrigen, new AristaPair(origen, arista));
                    }

                    AristaPair aristaActualDestino = aristasMasBaratas.get(raizDestino);
                    if (aristaActualDestino == null || arista.getPeso() < aristaActualDestino.getPeso()) {
                        aristasMasBaratas.put(raizDestino, new AristaPair(origen, arista));
                    }
                }
            }
        }

        long componentesAntesDeUnion = contarComponentes(grafoOriginal.getVertices(), uf);

        for (AristaPair aristaBarata : aristasMasBaratas.values()) {
            Vertice u = aristaBarata.origen;
            Vertice v = aristaBarata.getDestino();

            if (!uf.find(u).equals(uf.find(v))) {
                mst.agregarArista(u, v, aristaBarata.getPeso());
                uf.union(u, v);
            }
        }

        numComponentes = contarComponentes(grafoOriginal.getVertices(), uf);

        if (numComponentes == componentesAntesDeUnion) {
            break;
        }
    }
    return mst;
}
```

O(n log n)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda

INSTITUTO TECNOLÓGICO DE SONORA

Prim:

```java
//prim
public static Grafo aplicarprim(Grafo grafoOriginal){
    Grafo mst = new Grafo();
    Set<Vertice> visitados = new HashSet<>();
    PriorityQueue<AristaPair> cola = new PriorityQueue<>(Comparator.comparingDouble(Arista::getPeso));

    if(grafoOriginal.getVertices().isEmpty()) return mst;

    Vertice inicio = grafoOriginal.getVertices().iterator().next();
    mst.agregarVertice(inicio);
    visitados.add(inicio);

    for (Arista arista : grafoOriginal.getVecinos(inicio)) {
        cola.add(new AristaPair(inicio,arista));
    }

    while(!cola.isEmpty() && visitados.size() < grafoOriginal.getVertices().size()){
        AristaPair arista = cola.poll();
        Vertice origen = arista.origen;
        Vertice destino = arista.getDestino();
        if (visitados.contains(destino)) continue;

        mst.agregarVertice(destino);
        mst.agregarArista(origen, destino, arista.getPeso());
        visitados.add(destino);
        for (Arista siguiente : grafoOriginal.getVecinos(destino)){
            if (!visitados.contains(siguiente.getDestino())){
                cola.add(new AristaPair(destino,siguiente));
            }
        }
    }
    return mst;
}
```

T(n) = 8n^2 + 22n + 15
O(n^2)

Abraham Coronel
Adolfo Ortega
Ramon Reynaga
Luis Lagarda