Grafos Dirigidos

• Un grafo dirigido G consiste en un conjunto de vértices V y un conjunto de aristas A.

• Una arista es un par ordenado de vértices (v,w) donde v → w.

• w es adyacente a v.

Caminos

• Camino: Secuencia de vértices v1 → v2 → v3 → ...

• Longitud: número de aristas.

• Camino simple: vértices distintos (excepto origen/destino).

• Ciclo simple: comienza y termina en el mismo vértice.

Representación de Grafos

Matriz de Adyacencia:

- Matriz n×n booleana o con costos.

- Espacio: $O(n^2)$. Lectura: $O(n^2)$.

Listas de Adyacencia:

- Lista de adyacentes al vértice.

- Espacio proporcional a vértices + aristas.

- Verificar conexión: O(n).

Dijkstra (Pseudocódigo Simplificado)

Procedimiento Dijkstra(Grafo, Costos, Origen):

Para cada vértice v:

distancia[v] ← ∞

predecesor[v] ← NULL

distancia[origen] ← 0

Q ← conjunto de todos los vértices

Mientras Q no esté vacío:

u ← vértice en Q con menor distancia

Q ← Q sin u

Para cada vecino v de u:

Si v ∈ Q y distancia[u] + costo[u][v] < distancia[v]:

distancia[v] ← distancia[u] + costo[u][v]

predecesor[v] ← u

Retornar distancia, predecesor

Floyd-Warshall (Pseudocódigo Simplificado)

Floyd(A, C):

Para cada i, j:

A[i][j] ← C[i][j]

P[i][j] ← 0

Para cada i: A[i][i] ← 0

Para k desde 0 hasta n:

Para i desde 0 hasta n:

Para j desde 0 hasta n:

Si A[i][k] + A[k][j] < A[i][j]:

A[i][j] ← A[i][k] + A[k][j]

P[i][j] ← k

Retornar P

Excentricidad y Centro

• Excentricidad de v: distancia máxima desde v a otro nodo.

• Centro: nodo con mínima excentricidad.

Pasos:

1. Aplicar Floyd-Warshall.

2. Obtener máximo por fila (excentricidad).

3. Elegir vértice con mínimo valor.

Búsqueda en Profundidad (DFS)

```
bpf(origen):
  visitados ← conjunto vacío
  bpfRecursivo(origen, visitados)
  retornar lista de vértices visitados

bpfRecursivo(actual, visitados):
  marcar actual como visitado
  Para cada adyacente de actual:
   Si no está visitado:
    bpfRecursivo(destino, visitados)
```

Obtención de Caminos

```
obtenerCamino(destino, caminoActual, todosLosCaminos):
        marcar actual como visitado
        agregar actual al caminoActual
        Si actual == destino:
        agregar copia de caminoActual a todosLosCaminos
        Para cada adyacente w de actual:
                Si w no visitado:
                        obtenerCamino(w, caminoActual, todosLosCaminos)
                        quitar actual de caminoActual
```

Grafos Dirigidos Acíclicos (DAG)

- No tienen ciclos.
- Útiles para expresar dependencias o expresiones comunes.
- Para detectar ciclos: ejecutar DFS, buscar arcos de retroceso.

Clasificación Topológica

- Ordenar nodos de modo que si i → j, entonces i precede a j.

Procedimiento:

1. Elegir nodo no visitado.

2. Realizar DFS.

3. Agregar nodo al principio de la lista (orden inverso).