I. Algoritmos

1. Algoritmo de dijkstra:

Dado un grafo conexo, dirigido o no dirigido, con distancias no negativas; el algoritmo de Dijkstra empieza por marcar el nodo fuente e inicializar a 0 o a infinito las distancias acumuladas de cada nodo. Posteriormente itera, mientras queden nodos sin marcar, seleccionando en cada etapa el arco con la suma más pequeña de la distancia acumulada por uno de los nodos marcados más la distancia del arco que lo une a uno de los nodos no marcados, guardando este último nodo como marcado y actualizando su distancia acumulada con la suma antes calculada. Al final del proceso tendremos un arbol de expansión del camino más corto.

Algorithm 1 Dijkstra
Entrada: $G = (V, A); D, s \in V$
Salida: $G^T = (V^T, A^T); W$

$A^T \leftarrow \emptyset$
$V^T \leftarrow \{s\}$
$W_{1,|V|} \leftarrow 0$
while $|V^T| < |V|$ **do**
    seleccionar arcos $(i, j) \in A$ con $i \in V^T$ y $j \in VV^T$
    elegir un arco (i, j) de los anteriores tal que $d_{ij} + W_i$ sea mínimo
    $A^T \leftarrow A^T \cup \{(i, j)\}$
    $V^T \leftarrow V^T \cup \{j\}$
    $W_j \leftarrow d_{ij} + W_j$
end while

2. Algoritmo de Bellman-Ford:

El algoritmo de Bellman-Ford empieza inicializando a 0 la distancia asociada a la fuente y a infinito las distancias acumuladas del resto de nodos. Posteriormente, lleva a cabo el proceso de relajación iterando en todos los nodos distintos de la fuente, comprobando en cada arco que sale de él si la distancia hacia otro nodo es mayor que la suma de la distancia acumulada del primero más el peso del arco que los une, en caso afirmativo establece este resultado como la distancia acumulada por el nodo de llegada y guarda el nodo de partida como su predecesor.
Al final termina comprobando si se han encontrado ciclos negativos, revisando arco por arco si se puede reducir la distancia acumulada del nodo de llegada.
Si no se topa con uno, el resultado final es un árbol de expansión del camino mas corto.

Algorithm 2 Bellman-Ford
Entrada: $G = (V, A); D; s \in V$
Salida: $G^T = (V^T, A^T); W$

$A^T \leftarrow \emptyset$
$W_{1:|V|} \leftarrow \infty; W_s \leftarrow 0$
$P \leftarrow \emptyset$
for cada nodo $i \in V\{s\}$ **do**
    for cada arco $(i, j) \in A$ **do**
        if $W_j > W_i + d_{ij}$ **then**
            $W_j \leftarrow W_i + d_{ij}$
            $P_j \leftarrow i$
        end if
    end for
end for
for cada arco $(i, j) \in A$ **do**
    if $W_j > W_i + d_{ij}$ **then**
        No hay solución
    end if
end for
for cada nodo $j \in V\{s\}$ **do**
    $i \leftarrow P_j$
    seleccionar arco (i, j) de A
    $A^T \leftarrow A^T \cup \{(i, j)\}$
end for
$V^T \leftarrow V$

3. **Algoritmo de Gusfield:**
La principal ventaja del algoritmo de Gusfield es, precisamente, evitar el requisito de comprimir nodos. El algoritmo arranca construyendo un arbol con el primer nodo como nodo único, e itera posteriormente añadiendo cada vez uno nuevo de acuerdo al orden $2, 3, ..., |V|$. En cada iteración la duda radica en escoger a qué nodo del árbol se une cada nuevo nodo k cuando existe más de una posibilidad.

Algorithm 3 Gusfield
Entrada: $G = (V, A); Z$
Salida: $G^{T'} = (V^{T'}, A^{T'}); Z^T$

$V^T \leftarrow \{1\}$
$A^T \leftarrow \emptyset$
$Z^T_{|V|X|V|} \leftarrow \infty$
for cada nodo $i \in V1$ **do**
    $V^T_k \leftarrow V^T; A^T_k \leftarrow A^T$
    while $|V^T_k| > 1$ **do**
        borrar un arco $(i, j) \in A^T_k$ cuyo $a_{i,j}$ sea mínimo
        determinar componentes T_1 y T_2 de V^T_k y A^T_k tal que $i \in T_1$ y $j \in T_2$
        obtener corte mínimo entre i y j en el grafo $G = (V, A)$ original
        determinar componentes S_1 y S_2 de V^T y A^T tal que $i \in S_1$ y $j \in S_2$
        if nodo $i \in S_1$ **then**
            $V^T_k \leftarrow$ nodos de la componente T_1
            $A^T_k \leftarrow$ arcos de la componente T_1
        else
            $V^T_k \leftarrow$ nodos de la componente T_2
            $A^T_k \leftarrow$ arcos de la componente T_2
        end if
    end while
    añadir arco (i, k), con $k \in V^T_k$, al árbol A^T
    obtener corte mínimo entre i y k en el grafo $G = (V, A)$ original
    $Z^T_{ik} \leftarrow$ capacidad del corte mínimo $i - k$
    $V^T \leftarrow V^T \cup \{i\}$
end for