“Año del Bicentenario del Perú: 200 años de Independencia”

UNIVERSIDAD NACIONAL DE SAN AGUSTIN

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

CURSO:

LABORATORIO ESTRUCTURA DE DATOS Y ALGORITMOS

ALUMNO:

FABRICIO ALONSO BALAREZO DELGADO

AREQUIPA – PERÚ
6-8-2021

1. **¿Cuántas variantes del algoritmo de Dijkstra hay y cuál es la diferencia entre ellas?**

2. **Investigue sobre los ALGORITMOS DE CAMINOS MINIMOS e indique, ¿Qué similitudes encuentra, qué diferencias, en qué casos utilizar y por qué?**

A continuación, algunos algoritmos encontrados, con una pequeña descripción de cada uno, creador y sus aplicaciones. Al final se presentarán las similitudes y diferencias.

- Algoritmo de Johnson

  Este algoritmo se isa para hallar el camino mas corto entre todos los vértices de un grafo dirigido. Importante saber que este algoritmo utiliza a otros, específicamente, Bellman-Ford y Dijkstra. Su creador fue Donald B. Johnson.

  Para aplicar este algoritmo se crea un nodo externo V(q) conectado a todos los demás con una arista de peso 0. Se utiliza el algoritmo de BELLMAN-FORD, empezando por el nuevo vértice V(q). Este determinara la longitud del camino mínimo desde V(q) hasta V(i) para cada vértice V(i). Si se detecta un circuito negativo, el algoritmo termina.

  Luego para cada nodo V(i) ∈ V se usa el algoritmo de DIJKSTRA para determinar el camino más corto entre V(i) y los otros nodos del grafo, usando el grafo con pesos modificados, los cuales, nos hemos asegurado de que son todos positivos, y por tanto, nos aseguramos que el algoritmo de DIJKSTRA encuentra caminos mínimos óptimos.

  Pseudocódigo

```
Tipos
Arista = REGISTRO
    o : NATURAL
    d : NATURAL
    peso : INT
    sig : NATURAL
FIN
LAristas = PUNTERO A Arista
TGrafo = ARRAY [1..N] DE LAristas
THv = ARRAY [1..N] DE ENTERO
TVector = ARRAY [1..N] DE ENTERO
TMatriz = ARRAY [1..N] DE TVector
```

```
//suponemos ig>1
PROC Johnson (↓grafo: TGrafo; ↓ig: NATURAL; ↑ distancias: TMatriz ; ↑ previos: TMatriz)
VARIABLES
    i : NATURAL
    p : LAristas
    min_caminos : THv
    aux_dist, aux_prev : TVector
INICIO
    grafo[ig] ← nueva_arista(ig,1,0,NULO)
    inc(ig)
    p ← grafo[ig]
    PARA i ← 2 HASTA ig-2 HACER
        p^.sig ← nueva_arista(ig,i,0,NULO)
        p ← p^.sig
    FIN
    BellmanFord(grafo,ig, min_caminos)
    PARA i ← 1 HASTA ig-1 HACER
        p ← grafo[i]
        MIENTRAS (p != NULO) HACER
            p^.peso ← p^.peso + min_caminos[p^.o] - min_caminos[p^.d]
            p ← p^.sig
        FIN
    FIN
    PARA i ← 1 HASTA ig-2 HACER
        Dijkstra(grafo,i, aux_dist,aux_prev)    // devuelve los caminos mínimos desde el último nodo
                                                // a todos los demás
        previos[i] ← aux_prev;
        CalcularDistancias(grafo, previos, aux_dist,distancias); // este algoritmo realiza la transformación inversa a la
                                                                 // que habíamos hecho antes sobre los pesos, para obtener
                                                                 // las distancias reales
    FIN
FIN
```

- Algoritmo de Floyd-Warshall

  Este es un Algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. Este algoritmo es un claro ejemplo de **programación dinámica**. Fue descrito en 1959 por Bernard Roy. Y reúne a dos algoritmos Floyd y Warshall.

  Puede comprobar, además de encontrar caminos mínimos, si un grafo es bipartito. Para su ejecución se crea dos matrices nxn siendo n la cantidad de vértices. En una es denominada matriz de peso y distancia y la segunda matriz es la del recorrido. Al finalizar el algoritmo se obtendrá los caminos mas óptimos para llegar de un nodo a otro (“Todos contra todos”) y en una sola **ejecución.**

```
1 /* Suponemos que la función pesoArista devuelve el coste del camino que va de i a j
2    (infinito si no existe).
3 También suponemos que n es el número de vértices y pesoArista(i,i) = 0
4 */
5
6 int camino[][];
7 /* Una matriz bidimensional. En cada paso del algoritmo, camino[i][j] es el camino mínimo
8 de i hasta j usando valores intermedios de (1..k-1). Cada camino[i][j] es inicializado a
9
10 */
11
12 procedimiento FloydWarshall ()
13    para k: = 0 hasta n – 1
14
15          camino[i][j] = mín ( camino[i][j], camino[i][k]+camino[k][j])
16
17    fin para
```

Algoritmo de Floyd-Warshall en Pseudocódigo

- Algoritmo de Bellman-Ford

  El algoritmo de Bellman-Ford genera el camino más corto en un grafo dirigido ponderado, incluso el peso de las aristas puede ser negativo. Fue desarrollado por Richard **Bellman**, Samuel End y Lester **Ford.**

La ejecución comienza por cada nodo y calcula la distancia entre él mismo y todos los demás dentro de un AS y almacena esta información en una tabla. Luego cada nodo envía su tabla a todos los nodos vecinos. Finalmente, cuando un nodo recibe las tablas de distancias de sus vecinos, éste calcula la ruta más corta a los demás nodos y actualiza su tabla para reflejar los cambios.

```
bool BellmanFord_Optimizado(Grafo G, nodo_origen s)
     // inicializamos el grafo. Ponemos distancias a INFINITO menos el nodo origen que
     // tiene distancia 0. Para ello lo hacemos recorriéndonos todos los vértices del grafo
     for v ∈ V[G] do
         distancia[v]=INFINITO
         padre[v]=NULL
     distancia[s]=0
     encolar(s, Q)
     en_cola[s]=TRUE
     while Q!=0 then
         u = extraer(Q)
         en_cola[u]=FALSE
         // relajamos las aristas
         for v ∈ ady[u] do
             if distancia[v]>distancia[u] + peso(u, v) then
                 distancia[v] = distancia[u] + peso (u, v)
                 padre[v] = u
                 if en_cola[v]==FALSE then
                     encolar(v, Q)
                     en_cola[v]=TRUE
```

Algoritmo de Bellman-Ford en Pseudocódigo