HOJA Nº

FECHA

Dijkstra para ciudad: A

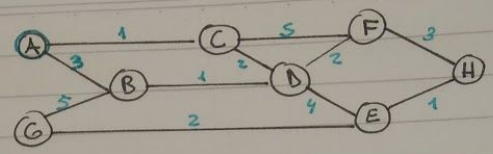
Iteración 0

$S = \{A\}$

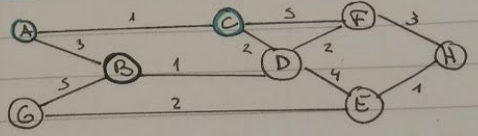
$C = \{A, B, C, D, E, F, G, H\}$

Vertice	Padre	Peso
A	–	0
B	–	–
C	–	–
D	–	–
E	–	–
F	–	–
G	–	–
H	–	–

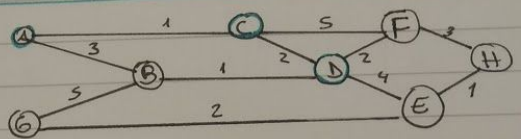
I0



I1



I2



Iteración 1

$S = \{A, C\}$

V	P	Peso
A	–	0
B	A	3
C	A	1
D	–	–
E	–	–
F	–	–
G	–	–
H	–	–

iTe: 2

$S = \{A, C, D\}$

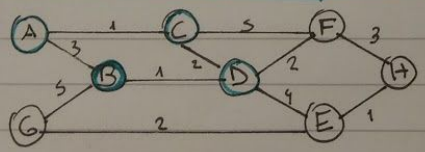
V	P	Peso
A	–	0
B	A	3
C	A	1
D	C	3
E	–	–
F	C	6
G	–	–
H	–	–

iTe: 3

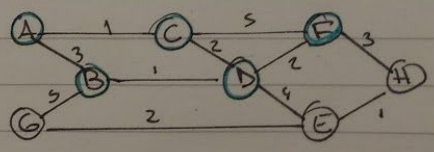
$S = \{A, C, D, B\}$

V	P	Peso
A	–	0
B	A	3
C	A	1
D	C	3
E	D	7
F	D	5
G	–	–
H	–	–

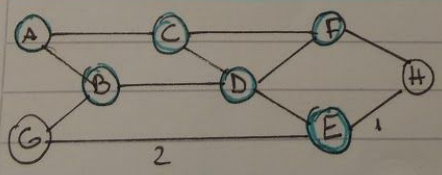
iTe 3



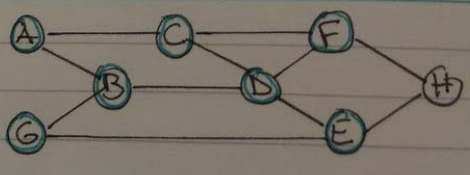
iTe 4



iTe 5



iTe 6



NOTA

HOJA N°

FECHA

ite 4

S = {A, C, D, B, F}

V	P	Peso
A	-	0
B	A	3
C	A	1
D	C	3
E	D	7
F	D	5
G	B	8
H	-	-

ite 5

S = {A, C, D, B, F, E}

V	P	Peso
A	-	0
B	A	3
C	A	1
D	C	3
E	D	7
F	D	5
G	B	8
H	F	8

ite 6

S = {A, C, D, B, F, E, G}

V	P	Peso
A	-	0
B	A	3
C	A	1
D	C	3
E	D	7
F	D	5
G	B	8
H	F	8

Iteracion 7 S = {A, C, D, B, F, E, G, H}

Condiciones:

{ Ciudad = A — Puertos = F, H }

Recorro S para obtener el puerto mas cercano

S = { A, C, D, B, F, E, G, H }

Obtengo que es F.

Recorro d para obtener el camino

d	Vertice	A	B	C	d	e	f	g	h
	Peso	0	3	1	3	7	5	8	5
	Padre	-	A	A	C	D	D	B	F

Solucion para ciudad A = {A, C, D, F}

NOTA

```
Class ServicioEnvio {
    private Grafo g;
    private Hashtable<String, Resultado> d; // contenedor para dijkstra
    private list<String> s; // solución dijkstra
    private list<String> puertos;
    private void dijkstra ( String ciudad ) {
        candidatos = g.getVertices();
        S.clear(); // S = { ciudad }
        S.add(ciudad);
        For (String candidato : candidatos)
            // Inicializo el Hash d
            d.put (candidato, new Resultado (null, 0);
        candidatos.quitar(ciudad);
        While (! candidatos.vacio() ) {
            X = seleccionar (candidatos); (1)
            candidatos.quitar (X);
            If (esFactible (X, C) { (2)
                S.add (X);
                For ( String candidato: candidatos)
                IF { Si el peso es menor desde la nueva ciudad
                MIN (d.get(X).getPeso(); g.getArco(X, candidato).getPeso(),
                     d.get (candidato).getPeso()
                ACTUALIZO el valor en d
                Sino lo dejo
            }
        } if ! solucion(S) (3)
            d.clear();
    }
```

```
class Resultado {
    String padre;
    int peso;
}
```

Funciones para Dijkstra

(1)
```
String seleccionar ( candidatos c) {
    String min = c.get(0);
        For (   cand : c) {
            if d.get(cand).getPeso() < d.get(min).getPeso();
                min = cand;
        } return min;
}
```
// Selecciona el que tiene menor peso de los candidatos

(2)
```
bool esfactible ( C, x) {// No sea un vértice aislado
    LIST adyacentes = g.getAdyacentes(x);
    For ( String ady : adyacentes ) {
        if ( S.contains(ady) || C.contains(ady) {
            return true; }
        }
    return false;
```

(3)
```
bool solucion (S) {
    return S.size() > 1; // hayan vertices a los que se pueda
}                          llegar
```

```
public List<List<String>> getCaminosMinimosPuerto(List<String> ciudades){
    List<List<String>> soluciones = new ArrayList<ArrayList<String>>();
    for(String ciudad : ciudades)
        soluciones.add(getCaminoMinimoPuerto(ciudad));
}
public List<String> getCaminoMinimoPuerto(String ciudad){
    dijkstra(ciudad); // actualizo sgd
    List<String> solucion = new ArrayList();
    String puerto = obtenerPuertoMasCercano(); ④
    if(puerto.vacio()){
        return solucion; // vacia
    }else{
        return obtenerCamino(ciudad, puerto); ⑤
}
④ String obtenerPuertoMasCercano(){
    String puerto = "";
    For(String ciudad : s){
        if(puertos.contains(ciudad)).
            return ciudad;
    return puerto;
} // retorno vacio si ninguno de los vertices es un puerto, de lo contrario tomo el primero que seria el mas cercano
⑤ List<String> obtenerCamino(String ciudad, puerto){
    x = puerto; List solucion; // vacia
    While(!x.equals(ciudad)){
        solucion.add(0,x); // agrego al inicio
        x = hash.get(x).getPadre();
    }
    solucion.add(0, ciudad); // agrego ciudad al inicio
return solucion;
}
```