

### Entregable 3 - Programación 3 - Bianco Manuel

Este pseudo código retorna la distancia q hay q recorrer para llegar un destino que nosotros pasemos, partiendo por el primero nodo del arreglo de nodos cargados.

El programa está implementado en base a una matriz de adyacencias y un grafo chico. Pero podría recibir un grafo de cualquier tamaño.

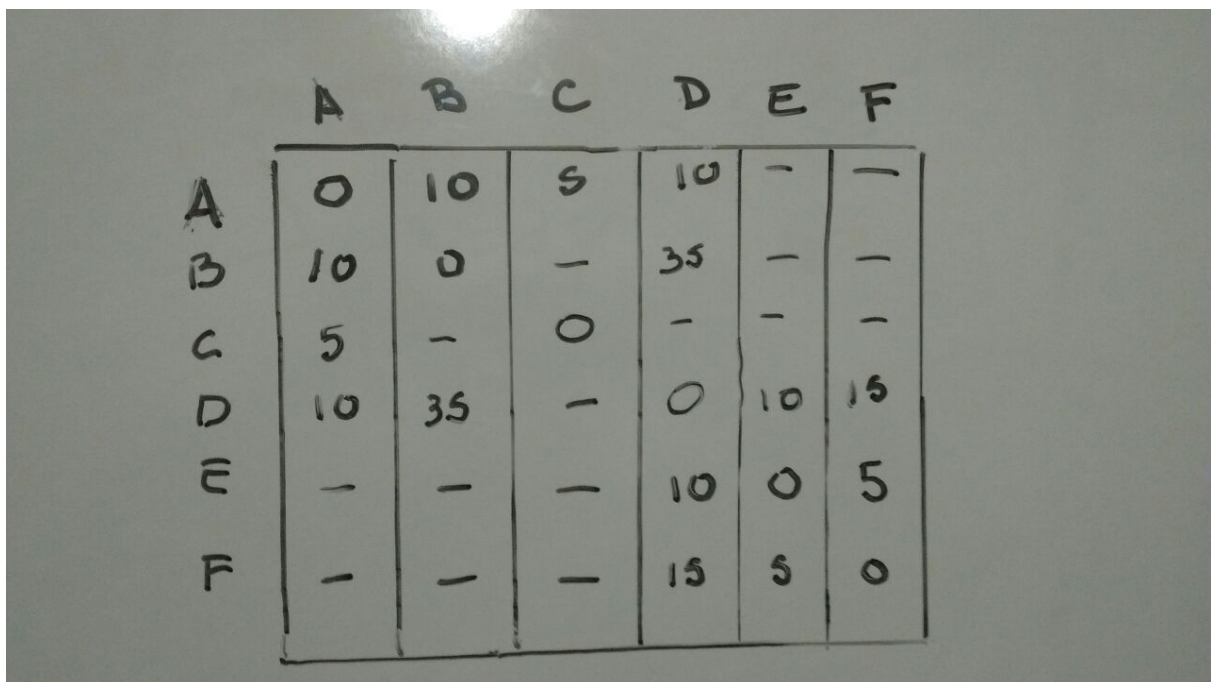
W=arreglo temporal seado con valores infinitos.

C=arreglo de vectores a analizar.

S=arreglo con los nodos con el valor actual mínimo.

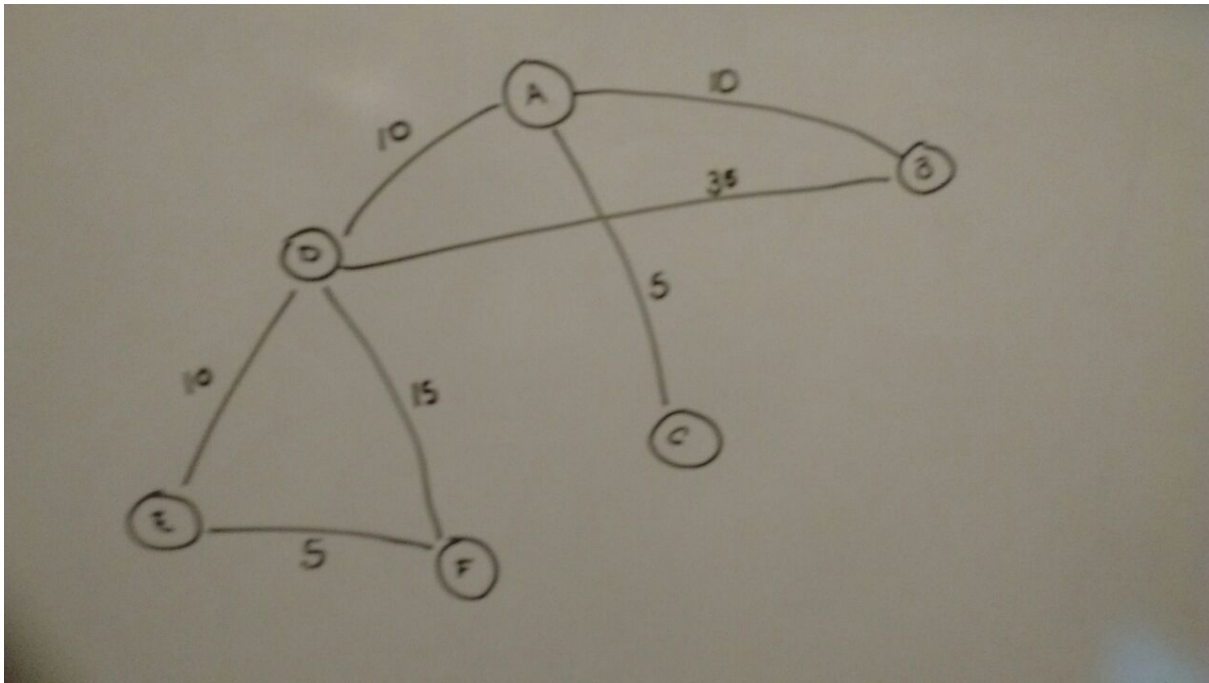
P=arreglo de padres de cada nodo.

M=la matriz dada con los valores



	A	B	C	D	E	F
A	0	10	5	10	-	-
B	10	0	-	35	-	-
C	5	-	0	-	-	-
D	10	35	-	0	10	15
E	-	-	-	10	0	5
F	-	-	-	15	5	0

Parto del siguiente grafo. Dicho grafo está propuesto para poder elegir q nodo es un puerto al momento de ejecutar el programa.



```

function void greedy(destino){ // retorna el tamaño del recorrido hasta el destino
    while (!c.vacio() && !solucion(s)){
        v = w.seleccionar();
        if(factible(v)){
            for(adyacentes de v q no esten en s, i++){
                if(w[i] > w[v] + m[v][i]) {
                    w[i] = w[v] + m[v][i]
                    p[i] = v //se vuelve el padre
                }
            }
        }
    }
}

return w.getAt(posOf(destino)) // retorna el valor total q tiene en la posición a la que
pertenece el destino q nosotros preguntamos
}

```

// en este momento, si termina el while, tiene que quedar el arreglo de padres llenó, el arreglo de vectores vacío, el arreglo de solución lleno, y el arreglo w con los mínimos valores para ir a cada nodo partiendo del a.

```

function boolean factible (vertice v ){
    if(v.adyacentes.size () != 0 && !s.contains(v)){
        s.add(v)
        return true
    }else{
        false
    }
}

```

```

    }
}

function vector seleccionar (){
    int pos = 0, aux = infinito
    vector aux2
    for(w.size(), i++){
        if(w.get(i) < aux && c.get(i) != null && !s.contains(i)){
            aux = w.get(i)
            pos = i
        }
    }
    aux2 = c.get(pos)
    c.borrarEnPos(pos)
    return aux2
}

function boolean solución (s[]){
    return s.size() == cantVectores //cantVectores es una variable seteada con la cantidad
    de nodos a analizar
}

```

Paso a paso:

1- Analiza el nodo A:

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

S={A}

W= | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |

P= | N | N | N | N | N | N |

2- Analiza el nodo C:

C={B,D,E,F}

S={A,C}

W= | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |

P= | N | N | N | N | N | N |

3- Analiza el nodo B:

C={D,E,F}

S={A,C,B}

W= | 0 | 10 | 5 | 10 | ∞ | ∞ |

P= | N | A | A | A | N | N |

4- Analiza el nodo D:

C={E,F}

S={A,C,B,D}  
W= | 0 | 10 | 5 | 10 | 20 | 25 |  
P= | N | A | A | A | D | D |

5- Analiza el nodo E:

C={F}  
S={A,C,B,D,E}  
W= | 0 | 10 | 5 | 10 | 20 | 25 |  
P= | N | A | A | A | D | D |

6- Analiza el nodo F:

C={}  
S={A,C,B,D,E,F}  
W= | 0 | 10 | 5 | 10 | 20 | 25 |  
P= | N | A | A | A | D | D |

Luego de terminar con todos los nodos quedan cargados los arreglos de padres y de distancias hacia cualquier nodo, partiendo del primer nodo analizado.