# Ejercicio 5- TP N4
# entrega greedy

```
dijkstra(puerto ,lista ciudades)
        lista costos[ciudades]; → //ciudades refiere a la cantidad de ciudades
        lista boolean visitado[ciudades]
        lista padres[ciudades]
        para cada costo de costos{ → //inicialización de todos los costos en infinito
                costo=inf
                }
        costo[puerto]=0; →// el costo de mi ciudad actual es igual a 0 ya que no existe costo
        mientras haya ciudades sin visitar {
                actual=seleccionar(costos,visitados) → //seleccionar() la ciudad no visitada
de menor costo

                adyacentes=actual.adyacentes → //se obtienen adyacentes de mi ciudad
actual
                para cada adyacente de adyacentes{
                        //Si encuentro un costo menor del que ya tengo en una ciudad,
actualizo los datos
                        si costo[adyacente]>adyacentes.costo+costo[actual]{
                        costo [adyacente] = adyacente.costo
                        padre[adyacentes]= actual
                                }
                        }
                visitados[actual]=verdadero
                }
        return costos[],padres[];
}

Funcion CaminosMasCortos(Grafo){
ciudades[] = Grafo.ciudades; → //se obtienen todas las ciudades de un grafo dado
puertos[] = Grafo.puertos; → //se obtienen todos los puertos de un grafo dado
caminos[ciudades] = nulo;//conjunto solución que será retornado, el tamaño se corresponde
con la cantidad de ciudades existentes

dijkstras[];

Por cada puerto de puertos{
        dijkstras.agregar(Dijkstra(puerto, ciudades));
        }
Por cada ciudad de ciudades{
        costo_menor = infinito;
        mejor_dijkstra = nulo;

        Por cada dijkstra de Dijkstras{
```

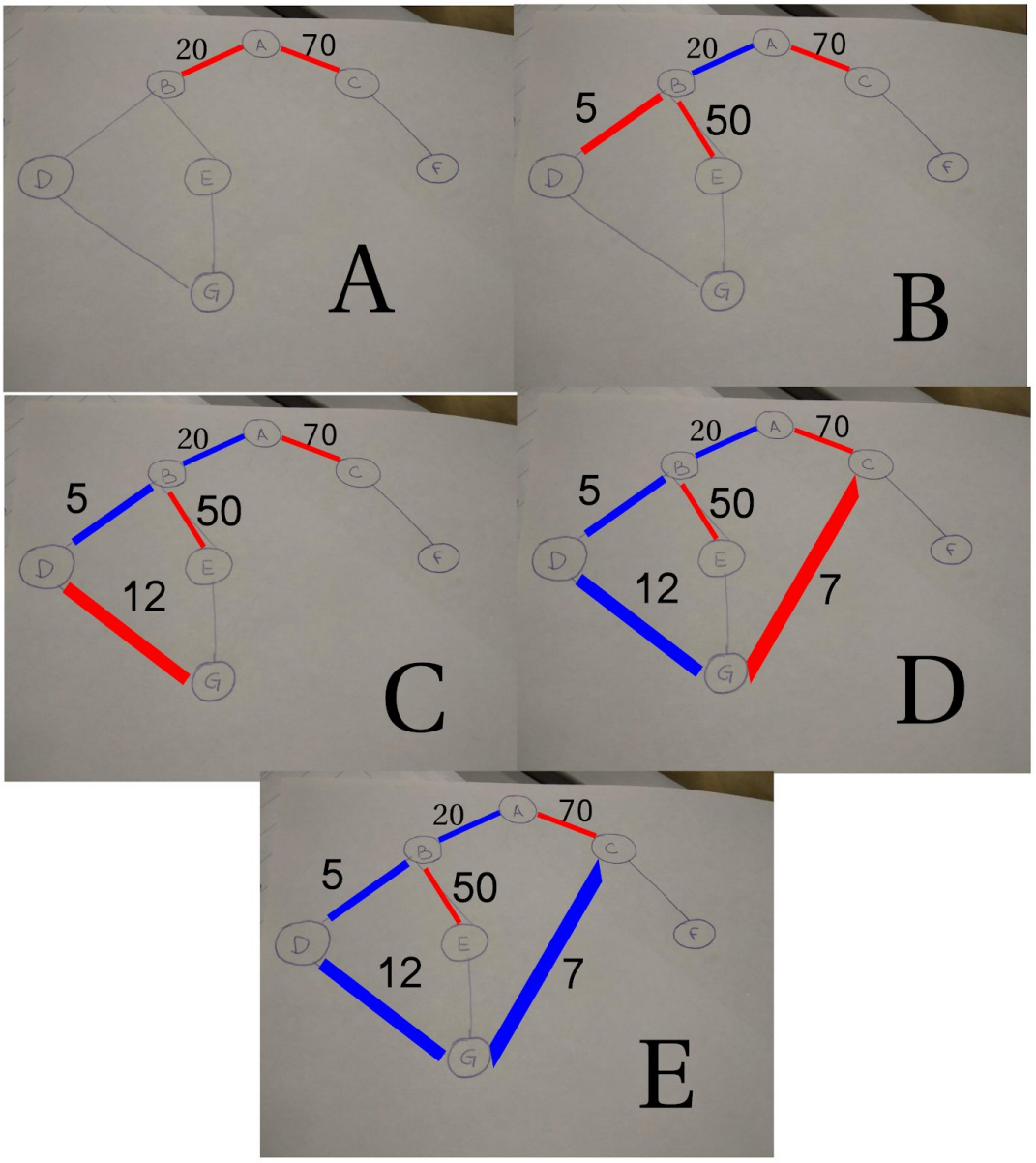
```
        costo = dijkstra.costo[ciudad];
        Si costo < costo_menor{
        costo_menor = costo;
        mejor_dijkstra = dijkstra;
                }

        caminos.agregar(mejor_dijkstra.camino[ciudad]);
        }
retornar caminos;
}
```

Implementamos dijkstra para obtener el mejor camino desde una ciudad hacia todas las obtenidas en una lista de ciudades.

En el método del caminoMasCorto hacemos por cada puerto un dijkstra que retorne el camino más corto a todas las ciudades. Y a partir de estos tres dijkstra calculamos por cada ciudad cual es el puerto más cercano según el que tenga menor costo. Esto se va a retornar en una lista , que incluye los mejores caminos de todas las ciudades a sus puertos más cercanos.

# Seguimiento del algoritmo del Dijkstra




En la figura A se puede observar cómo el algoritmo toma los adyacentes partiendo de mi origen (A) y calcula su costo asociado. Luego opta por el vértice no visitado de menor costo (B) y así sucesivamente hasta encontrar una ruta alternativa a C de costo menor a la encontrada con anterioridad y actualiza la data del camino más corto que en este caso sería {A, B, D , G, C}.