Programación 3 – Ejercicio Entregable TP4
Alumno: Sandoval Fernando

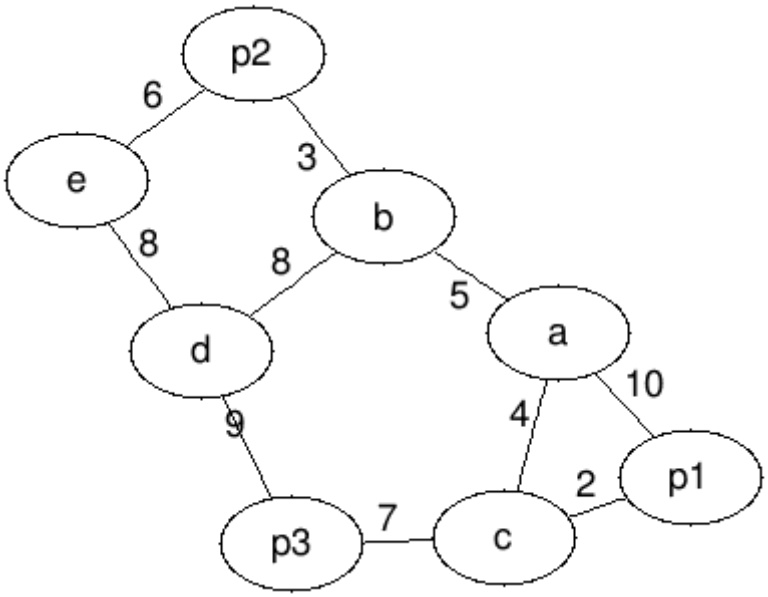
Se nos solicita resolver el ejercicio 5, en el cual debemos, dado un grafo de ciudades y puertos, hallar el camino de menor costo desde cada ciudad hasta el puerto de menor costo.
Para esto vamos a tomar el grafo de ciudades y puertos, y elegiremos un nodo que no sea puerto para comenzar la busqueda. La busqueda se realizara sobre todos los nodos del grafo que no sean puertos, por lo tanto mientras no queden nodos sin visitar, si estos no son puertos, se procederá a ejecutar el algoritmo de selección.
El algoritmo de selección trabaja internamente de forma similar al algoritmo de Dijstkra, o sea que dado un nodo ira buscando el camino de menor costo hacia alguno de sus nodos no visitados, y una vez seleccionado el nodo (que será el que tenga el arco con menor peso), irá construyendo un camino parcial (con el costo más bajo) y luego se trasladará al nuevo nodo, guardando el padre de éste, y actualizando los costos. Luego repite el procedimiento hasta llegar a un nodo final. Al ir seleccionando siempre el camino de menor costo a medida que va construyendo el camino parcial, al finalizar nos aseguramos que el camino obtenido es el de menor costo en todo el grafo. Y luego para obtener el camino recorrido vamos buscando hacia atrás el padre del nodo actual, y luego el padre de este, y asi sucesivamente hasta llegar al nodo inicial.
En este caso particular, el algoritmo debe cortar su busqueda al llegar a un nodo que sea puerto, por lo tanto no hace falta que recorra todos los nodos del grafo como lo hace Dijsktra. Y además, como el algoritmo nos va dando a cada paso el camino parcial de menor costo, al encontrarnos con el primer nodo que sea puerto, ya sabremos que ese puerto será el de menor costo de todo el grafo, ya que desde ahi en adelante no se podrá mejorar el menor costo del camino encontrado hasta ese momento.
Por lo tanto esa será la solución para el nodo con el que iniciamos la busqueda. Luego hay que repetir el procedimiento para cada nodo que no sea puerto y tendremos la solución para todo el grafo.

```
funcion caminos (grafo ciudades){                        //le pasamos el grafo como parámetro
      mientras (no estén visitados todos los nodos){   //recorremos el grafo
           si(nodoActual != puerto){                          // si el nodo no es puerto, entramos
                caminoMínimo = nodoActual.dijsktraModificado();
                                                                  // para cada nodo ejecutamos el
                                                                  // algoritmo Dijkstra modificado que
                                                                  // nos devuelve la solucion para ese nodo
           }
      return ('El camino al puerto de menor costo para '+nodoActual.nombre() +' es '
                +caminoMínimo);
      nodoActual = nodoActual.siguiente();           //paso al siguiente nodo
      }    //fin mientras
}

funcion dijsktraModificado(){
       visitados = {actual.nombre};                 // agrego al arreglo solución el nodo actual
       distancias.inicializar();                      //inicializo arreglo de distancias con los valores
                                                        de los arcos que accedo directamente
       padres.inicializar();                  // tambien inicializo arreglo de padres de cada nodo
       mientras  (tenga nodos sin visitar){
            w = seleccionar();                      //selecciono el vertice con arco de menor costo
            visitados.agregar(w);                  //lo agrego a los visitados
            si w.esPuerto = verdadero   // acá hago un corte si llegué a un puerto
              return devolverSolución();              // y retorno el camino recorrido hasta ahora
               sino
                     para (cada vertice v adyacente a w que no esta en solucion){
                           si distancias[w]  + distanciaEntre(w,v) < distancias [v]{
                           distancias[w] = min (distancias[v], distancias[w]  +
                                                  distanciaEntre(w,v));
                           // si la nueva distancia obtenida es menor a la que esta en el arreglo
                           // de distancias, actualizo el arreglo de distancias con el nuevo valor
                           padres[v] = w                  //actualizo el arreglo de padres
                            } // fin para

       }// fin mientras
}
```

La modificación en el Dijsktra está en que éste recorre cada nodo del grafo hasta visitarlo en su totalidad para retornar la solución. Aquí tenemos un corte en caso de haber llegado al primer puerto, ya que como dijimos anteriormente, este será el de menor costo de todo el grafo y por lo tanto, la solución. La funcion devolverSolucion (no implementada) recorre el arreglo de padres a la inversa, retornando como primer elemento el nodo final, luego el padre de este, y luego el padre de este, hasta llegar al nodo inicial que no tiene padre.

Ahora se realizará un seguimiento del algoritmo sobre un grafo cuyas ciudades son letras desde A hasta E, y los 3 puertos son p1, p2 y p3

Grafo de Ciudades y Puertos



La funcion caminos simplemente recorre el grafo buscando nodos que no sean puertos. Si iniciamos el recorrido del grafo en el nodo p1, este se descarta y se pasa al nodo A. Luego ahora si usando el nodo A entramos a la funcion Dijsktra Modificado. Este agrega A al arreglo solución, y calcula las distancias directas hacia los nodos adyacentes para ponerlas en el arreglo Distancias, quedando de la siguiente manera:

A	B	C	D	E	p1	p2	p3
-	5	4	Inf	Inf	10	Inf	inf

Para los nodos que no son adyacentes, la distancia se pone en infinito, para que al momento de ser descubiertas durante el recorrido del grafo, puedan actualizarse comparandolo con el valor del vertice. Mientras tanto el arreglo de padres queda de la siguiente manera

A	B	C	D	E	p1	p2	p3
-	A	A	-	-	A	-	-

Luego pasamos a la segunda iteración. En este caso elegimos el nodo no visitado con menor peso en el arreglo de distancias, que es el nodo C.

Actualizamos los valores en el arreglo de distancias quedandonos de la siguiente manera

A	B	C	D	E	p1	p2	p3
-	5	4	inf	Inf	6	inf	11

Y el arreglo padres nos queda asi

A	B	C	D	E	p1	p2	p3
-	A	A	-	-	C	-	C

Luego pasamos a la tercer iteración. En este caso elegimos el siguiente nodo no visitado con menor peso en el arreglo de distancias, que es el nodo B. Actualizamos los valores en el arreglo de distancias quedandonos de la siguiente manera

A	B	C	D	E	p1	p2	p3
-	5	4	13	Inf	6	8	7

Y el arreglo padres ahora nos queda asi

A	B	C	D	E	p1	p2	p3
-	A	A	B	-	C	B	C

Por ultimo, llegamos al siguiente nodo no visitado con menor peso que es el nodo p1. Aqui el algoritmo detecta que se ha llegado a un nodo que es puerto y termina la ejecución del recorrido del grafo. Luego llama a la funcion devolverSolucion que nos retorna el camino desde el nodo inicial hasta el final, recorriendo en forma inversa el camino que se genera recorriendo el arreglo de padres (Comienza agregando el nodo final, luego busca cual es su padre y lo agrega, y asi para cada nodo hasta llegar al primero).

En este caso el recorrido del arreglo de padres nos da la secuencia

p1 – c – a

la cual si se recorre de forma inversa nos da el camino buscado.