Tercer Entrega Grafos
Miércoles 23/11- MT

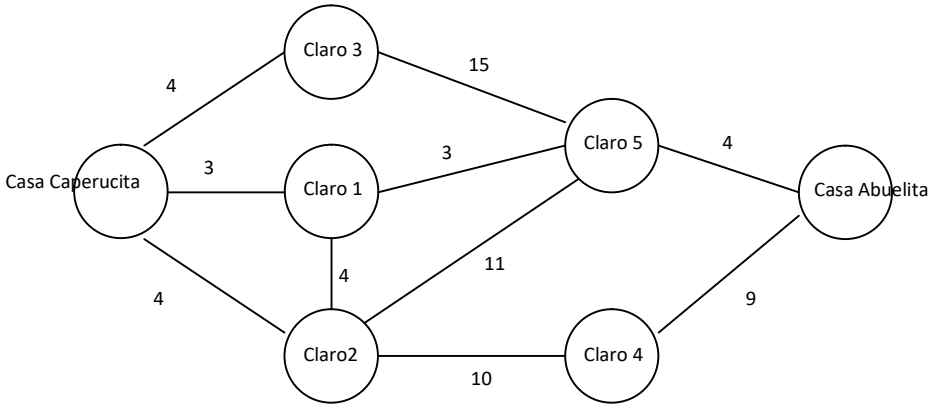
Un día, Caperucita Roja decide ir desde su casa hasta la de su abuelita, recolectando frutos del bosque del camino y tratando de hacer el paseo de la manera más segura posible. La casa de Caperucita está en un claro del extremo oeste del bosque, la casa de su abuelita en un claro del extremo este, y dentro del bosque entre ambas hay algunos otros claros.
El bosque está representado por un grafo, donde los vértices representan los claros (identificados por un String) y las aristas los senderos que los unen. Cada arista informala cantidad de árboles frutalesque hay en el sendero. Caperucita sabe que el lobo es muy goloso y le gustan mucho las frutas, entonces para no ser capturada por el lobo, desea encontrar un camino que no pase por los senderos con cantidad de frutales >= 5. Como miembros de la sociedad Ayuda a Caperucita a Moverse (ACM), están llamados a ayudarla a encontrar ese camino, y cuando lo hayan hecho, colorín, colorado, este problema habrá terminado!

Su tarea es definir una clase llamada **BuscadorDeCamino**, con una variable de instancia llamada **"bosque"** de tipo **Grafo**, que representa el bosque descripto e implementar un método de instancia con la siguiente firma:

publicListaGenerica<String>recorridoMasSeguro()

que devuelva un listado con los claros que conforman el recorrido encontrado**.** Si existe más de uno, devuelva uno de ellos.Si no existe devuelva un listado vacío.

Nota: La casa de caperucita debe ser buscada antes de comenzar a buscar el recorrido, y tiene la identificación: "Casa Caperucita".



Caminos posibles:

1) Casa Caperucita- Claro 1 -Claro 5-Casa Abuelita.
2) Casa Caperucita- Claro 2 – Claro 1 - Claro 5-Casa Abuelita.

Entonces, el método podría devolver la opción 1 o la opción 2. (Es indiferente)

```
public class BuscadorDeCaminos{

        privateGrafo<String>bosque;

        publicListaGenerica<String>recorridoMasSeguro(){

                ListaGenerica<String>lis=new ListaGenericaEnlazada<String>();
                boolean [] marca= new Boolean [bosque.listaDeVertices().tamanio()];
                Vertice<String> v;

                // Búsqueda de la casa de Caperucita Roja
                v = buscarCasaCaperucita();

                dfs(v.posicion(), lis, marca);
                returnlis;
        }

        privatebooleandfs(int posV;ListaGenerica<String>lis; boolean[] marca){

        boolean encontre ;
        marca[posV] = true;
        Vertice<String> vert  = bosque.vértice(posV);
        lis.agregar(vert.dato(), lis.tamanio());

        if (vert.dato().equals ("Casa Abuelita"))

                encontre = true;

        else{
                ListaGenerica<Arista<int>>ady= bosque.listaDeAdyacentes(vert);
                ady.comenzar();

                while( !ady.fin()&& !encontre){
                    Arista  a = ady.proximo();
                    int  j = a.verticeDestino().posicion();
                    if (! marca [j])
                        if(a.peso() <5)
                            encontre  = dfs (j, list, marca);
                }
                If ( !encontre)
                    lis.eliminarEn(lis.tamanio());
        }
        return encontre;
        }

        private  Vertice<String> buscarCasaCaperucita(){

                ListaGenerica<Vertice<String> >vertices  =  bosque.listaDeVertices();

                vertices.comenzar();
                while(!vertices.fin()){

                        Vertice<String>  v = vertices.proximo();
                        If (v.dato().equals ("Casa Caperucita"))
                                return  v;
                }
        }
}
```