Justificacions – Ferran Blanchart Reyes

Assignar estació

```
      void Estaciones::assig_est_aux(const
BinTree<string>&ids,int&ests,int&libres,double&max_desocup,string&ide) {

    if (ids.empty()) {
        ests = 0;
        libres = 0;
    }
    else {
        int dre,izq,lib_dre,lib_izq;
        assig_est_aux(ids.left(),izq,lib_izq,max_desocup,ide);
        assig_est_aux(ids.right(),dre,lib_dre,max_desocup,ide);
        //actualitzar en nombre d'estacions recorregudes i el nombre de
places lliures totals i el coeficient de desocupacio actual
        if (desocup > max_desocup) {
            ide = ids.value();
            max_desocup = desocup;
        }
        else if (desocup == max_desocup) {
            if (ide == " " or ids.value() < ide) ide = ids.value();
        }
    }
}

void Estaciones::assig_est(string& ide) {
    double max_desocup = 0;
    string ide_act = "";
    int ests,libres;
    assig_est_aux(ids_estaciones,ests,libres,max_desocup,ide_act);
    ide = ide_act;
}
```

- En el meu cas utilitzo immersió ja que tinc una funció que realitza el càlcul del coeficient de desocupació y determina quina és la estació correcta guardant el seu id a una variable de les que es passen com a paràmetre. La altre funció inicialitza els valors que s'utilitzen a la funció recursiva i crida a aquesta mateixa.
- Justificació:
  →Cas senzill: si el node de l'arbre està buit això vol dir que no hi ha estacions i per tant, hi ha zero places lliures per sota d'aquell node.
  →Cas recursiu: si el node no està buit, la funció realitza crides recursives a ella mateixa per al arbre dret i al arbre esquerra. Amb a informació obtinguda després de la recursivitat, calcula el nombre d'estacions per les que s'ha passat i el nombre total de places lliures d'aquestes. Després d'això calcula el coeficient de desocupació i el

comparà amb el màxim fins al moment i actualitza la estació amb major coeficient si es necessari.
→Decreixement: a cada crida recursiva la funció treballa amb un subarbre més petit.

Pujar bicis

```
      auto iv = estaciones.find(valor);

        auto ie = estaciones.find(ids.left().value());
        auto id = estaciones.find(ids.right().value());

        int bicis_disponibles = ie->second.ocupacion() + id->
second.ocupacion();

        while (iv->second.sitios_libres() > 0 && bicis_disponibles > 0) {
            if (estació Esquerra més bicis que dreta) {
                realizar operacions necesaries
            }
            else if (estació dreta més bicis que la Esquerra) {
                realizar operacions necesaries
            }
            else {(estacions amb el mateix nombre de bicis)
               if {(primera bici de la estacio Esquerra amb l'id més
petit que la estació de la dreta) {
                   realizar operacions necesaries
                }
                else {(primera bici de la estacio dreta amb l'id més
petit que la estació de la esuqerra)
                  realizar operacions necesaries
                }
            --bicis_disponibles;
        }
```

- Justificació:
  →Inicialitzacions: inicialment no hem pujat cap bici de les 2 estacions següents, per tant, he inicialitzat la variable "bicis_disponibles" amb la suma del nombre de bicis de la estació esquerra i dreta satisfent així la primera part de l'invariant. Per satisfer la segona part, he fet la comprovació de si la estació a la que estic te llocs lliures, ja que si no en te no se li poden posar més bicis; això ho faig comprovant si el nombre de llocs lliures de la estació es > 0.
  →Condició de sortida: poden haver dos. La primera es que la estació a la que li estem posant bicis estigui plena, això voldria dir que l'invariant indicaria que el nombre de bicis de la estació és = 0 i per tant, surt del bucle. La segona condició de sortida es que no hi hagin bicis, això voldria dir que "bicis_disponibles" és = 0 i surt del bucle.
  →Cos del bucle: es comprova quina de les dos estació té més bicis o si son iguals. Si la esquerra te més es treu una bici de la estació es esquerra i se li adjunta a la estació actual. Segueix el mateix esquema en cas que la dreta tingués més bicis. En cas que tinguin el mateix nombre de bicis mira quina de les primeres bicis de les dos estacions

te l'id més petit. Depenent di quina la tingui es segueix els esquemes mencionats anteriorment.
→Acabament: Al final de cada iteració es decrementa en una unitat la variable "bicis_disponibles".