Mateus Grandolfi Albuquerque, grup 43

# Justificacions de mètodes: Pràctica de PRO2 (*Bicing Bifurcado*)

**Justificació del mètode recursiu (asignar_estacion):**

```
asignar_estacion:
    //pre El parámetro implícito está inicializado
    //post Se ha añadido la bici <em>idb</em> al parámetro implícito
    if (a.left().empty() and a.right().empty()) {
        pl = ce[a.value()].plazas_libres();
        num_est = 1;
        if ((coef_max < pl or (coef_max == pl and ide > a.value()))) {
            coef_max = pl;
            ide = a.value();
        }
    }
    else {
        int est_izq, est_der, pl_izq, pl_der;
        i_asignar_estacion(a.left(), ide, coef_max, est_izq, pl_izq);
        i_asignar_estacion(a.right(), ide, coef_max, est_der, pl_der);
        pl = pl_izq + pl_der + ce[a.value()].plazas_libres();
        num_est = est_izq + est_der + 1;
        double coef_max_nuevo = double(pl)/double(num_est);
        if ((coef_max < coef_max_nuevo or (coef_max == coef_max_nuevo and ide > a.value())) and ce[a.value()].plazas_libres() > 0) {
            coef_max = coef_max_nuevo;
            ide = a.value();
        }
    }
```

-Hipòtesi d'inducció:

H.I. (1): S'ha calculat recursivament al subarbre esquerre quina és l'estació amb el coeficient de desocupació més gran. Si aquest és el coeficient més gran de tot l'arbre, s'ha assignat aquesta estació a la bici nova.

H.I. (2): S'ha calculat recursivament al subarbre dret quina és l'estació amb el coeficient de desocupació més gran. Si aquest és el coeficient més gran de tot l'arbre, s'ha assignat aquesta estació a la bici nova.

-Justificació:

-Cas base: Quan l'arbre no té fills i només té arrel, si l'estació arrel té espai, s'assigna aquesta estació per a la bici nova.

-Cas recursiu: Cridem recursivament a *a.left()* y *a.right()*, recalculant el nou nombre de places lliures entre totes les estacions inferiors i el nombre d'estacions que queden per analitzar. Si el nou coeficient és major o igual a l'actual, l'identificador d'estació és més gran que el node pare actual i aquest node té places lliures, el coeficient actual passa a ser el coeficient nou i la nova millor estació per assignar passa a ser aquest node.

-Acabament: A cada crida recursiva l'arbre es fa més petit i queden menys nodes per analitzar.

**Justificació del mètode iteratiu (subir_bicis):**

```
subir_bicis:
        //pre El parámetro implícito está inicializado
        //post Se han subido a estaciones superiores del árbol todas las bicis posibles
        map<string, Estacion>::iterator it = ce.find(a.value());
        map<string, Estacion>::iterator izq = ce.find(a.left().value());
        map<string, Estacion>::iterator der = ce.find(a.right().value());
        while (not (*it).second.estacion_llena() and ((*izq).second.cantidad_bicis() != 0 or (*der).second.cantidad_bicis() != 0)) {
            if ((*izq).second.cantidad_bicis() > (*der).second.cantidad_bicis()) {
                string idb = (*izq).second.bici_menor();
                baja_bici(idb, a.left().value());
                alta_bici(idb, a.value());
                b.modificar_estacion(idb, a.value());
            }
            else if ((*izq).second.cantidad_bicis() < (*der).second.cantidad_bicis()) {
                string idb = (*der).second.bici_menor();
                baja_bici(idb, a.right().value());
                alta_bici(idb, a.value());
                b.modificar_estacion(idb, a.value());
            }
            else {
                string bici_izq = (*izq).second.bici_menor();
                string bici_der = (*der).second.bici_menor();
                if (bici_izq < bici_der) {
                    string idb = (*izq).second.bici_menor();
                    baja_bici(idb, a.left().value());
                    alta_bici(idb, a.value());
                    b.modificar_estacion(idb, a.value());
                }
                else {
                    string idb = (*der).second.bici_menor();
                    baja_bici(idb, a.right().value());
                    alta_bici(idb, a.value());
                    b.modificar_estacion(idb, a.value());
                }
            }
        }
```

-Invariant del bucle: El nombre d'estacions (exactament tres: l'arrel, el node esquerre i el node dret), la capacitat d'aquestes tres estacions, el nombre de bicis totals entre aquestes tres estacions i els iteradors *it*, *izq* i *der*.

-Justificació:

-Inicialització: Necessitem tres iteradors que farem servir dins del bucle, un que apunta a l'estació arrel, un que apunta al fill esquerre i un altre que apunta al fill dret. Com que encara no sabem on són ni quantes són les bicis que hem de pujar, necessitem aquests iteradors per moure'ns per les tres estacions amb la finalitat de saber diferents atributs d'aquestes que ens interessen.

-Condició de sortida: Com que la postcondició diu que s'han pujat totes les bicis possibles, només sortirem del bucle si l'estació arrel (*a.value()*) ha arribat a la seva capacitat màxima o si tant l'estació esquerra com la dreta estan buides.

-Cos del bucle: Tenim tres casos principals. El primer és quan el nombre de bicis de l'estació esquerra és més gran que el nombre de bicis de l'estació dreta. En aquest cas, *idb* puja a *a.value()*. L'estació esquerra passa a tenir una bici menys i l'estació arrel passa a tenir una més. El segon cas és igual al primer, però en comptes de que l'estació esquerra, s'escull l'estació dreta. L'últim cas és quan el nombre de bicis és

igual a les dues estacions filles. Aqui, s'ha de veure a les dues estacions quina és la que conté la bici amb l'identificador d'ordre més petit. Si és a l'estació esquerra, s'elimina la bici d'aquesta estació i puja a *a.value()*. Si és a l'estació dreta, s'elimina la bici d'aquesta estació i puja a *a.value()*.

-Acabament: A cada volta decreix tant el nombre de bicis per pujar com les places lliures de l'estació arrel.