Arnau Hernández Navarro Grup 12 PRO2

Justificació d'operacions de la pràctica de PRO2

-Función Subir Bicis

```
void Cjt_estacions::Reestructurar_ubicacion(const BinTree<string>& b)
    /* Pre: esp_actual >= 0 y bicis_izq >= 0 y bicis_drc >= 0. */
    /* Post: Las bicis han sido reestructuradas en direccion hacia la primera estacion declarada. */
{
    if(b.empty()) return;
    if(b.left().empty()) return;
    string estacion_actual = b.value();
    string estacion_izq = b.left().value();
    string estacion_drc = b.right().value();

    int esp_actual = estaciones[estacion_actual].espacio_libre_estacion();
    int bicis_izq = estaciones[estacion_izq].cuantas_bicis();
    int bicis_drc = estaciones[estacion_drc].cuantas_bicis();

    bool stop = (bicis_izq == 0 and bicis_drc == 0);

    /*Inv:  1. La variable stop es false al comienzo del bucle si bicis_izq y bicis_drc son >=0, si ambos son 0, stop = true.
            2. La variable esp_actual mayor o igual a 0 al comienzo del bucle.
    */

    while((not(stop)) and (esp_actual != 0)){
        if(bicis_izq > bicis_drc){
            mover_bici_sin_viaje(estaciones[estacion_izq].consultar_bici(), estacion_actual);
            --bicis_izq;
        }
        else if(bicis_drc > bicis_izq){
            mover_bici_sin_viaje(estaciones[estacion_drc].consultar_bici(), estacion_actual);
            --bicis_drc;
        }
        else{
            string id_bici_izq = estaciones[estacion_izq].consultar_bici();
            string id_bici_drc = estaciones[estacion_drc].consultar_bici();
            if(id_bici_izq < id_bici_drc){
                mover_bici_sin_viaje(id_bici_izq, estacion_actual);
                --bicis_izq;
            }
            else{
                mover_bici_sin_viaje(id_bici_drc, estacion_actual);
                --bicis_drc;
            }
        }
        if(bicis_izq == 0 and bicis_drc == 0){
            stop = true;
        }
        --esp_actual;
    }
    Reestructurar_ubicacion(b.left());
    Reestructurar_ubicacion(b.right());
}
```

· *Inicialitzacions:*

-Inicialment si un dels dos valors de bicis_izq o bicis_drc es mayor de 0, es complirà la primera condició de l'invariant.

-Esp_actual es un valor que es dona trucant a la funció espacio_libre_estacion, i si aquesta retorna un valor > 0, es complirà la segona condició de l'invariant. Mai serà < 0 ja que una estructura de dades mai pot tenir espai negatiu, complint així la segona condició de l'invariant.

· *Condició de sortida:* Es pot sortir del bucle per dos motius:

-Si STOP = true, que per invariant, significa que bicis_izq y bicis_drc = 0.

-Si esp_actual = 0, que per invariant, no tenim espai a l'estació actual per moure bicis.

· *Cos del bucle:*

Abans de la següent iteració, hem de satisfer l'invariant. Si hem entrat al bucle vol dir que stop = false (bicis_izq o bicis_drc un dels dos no es 0) y esp_actual != 0 i per invariant significa que tenim almenys un espai entre bicis_izq i bicis_drc, i esp_actual > 0. Una vegada executades les instruccions, es decrementa en 1 el valor de esp_actual que aquest sempre serà >= 0 després de l'instruccions, cumplint la condició de l'invariant i també es decrementa en 1 bicis_izq o bicis_drc depenent de les condicions dels if que fan l'activitat de la funció i aquests seràn >= 0 tornant a complir la invariant.

· *Acabament:*

A cada volta es decrementa el valor de esp_actual i bicis_izq o bicis_drc depenent de les condicions que demani la funció per fer la seva activitat.

-Funció auxiliar de Asignar Estació

```
void Cjt_estacions::coeficiente(const BinTree<string>& b, string& id_estacion, double& max, int& el_b, int& hijos_b) {
    /* Pre: cierto */
    /* Post: el resultado es la estacion con mayor coeficiente.*/
    if(b.empty()) return;
    int el_izq, el_drc, hijos_izq, hijos_drc;
    el_izq = el_drc = hijos_izq = hijos_drc = 0;

    coeficiente(b.left(), id_estacion, max, el_izq, hijos_izq);
    coeficiente(b.right(), id_estacion, max, el_drc, hijos_drc);
    /* HI: La funcion coeficiente opera al calcular los coeficientes de la estacion raiz y de sus subarboles
           izquierdo y derecho y selecciona la estacion con mayor coeficiente.
    */
    el_b = estaciones[b.value()].espacio_libre_estacion() + el_izq + el_drc;
    hijos_b = 1 + hijos_izq + hijos_drc;

    if(hijos_izq == 0 or hijos_drc == 0) return;

    double coef_b = el_b / double(hijos_b);
    double coef_izq = el_izq / double(hijos_izq);
    double coef_drc = el_drc / double(hijos_drc);

    if(coef_izq > max) {
        max = coef_izq;
        id_estacion = b.left().value();
    }
    else if(coef_izq == max) {
        if(b.left().value() < id_estacion) {
            max = coef_izq;
            id_estacion = b.left().value();
        }
    }

    if(coef_drc > max) {
        max = coef_drc;
        id_estacion = b.right().value();
    }
    else if(coef_drc == max) {
        if(b.right().value() < id_estacion) {
            max = coef_drc;
            id_estacion = b.right().value();
        }
    }

    if(coef_b > max) {
        max = coef_b;
        id_estacion = b.value();
    }
    else if(coef_b == max) {
        if(b.value() < id_estacion) {
            max = coef_b;
            id_estacion = b.value();
        }
    }
}
```

· *Cas senzill:*
Si l'arbre és buit, té zero elements i es retorna id_estacion sense identificador i de coeficient 0.

· *Cas recursiu:*
Si l'arbre no és buit, poden existir o no els seus subarbres esquerre i dret. Es faran les trucades i en cas de que no existeixin el left i el right, es farà return; de la funció.
La funció retornarà el paràmetre id_estacion actualitzat seleccionat a la funció en funció del major coeficient entre l'arrel i els seus subarbres esquerre i dret.

Els coeficients dels subarbres s'obtenen per hipòtesi d'inducció amb les crides recursives, que són correctes perquè els paràmetres s'obtenen correctament i compleixen amb la precondició. Només cal fer les comprobacions per seleccionar l'estació amb major coeficient i el programa queda complet.

· *Decreixement*:
Cada crida recursiva, en cas de que existeixin subarbres esquerre i dret, es farà cada vegada amb un arbre més petit.