Roger Bitlloch Galceran

La següent funció, “**busca_estacion**” és l’operació auxiliar encarregada d’escollir a quina estació se li assigna la nova bicicleta.

- **Especificació:**
  @brief Busca l'estació amb una major desviació estàndard quant a places lliures.
  @pre Cert
  @post Retorna l'identificador de l'estació amb major coeficient de desocupació ("error" si l'arbre és buit)
  “des” conté el coeficient de desocupació de l’estació amb major desocupació
  “pl” conté el nombre de places lliures de la branca
  “n” conté el nombre de nodes de la branca
  @cost lineal respecte al nombre total d'estacions en l'arbre.

- **Implementació:**

```
string Cjt_estaciones::busca_estacion(const BinTree<string>& arbre, float& des, float& pl, float& n) {
    //CAS SENZILL 1
    if (arbre.empty()) {
        des = -1;
        return "error";
    }
    //CAS SENZILL 2
    if (arbre.left().empty() && arbre.right().empty()) {
        n = 1;
        des = pl = mapa_estaciones[arbre.value()].consultar_espais_lliures();
        return arbre.value();
    }

    //CAS RECURSIU
    float des_left, des_right, pl_left, pl_right, n_left, n_right;
    pl_left = pl_right = n_left = n_right = 0;
    string id_left = busca_estacion(arbre.left(), des_left, pl_left, n_left);
    string id_right = busca_estacion(arbre.right(), des_right, pl_right, n_right);
    pl = pl_left + pl_right + mapa_estaciones[arbre.value()].consultar_espais_lliures();
    n = n_left + n_right + 1;
    des = pl / n;
    if (des_left > des_right || (des_left == des_right && id_left < id_right)) {
        if (des_left > des || (des_left == des && id_left < arbre.value())) {
            des = des_left;
            return id_left;
        }
        else return arbre.value();
    }
    else if (des_right > des || (des_right == des && id_right < arbre.value())) {
            des = des_right;
            return id_right;
    }
    else return arbre.value();
}
```

- **Hipòtesi d'Inducció (HI):**
  Suposem que per a un valor de n qualsevol (on n representa la mida de l'arbre en aquella crida específica), la funció busca_estacion compleix que:
  - retorna l’ID de l’estació amb el grau de desocupació més alt.
  - des = El grau de desocupació més alt.
  - pl = suma places lliures tots els nodes
  - n = nombre de nodes

Roger Bitlloch Galceran

La justificació informal d'aquesta funció recursiva és la següent:

- **Cas senzill 1:**
  Si l'arbre és buit, significa que ja no hi ha cap estació disponible per a la redistribució de bicicletes.
  Per tant, es retorna "error" indicant que ja no es poden assignar més bicicletes.

- **Cas senzill 2:**
  Quan l'arbre és una fulla (no té fills), s'identifica com una estació simple sense subestacions. En aquest cas, es calculen les variables des, pl, i n amb les dades de l'estació actual, i es retorna l'ID d'aquesta estació, ja que és l'única disponible, per tant, té el grau de desocupació més gran.

- **Cas Recursiu:**
  Si l'arbre té fills (subarbres esquerra i dret), la funció es crida recursivament en aquests subarbres per calcular les desocupacions, places lliures i nombre d'estacions de cada branca.
  Després de les crides recursives, es calcula la desocupació, places lliures i nombre d'estacions de l'estació actual, i es compara amb els resultats de les branques esquerra i dreta.

  Es selecciona l'ID de l'estació amb el grau de desocupació més gran entre l'estació actual, i l'estació amb major grau de desocupació de les branques esquerra i dreta. Finalment, s'actualitza el coeficient de desocupació màxim (des) i el nombre de places lliures i es retorna l'ID de l'estació seleccionada, indicant la millor opció per assignar la nova bicicleta.

- **Creixement:**
  A cada crida recursiva, s'incrementa el nombre d'estacions (n) i es suma el nombre de places lliures de l'estació actual (pl) a la suma de les places lliures de cada subarbre.

Roger Bitlloch Galceran

La següent funció, “**llenar_estaciones**” és l’operació auxiliar encarregada de recol·locar les bicicletes entre les estacions.

- **Especificació:**
  @brief Omple les estacions amb les bicicletes disponibles segons un arbre binari.
  @pre Cert
  @post Les estacions del sistema (`mapa_estaciones`) s'omplen amb bicicletes disponibles segons l'arbre donat i el cjt_Bicis bicicletes s'actualitza amb els canvis d'estacions.
  @cost lineal en la quantitat total d'estacions i bicicletes a moure.,

- **Implementació:**

```
void Cjt_estaciones::llenar_estaciones(Cjt_Bicis& bicicletes, BinTree<string> arbre) {
    if (!arbre.left().empty() && !arbre.right().empty()) {
        string left = arbre.left().value();
        string right = arbre.right().value();
        int ocupats_left = mapa_estaciones[left].consultar_espais_ocupats();
        int ocupats_right = mapa_estaciones[right].consultar_espais_ocupats();
        while (mapa_estaciones[arbre.value()].consultar_espais_lliures() > 0 && ocupats_left + ocupats_right != 0) {
            if (ocupats_left > ocupats_right) {
                pujo_bici(bicicletes, left, arbre.value());
            }
            else if (ocupats_left < ocupats_right) {
                pujo_bici(bicicletes, right, arbre.value());
            }
            else if (mapa_estaciones[left].consultar_id_primera() < mapa_estaciones[right].consultar_id_primera()) {
                pujo_bici(bicicletes, left, arbre.value());
            }
            else pujo_bici(bicicletes, right, arbre.value());
            ocupats_left = mapa_estaciones[left].consultar_espais_ocupats();
            ocupats_right = mapa_estaciones[right].consultar_espais_ocupats();
        }
        llenar_estaciones(bicicletes, arbre.left());
        llenar_estaciones(bicicletes, arbre.right());
    }
}
```

- **Inicialitzacions:**
  Comprovem que el node actual té estacions inferiors (no és fulla) i mirem quantes bicis tenen cadascuna d’aquestes estacions.

- **Condició de sortida:**
  Es pot sortir del bucle per dues raons:
  - Que no hi hagi espais lliures a l’estació actual.
  - Que no hi hagi bicis a les estacions directament inferiors.

- **Cos del Bucle:**
  Dins del bucle, es fa una sèrie de comparacions per determinar quina branca té més bicicletes i es crida la funció pujo_bici per moure una bicicleta des de la branca amb més bicicletes a l'estació actual.

  Després de cada crida a pujo_bici, s'actualitzen les variables ocupats_left i ocupats_right amb la quantitat d'espais ocupats a les estacions esquerra i dreta, respectivament.

- **Acabament:**
  És crida la funció de forma recursiva amb els subarbres esquerre i dret de l'arbre original, continuant així amb la cerca i la redistribució de bicicletes a través de l'estructura jeràrquica de les estacions representada pel BinTree.