BST Augmentat — T56142_ca

Volem fer una classe **`BST_Augmented`** tal que a cada node (instància de **`_Node`**) del BST es guardi la mida del subarbre del que aquest node és arrel. Si el node és una fulla, aquesta mida és 1, ja que l'arrel s'inclou en aquest comptatge. En un node qualsevol, aquest nombre és 1 més la mida del fill esquerre més la mida del fill dret. És el concepte de *mida d'arbre* que tots ja coneixem.
A més, les instàncies d'aquests **`BST_Augmented`** només creixeran, no caldrà considerar l'eliminació d'elements. Així, en lloc de fer una subclasse de **`BST`**, definirem la classe **`BST_Augmented`** de nou (inspirant-nos, això sí, en la classe **`BST`** que ja coneixem), on afegirem una variable d'instància **`_size`** a la classe interna **`_Node`**, i només modificarem els mètodes **`afegeix`** i **`_afegeix`** (hem tret els mètodes relacionats amb l'eliminació d'elements, i altres que no serveixen en aquest exercici). El mètode **`_inordre`** també ha patit una molt lleugera modificació. Vegeu el fitxer **`code.py`**.
L'únic que queda pendent en el fitxer **`code.py`** per acabar d'implementar la classe **`BST_Augmented`** és adaptar el mètode **`_afegeix`** als canvis que hem fet. És a dir, el mètode **`_afegeix`** haurà de tenir en compte que afegir un node canvia la mida de cada instància de **`_Node`** present en el camí de l'arrel del BST fins al lloc on hem afegit la nova instància de **`_Node`**.
La resta del codi que hi ha a **`code.py`** *no s'ha de tocar!!*. Ja està bé com està.
Així, resumint, l'exercici consisteix en adaptar el mètode **`_afegeix`** de la classe **`BST_Augmented`** als canvis especificats més amunt, dins el fitxer **`code.py`**.

Entrada
L'entrada al programa serà: un nombre `n` i una llista d'`n` nombres enters diferents amb els que construirem el **`BST_Augmented`**.
Vegeu els exemples del joc de proves públic.
Sortida
La sortida són una llista de parelles `(valor node, mida subarbre)` ordenada segons el valor de cada node. Això és el resultat d'invocar el mètode **`elements`**, que ja està fet, com podeu veure al fitxer **`code.py`**.
Vegeu els exemples del joc de proves públic.
Observacions
Heu de baixar-vos el fitxer **`code.py`** (icona de la serp). Aquest fitxer és un programa amb **tot** el que cal per executar els jocs de prova públics. Només falta, clar, que modifiqueu el mètode que demana l'enunciat. Aquest fitxer l'heu de completar amb el vostre codi, i això, **tot**, és el que heu d'enviar al Jutge com a solució.

L'eficiència i la qualitat de la solució es tindran en compte a la correcció manual.

Exemple d'entrada 1

```
18
679 775 999 938 619 908 623 752 624 980 348 567 569 634 667 828 638 604
```

Exemple de sortida 1

```
[(348, 4), (567, 3), (569, 2), (604, 1), (619, 10), (62
```

Exemple d'entrada 2

```
14
993 545 36 777 362 269 367 431 926 150 759 440 733 126
```

Exemple de sortida 2

```
[(36, 8), (126, 1), (150, 2), (269, 3), (362, 7), (367,
```

Exemple d'entrada 3

```
15
450 483 899 805 38 871 936 457 616 780 881 532 118 248 221
```

Exemple de sortida 3

```
[(38, 4), (118, 3), (221, 1), (248, 2), (450, 15), (457
```

Exemple d'entrada 4

```
6
609 8 427 54 407 346
```

Exemple de sortida 4

```
[(8, 5), (54, 3), (346, 1), (407, 2), (427, 4), (609, 6
```

Exemple d'entrada 5

```
15
450 255 642 323 351 456 523 939 590 495 688 949 725 382 959
```

Exemple de sortida 5

```
[(255, 4), (323, 3), (351, 2), (382, 1), (450, 15), (45
```

Informació del problema

Autoria: Jordi Delgado

Generació: 2026-06-27T11:59:31.638Z