Lowest Common Ancestor S30000_ca

Donat un BST **a** (no buit, que conté nombres enters com a elements, i tots són diferents) i dos nombres enters **p** i **q** tal que:

- són elements d'**a**
- són diferents

volem trobar el *Lowest Common Ancestor* (LCA, en català seria el "node més baix comú") de **p** i **q** dins l'arbre **a**.
L'LCA de **p** i **q**, respecte d'**a**, es defineix com el node més baix de l'arbre **a** que té tant **p** com **q** com a descendents. Cal tenir en compte que un node es pot considerar descendent de sí mateix, és a dir, si un dels nodes **p** o **q** és un avantpassat de l'altre, aleshores aquest node avantpassat seria el LCA.
Escriviu **dues** funcions que calculin l'LCA de **p** i **q**, respecte d'**a**. Una d'elles (**`lca_recur(a,p,q)`**) ho farà recursivament i l'altre (**`lca_iter(a,p,q)`**) iterativament.

Paràmetres i retorn de la funció demanada
Els paràmetres de les dues funcions són: un arbre **a** que és un BST no buit amb nombres enters, tots diferents, com a elements, dos enters diferents **p** i **q**, que són elements del BST **a**. El retorn de totes dues funcions és un nombre enter.
Entrada
L'entrada al programa serà: un nombre `n` i una llista d'`n` nombres enters diferents amb els que construirem el BST.
Després venen dos nombres enters diferents (tots dos han d'estar en la llista amb la que acabem de construir el BST).
Vegeu els exemples del joc de proves públic.
Sortida
La sortida han de ser dos nombres enters, separats per un espai. Tots dos haurien de ser iguals. Un d'ells és l'LCA de l'entrada calculat recursivament, i l'altre és l'LCA de l'entrada calculat iterativament (vegeu el programa principal a **`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, les funcions que us demana l'enunciat. Aquest fitxer l'heu de completar amb el codi que falta, i això, **tot**, és el que heu d'enviar al Jutge com a solució.

Dins el fitxer **`code.py`** teniu la classe **`BST`** que hem treballat a les classes de l'assignatura. Ha estat modificada per fer més senzilla la solució del problema (vegeu els comentaris a **`code.py`**). No caldrà que la vostra solució faci cap *import* ni res. Tot el codi que us cal el teniu dins de **`code.py`**.

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

Exemple d'entrada 1	Exemple de sortida 1

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

Exemple d'entrada 2

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

Exemple de sortida 2

```
545 545
```

Exemple d'entrada 3

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

Exemple de sortida 3

```
805 805
```

Exemple d'entrada 4

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

Exemple de sortida 4

```
427 427
```

Exemple d'entrada 5

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

Exemple de sortida 5

```
450 450
```

Informació del problema

Autoria: Jordi Delgado

Generació: 2026-06-10T17:03:15.637Z