Partició amb un diccionari BST. Cercar el representant del bloc al que pertany una clau. X96496_ca

Donada la classe *mfset* que permet gestionar particions (MFSets) on només hi guardem claus úniques usant arbres binaris de cerca (BST), cal implementar el mètode

```
// Pre: cert
// Post: Si k hi és, retorna un punter al node representant del bloc al que pertany k.
// Si k no hi és, retorna nullptr.
node* find_aux(const Clau &k) const;
```

Les claus són del tipus *Clau* que admet una relació d'ordre total, és a dir, tenim una operació de comparació $<$ entre claus. Les claus que pertanyen a un mateix bloc de la partició tenen el mateix representant encara que no necessàriament el node que conté la clau apunta directament al seu representant (punter *_pare_mfset*), ja que el mètode *merge*, que ja està implementat, utilitza l'estratègia Quick-union.
Cal enviar a jutge.org la següent especificació de la classe *mfset* i la implementació del mètode dins del mateix fitxer. La resta de mètodes públics i privats ja estan implementats.

```
#include <iostream>
using namespace std;
typedef unsigned int nat;


template <typename Clau>
class mfset {
  // Partició on les operacions find i merge s'han implementat amb l'estratègia
  // Quick-union. Les claus de la partició es guarden en un BST.

  public:
    // Constructora per defecte. Crea una partició buida.
    mfset ();

    // Destructora
    ~mfset ();

    // Pre: cert
    // Post: Insereix la clau k en la partició posant-la en un nou bloc.
    // Si ja hi era, no fa res.
    void insereix (const Clau &k);

    // Pre: cert
    // Post: Fusiona els blocs de les claus k1 i k2 amb l'estratègia Quick-union.
    // Si k1 o k2 no hi és, no fa res.
    void merge(const Clau &k1, const Clau &k2);

    // Pre: cert
    // Post: Si k hi és, retorna true i la clau del representant del bloc al que pertany k.
```

```
    // Si k no hi és, retorna false i la clau k.
    pair<bool, Clau> find(const Clau &k) const;

  private:
    struct node {
      Clau _k;          // Clau
      node* _esq;       // fill esquerre del BST
      node* _dret;      // fill dret del BST
      node* _pare_mfset;   // pare de la partició, apunta a nullptr si és el representant del
bloc
      node(const Clau &k, node* esq = nullptr, node* dret = nullptr);
    };
    node * _arrel;      // punter a l'arrel del BST

    static void esborra_nodes(node* m);
    static node* insereix_bst(node *n, const Clau &k);

    // Pre: cert
    // Post: Si k hi és, retorna un punter al node representant del bloc al que pertany k.
    // Si k no hi és, retorna nullptr.
    node* find_aux(const Clau &k) const;

    // Aquí va l'especificació dels mètodes privats addicionals

};

// Aquí va la implementació del mètode find_aux i dels mètodes privats addicionals
```

Degut a que jutge.org només permet l'enviament d'un fitxer amb la solució del problema, en el mateix fitxer hi ha d'haver l'especificació de la classe i la implementació del mètode *find_aux* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*).
Per testejar la classe disposes d'un programa principal que processa fragments que contenen una partició amb claus enteres seguit de comandes per fer fusions de blocs (merge) i cercar representant d'un bloc (find).

Entrada

L'entrada conté varis fragments separats per línies amb 10 guions (————). Cada fragment consisteix en una línia que conté una seqüències d'enters, són els elements que tindrà originalment la partició. A continuació segueixen vàries comandes, una per línea, amb el següent format, on k, $k1$ i $k2$ són claus enteres:

- *find k*
- *merge k1 k2*

Sortida

Per a cada línia d'entrada, escriu una línia amb el resultat:

- Si la línia conté els elements inicials de la partició, mostra el nombre de claus de la partició un cop inserit tots els seus elements.
- Si la línia és una comanda, mostra la comanda, el separador ": " i el resultat. Si la comanda és un *merge* no es mostra res més. Si la comanda és un $find$ es mostra el representant del bloc de la clau donada si existeix.
- Si la línia és el separador de blocs format per 10 guions, mostra els mateixos 10 guions.

Observació

Només cal enviar la classe requerida i la implementació del mètode $find\_aux$. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat. El mètode $find\_aux$ almenys ha de tenir cost logarítmic (en el cas mig) per superar els jocs de prova privats.

Exemple d'entrada 1

```
find 3
merge 2 3
find 3
```

Exemple de sortida 1

```
0
find 3:
merge 2 3:
find 3:
```

Exemple d'entrada 2

```
5
find 5
find 6
merge 5 5
find 5
```

Exemple de sortida 2

```
1
find 5: 5
find 6:
merge 5 5:
find 5: 5
```

Exemple d'entrada 3

```
7 5
find 5
find 6
find 7
merge 6 7
find 5
find 6
find 7
merge 5 7
find 5
find 6
find 7
```

Exemple de sortida 3

```
2
find 5: 5
find 6:
find 7: 7
merge 6 7:
find 5: 5
find 6:
find 7: 7
merge 5 7:
find 5: 7
find 6:
find 7: 7
```

Exemple d'entrada 4

```
-5 3 -2 1 -7 7 6
find -7
find -5
find -2
merge -5 -2
find -7
find -5
find -2
merge -2 -7
find -7
```

```
find -5
find -2
merge -5 -7
find -7
find -5
find -2
find 1
find 6
merge 6 1
find 1
find 6
merge 1 -5
```

```
find -7
find -5
find -2
find 1
find 6
merge 7 3
merge -2 7
find -7
find -5
find -2
find 1
find 3
find 6
find 7
```

Exemple de sortida 4

```
7
find -7: -7
find -5: -5
find -2: -2
merge -5 -2:
find -7: -7
find -5: -2
find -2: -2
merge -2 -7:
find -7: -7
find -5: -7
find -2: -7
merge -5 -7:
find -7: -7
find -5: -7
find -2: -7
find 1: 1
find 6: 6
merge 6 1:
find 1: 1
find 6: 1
merge 1 -5:
find -7: -7
find -5: -7
find -2: -7
find 1: -7
find 6: -7
merge 7 3:
merge -2 7:
find -7: 3
find -5: 3
find -2: 3
find 1: 3
find 3: 3
find 6: 3
find 7: 3
```

Informació del problema

Autor : Jordi Esteve
Generació : 2024-01-09 13:19:09