Interval Finder S54260_ca

Preàmbul: Intervals

Parlarem d'un **interval** com un conjunt de nombres *enters* compresos entre un límit inferior (`begin`) i un límit superior (`end`), ambdós inclosos. Els intervals es poden veure, també, com un cas particular de rectangles en una dimensió.

La representació en C++ per a intervals que farem servir en aquest problema és la tupla `Interval` (fitxer `interval.hh`), amb la següent declaració:

```
struct Interval {
    string id;       // Identificador únic
    int begin, end;  // Inici i final (ambdós inclosos)
};
```

Els intervals també tenen un identificador (el camp `id`), que és únic per a cada interval. Pot haver-hi intervals amb els mateixos límits però mai amb el mateix `id`. Pels intervals també tenim definit un operador de comparació, que simplement utilitza el camp `id`, cosa que permet utilitzar-los com a claus d'un `map` i elements d'un `set`:

```
bool operator<(const Interval& a, const Interval& b) {
    return a.id < b.id;
}
```

(També s'inclou una funció `intersects` que implementa la intersecció d'intervals.)

Objectiu del problema

Implementa la classe `IntervalFinder`, amb la següent especificació:

```
class IntervalFinder {
 public:
    IntervalFinder();

    void add(Interval range);
    std::set<string> query(int begin, int end) const;
};
```

Aquesta classe és una versió en petit del `Finder` utilitzat a la pràctica "Mario PRO2", amb només un constructor, un mètode per afegir intervals (`add`), i un mètode de consulta (`query`). El mètode `add` permet afegir un `Interval`, i `query` permet fer una consulta per obtenir els **identificadors** dels intervals que tenen algun punt entre `begin` i `end`, ambdós inclosos.

Tal com en el `Finder.hh` del Mario PRO2, l'objectiu és que `IntervalFinder` pugui emmagatzemar molts milers d'intervals. Però al fer una consulta, cal retornar, de forma eficient, aquells intervals que tenen algun punt entre `begin` i `end`.

Els fitxers públics (icona del gatet) contenen:

`interval.hh`	tupla `Interval` amb operador i funció `intersects`
`interval-finder.hh_`	**fitxer a editar, canvieu l'extensió a** `hh`**!**
`main.cc`	el programa principal (que ja s'encarrega de l'entrada i la sortida)
`Makefile`	per compilar amb `make` al terminal
`.vscode`	per compilar i debuggar amb F5

Observació

Per treballar, descarrega el codi públic, i edita el fitxer `interval-finder.hh_` (canvia l'extensió primer!).

Pots fer tots els mètodes de la classe `IntervalFinder` *inline* (és a dir, no cal fer un fitxer `interval-finder.cc` i es poden definir els mètodes a la declaració, a on hi ha les fletxes).

Cal enviar **un** `.tar` **només amb el fitxer** `interval-finder.hh`.

Entrada

De l'entrada se n'encarrega ja el programa principal. L'entrada consisteix en una sèrie de comandes, una per línia, que o bé afegeixen un interval al `IntervalFinder` (si comencen per `+`) o bé fan una consulta (si comencen per `?`). Els dos tipus de comandes tenen un interval al final (dos enters) però la comanda `+` també inclou l'identificador de l'interval, just després del signe `+`.

Sortida

De la sortida també se n'encarrega el programa principal. La sortida consisteix en els resultats de totes les consultes, un per línia. Quan el resultat és buit, el programa posa "empty" en una línia apart. Quan la consulta retorna un conjunt d'identificadors no buit, es mostra el prefix "`ids: `", i després tots els identificadors del resultat, per ordre lexicogràfic.

Exemple d'entrada

```
? 1 1
+ a 1 1
? 1 1
+ b 2 2
? 0 0
? 0 2
+ c 0 4
? 1 2
? 3 4
```

Exemple de sortida

```
empty
ids: a
empty
ids: a b
ids: a b c
ids: c
```

Informació del problema

Autor : Pau Fernández
Generació : 2025-06-06 13:42:11