Matrius esparses P64949_ca

Volem disposar d'una classe que permeti treballar amb matrius esparses. En aquest problema, una matriu esparsa és un matriu quadrada on la majoria dels seus elements són zero.

Per representar una matriu esparsa $n \times n$ eficientment, s'utilitza un vector d'n llistes, de forma que la llista i conté una successió de valors $\langle j, x_{i,j} \rangle$ (ordenats per j) que representen els valors $x_{i,j}$ de la matriu que no són zero.

Per exemple, la matriu

$$\begin{pmatrix} 3 & 0 & 4 & 0 \\ 0 & 6 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$$

es representa per un vector de quatre llistes:

$$\begin{aligned} 0: &\ \langle 0,3 \rangle \longleftrightarrow \langle 2,4 \rangle \longleftrightarrow \otimes \\ 1: &\ \langle 1,6 \rangle \longleftrightarrow \langle 3,1 \rangle \longleftrightarrow \otimes \\ 2: &\ \otimes \\ 3: &\ \langle 1,2 \rangle \longleftrightarrow \otimes \end{aligned}$$

on $\otimes$ denota l'*end()* d'una llista.

Les operacions públiques de la classe permeten:

- Construir una matriu de mida $n \times n$ plena de zeros.
- Consultar el valor d'una posició determinada d'una matriu (*get*).
- Modificar el valor d'una posició determinada d'una matriu (*set*). Per senzillesa, el nou valor no pot ser zero.
- Sumar una matriu a una altra (*add*).
- Escriure la matriu en format estàndard (*print*).

L'esquelet de la classe i el programa principal ja se us dona implementat (descarregueu el fitxer *code.cc*). Heu de completar les funcions de consulta (*get*), modificació (*set*) i de suma (*add*), sense canviar la interfície de les funcions, ni alterar la definició de les dades. A més, aquestes funcions no es poden cridar entre elles.

Les vostres implementacions han de ser eficients en el cas de matrius amb molt pocs elements diferents de zero: Suposant que el nombre d'elements no zero d'una matriu $n \times n$ sigui $\mathrm{O}(n)$, les vostres operacions han de tenir cost $\mathrm{O}(n)$ en el cas pitjor.

Exemple d'entrada

```
create a 4
print a
#
set a 0 0 3
set a 0 2 4
set a 1 1 6
set a 1 3 1
set a 3 1 2
print a
#
get a 0 0
get a 0 1
#
create b 4
set b 0 0 -3
set b 2 3 9
print b
#
add a b
print a
```

Exemple de sortida

```
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

3 0 4 0
0 6 0 1
0 0 0 0
0 2 0 0

3
0

-3 0 0 0
0 0 0 0
0 0 0 9
0 0 0 0

0 0 4 0
0 6 0 1
0 0 0 9
0 2 0 0
```

Informació del problema

Autor : Jordi Petit
Generació : 2024-05-02 21:14:54