RareMatrix - Documentatie

Andronache Costin B5

Prin matrice rara intelegem o matrice unde numarul de elemente nenule este substantial mai mic decat numarul elementelor nule.

In cazul matricilor rare, pentru a economisi spatiul de memorie se prefera a se memora doar tuple de forma ***(indiceLinie, indiceColoana, valoare)*** pentru valorile nenule, iar operatiile matriciale (adunarea, inmultirea) sa fie implementate peste aceasta multime de tuple.
Este important de observat ca in cazul acesta, pentru fiecare valoare nenula se memoreaza inca 2 valori intregi, ceea ce duce la un consum de memorie ***O(3*Nnen)***, unde ***Nnen*** este numarul elementelor nenule.
Aceasta schema de memorare ar fi de preferat deci, atunci cand stim ca ***Nnen < (N * M)/3***, unde ***N-*** numarul de linii ale matricei, ***M-***numarul de coloane.

Putem aplica o optimizare si sa combinam toate tuplele avand acelasi indice de linie intr-o lista, iar structura de date pentru memorarea matricii va fi in acest caz o multime de liste de perechi ***(indiceColoana, valoare).***

O prima varianta de implementare este folosirea unui vector/ array /lista de liste.
Declaram deci ***List< List< Pair<Integer, Double> > > linesList,*** iar ***linesList[i]*** contine lista de perechi ***(indiceColoana, valoare)*** de pe linia ***i*** .
In acest caz, consumul de memorie va fi ***O(2*Nnen + N),*** o imbunatatire fata de varianta precedenta.
Aceasta metoda are un dezavantaj in cazul in care exista un numar mare de linii nenule in matrice, si deci un numar mare de indecsi de liste goale.

Pentru a elimina acest inconvenient, vom folosi tabele asociative in cazul liniilor(***HashMap-***uri in Java), dar si in cazul coloanelor de asemenea, pentru ca acestea ofera performanta mai buna in cazul cautarii/ accesarii unui element mai intai dupa indicele liniei si apoi dupa indicele de coloana.

Structura de date pentru reprezentarea unei linii va fi deci o tabela asociativa ***indexColoana ⇒ valoare*** (***HashMap<Integer, Double>***) iar o pentru memorarea liniilor matricii vom folosi inca o tabela asociativa ***indexLinie ⇒ tabela_asociativa_linie*** (***HashMap< Integer, HashMap<Integer, Double> >***)

Fie matricea:

$$A = \begin{pmatrix} 102.5 & 0.0 & 2.5 & 0.0 & 0.0 \\ 3.5 & 104.88 & 1.05 & 0.0 & 0.33 \\ 0.0 & 0.0 & 100.0 & 0.0 & 0.0 \\ 0.0 & 1.3 & 0.0 & 101.3 & 0.0 \\ 0.73 & 0.0 & 0.0 & 1.5 & 102.23 \end{pmatrix}$$

Atunci, urmand schema de memorare propusa,

A : *HashMap<Integer, HashMap<Integer, Double> >* =
{
0 ⇒ *HashMap<Integer, Double>* = {
0 ⇒ 102.5,
2 ⇒ 2.5,
},
1 ⇒ *HashMap<Integer, Double>* = {
0 ⇒ 3.5,
1 ⇒ 104.88,
2 ⇒ 1.05,
4 ⇒ 0.33
},

2 ⇒ *HashMap<Integer, Double>* = { 2⇒ 100.0},
..etc
}

Prezentam acum algoritmul de adunare a doua matrici rare, urmand schema propusa:

Input: *N1: Integer, M1: Integer, A: HashMap, N2: Integer, M2: Integer, B: HashMap*
Output: *Sum: HashMap*

1. *Daca N1 =/= N2 SAU M1 =/= M2 return { NIL;}*
2. *For (i in 0 ..<N1)*
3. *{*

```
A_Line_i : HashMap = A.get(i)
B_Line_i: HashMap = B.get(i);


if( A_Line_i == NIL  AND  B_Line_i != NIL)
{
        Sum[i] = = B_Line_i.copy();

}else if( A_Line_i != NIL AND B_Line_i == NIL)
        {  // similar ca mai sus, doar ca vom copia A_Line_i }
else
{
```

   a) *Vom copia A_Line_i si o vom pune in Sum[i]*
   b) *Vom aduna la elementele din Sum[i] acele elemente din B_Line_i care se gasesc pe aceeasi indecsi de coloana*
   c) *Vom adauga in Sum[i] acele elemente din B_Line_i care nu au fost incluse in procesarea de la b)*

```
    Sum[i] = A_Line_i.copy()
    // De regula, in orice limbaj de programare un hashmap ofera
    // posibilitatea de a extrage cheile existente din el, sub forma unei
    // liste sau un set

    A_columnIndexes : Set<Integer> = A_Line_i.keySet()
    B_columnIndexes: Set<Integer> = B_Line_i.keySet();

    Common_columnIndexes =  A_columnIndexes.intersect(B_colum..)

    for( column: Integer in  Common_columnIndexes)
    {
        val_in_Sum: Double = Sum[i][column]
        val_in_B_Line_i : Double = B_Line_i[column]
```

```
                Sum[i][column] = val_in_Sum + val_in_B_Line_i;
            }

            only_B_indexes: Set<Integer> =
            B_columnIndexes.minus(A_columnIndexes)

            for(column : Integer in only_B_indexes)
            {
                Sum[i][column] = B_Line_i[column]
            }
    }
```