**Bài toán ma trận thưa**

1. Mô tả bài toán
2. Phân tích để chọn cách lưu trữ
3. Hiện thực
4. Một cách dùng
5. Bài tập
6. **Mô tả bài toán**

Có một ma trận **R** hàng **C** cột với R và C là các số lớn ( trị khoảng vài ngàn hoặc lớn hơn) nhưng chỉ thực sự chứa rất ít phần tử mang trị hữu ích. Hãy hiện thực các tác vụ cơ bản trên ma trận này như:

* Thêm một phần tử vào dòng i cột j
* Xoá phần tử vào dòng i cột j
* Xuất hàng i
* Xuất cột j
* …

Đây là bài toán điển hình về cân bằng hiệu suất sử dụng bộ nhớ với hiệu suất về thời gian truy xuất phần tử/dòng/cột.

**Vài áp dụng**

* Tìm ma trận tổng, tích của hai ma trận thưa trong đó đa phần là trị 0, rất ít trị khác 0.
* Quản lý bảng điểm sinh viên theo môn học: Có độ 500 môn học với 5000 sinh viên nhưng mỗi sinh viên chỉ học khoảng 30 môn học và mỗi môn học chỉ có khoảng 100 sinh viên 🡺 Khả năng: 500.5000=2500000 phần tử, hiệu dụng: 30.100= 3000 phần tử.

1. **Phân tích để chọn cách lưu trữ**

**2.1- Cách 1: Dùng mảng hai chiều**

Gọi S (size) là kích thước bộ nhớ của dữ liệu của mỗi phần tử.

Số phần tử cần: **RC** ( tích này có trị hàng triệu)

Tổng bộ nhớ cần: **RCS**

Nhưng số phần tử thực sự hữu ích rất nhỏ so với tích này 🡺 Phí bộ nhớ

Gọi **r** là số hàng trung bình có phần tử

Gọi **c** là số cột trung bình có phần tử

Số phần tử thực có: **rc**

**Nhận xét:**

* Rất phí bộ nhớ
* Truy xuất một phần tử rất nhanh nhờ hai chỉ số hàng và chỉ số cột 🡪 O(1)

Vậy có cách nào tiết kiệm bộ nhớ mà vẫn truy xuất phần tử khá nhanh không? 🡪 Góc nhìn cân bằng (trade-off) giữa hiệu suất về bộ nhớ và hiệu suất về thời gian

**2.2- Cách 2: Dùng danh sách liên kết đơn cơ bản**

Mổi nút trong DSLK đơn có cấu trúc <**row, col, info, next**> mang ý nghĩa phần tử này có dữ liệu là info( lưu trữ tốn kích thước bộ nhớ là S) nằm ở hàng **row**, cột **col**, và phần tử kế tiếp ở địa chỉ **next**.

Giả sử rằng 3 thành phần **row, col, next** cùng tốn bộ nhớ là 4 bytes.

Số phần tử trung bình là **rc** nên kích thước bộ nhớ cần lưu trữ: **rc( S+ 12)**

Tích này cũng vẫn rất bé so với **RCS** ( tham khảo phân tích trước)

Việc lưu trữ các info có thứ tự theo chỉ số hàng, trong một hàng lại có thứ tự theo chỉ số cột. Thí dụ: ( Nên hiểu rằng các phần tử được trình bầy cạnh nhau nhưng là danh sách liên kết đơn)

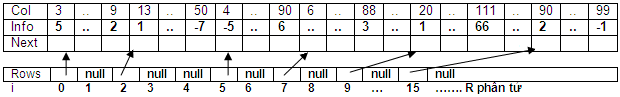
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row | 0 | .. | 0 | 2 | .. | 2 | 5 | .. | 5 | 7 | .. | 7 | .. | 9 | .. | 9 | .. | 15 | .. | 15 |
| Col | 3 | .. | 9 | 13 | .. | 50 | 4 | .. | 90 | 6 | .. | 88 | .. | 20 | .. | 111 | .. | 90 | .. | 99 |
| Info | **5** | **..** | **2** | **1** | **..** | **-7** | **-5** | **..** | **6** | **..** | **..** | **3** | **..** | **1** | **..** | **66** | **..** | **2** | **..** | **-1** |
| Next |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Nhận xét:**

* Sử dụng bộ nhớ: **rc(S+12)**
* Truy xuất một phần tử chậm vì phải duyệt tuyến tính: **O(rc)**

**2.3- Cách 3: Dùng danh sách liên kết đơn có mảng phụ trợ:**

Vẫn dùng cách lưu trữ như trên nhưng ta chịu tốn thêm mảng **Rows** chứa R tham khảo đến các phần tử đầu hàng và loại bỏ thành phần row trong cấu trúc phần tử.

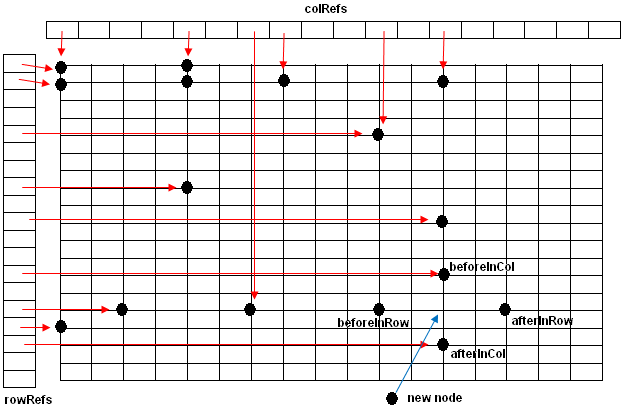


**Nhận xét:**

* Kích thước bộ nhớ phải dùng:  **4R + rc(S+8)**
  + Mảng địa chỉ Rows: **4R**
  + Một phần tử: **S + 8**
  + Tập phần tử: **rc(S+8)**
* Truy xuất một phần tử nhanh hơn cách tiếp cận danh sách liên kết đơn cơ bản khá nhiều vì từ chỉ số hàng **i**, thông qua **Rows[i]** ta biết ngay nhóm phần tử của dòng này duyệt nhóm con của dòng này ta lấy được phần tử ở cột **Col=j 🡺 O(c)**

**2.4- Cách 4: Dùng một biến thể của danh sách liên kết đôi**

Dùng 2 mảng: Mảng **rowRefs** chứa các địa chỉ của phần tử đầu của các dòng, mảng **colRefs** chứa các địa chỉ phần tử đầu của các cột.



Mỗi phần tử có cấu trúc <info, nextInCol, nextInRow> là đủ 🡪 Tốn (S + 4+4) = S + 8 bytes

Tuy nhiên, với cấu trúc này khi duyệt để tìm phận phần tử ở hàng i cột j chúng ta phải tiến hành như sau:

Từ rowRefs[i], ta duyệt dòng i. Khi đến một phần tử, chúng ta không biết phần tử này thuộc cột nào nên lại phải duyệt tất cả các cột.

🡪 Hiệu suất quá kém.

🡪 Đưa thêm thông tin về dòng, cột sẽ giúp tìm nhanh hơn rất nhiều. Cấu trúc của một nút: <row, col, info, nextInCol, nextInRow> 🡪 Bộ nhớ tốn 8 + S + 8= S+16 bytes

**Nhận xét**

* Tổng bộ nhớ: **4R + 4C + rc(S+16)** trong đó:
  + Bộ nhớ cho mảng rowRefs: **4R**
  + Bộ nhớ cho mảng colRefs: **4C**
  + Bộ nhớ cho rc phần tử: **rc(S+16)**
* Truy xuất phần tử dòng I, cột j: Chỉ cần tiến hành duyệt tuyến tính danh sách rowRefs[i] hoặc colRefs[j] 🡪 max(O(c), O(r))

**Tóm lại:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Cách 1: Mảng 2 chiều** | **Cách 2: DSLK đơn cơ bản** | **Cách 3: DSLK đơn có sử dụng màng địa chỉ hàng** | **Cách 4: Dùng biến thể của DSLK đôi** |
| **Lưu trữ** | RCS | rc(S+12) | **4R + rc(S+8)** | **4R + 4C + rc(S+16)** |
| **Thử với R=1000**  **C=1000, S=4 (int), r=20, c=30** | 1000.1000.4=  4000000 | 20.30.16=9600 | 4.1000 + 20.30.12= 11200 | 4.1000 + 4.1000 + 20.30.20= 20000 |
| **Tỉ lệ sử dụng bộ nhớ** | 100% | 0.24% | 0.28% | 0.5% |
| **Truy xuất 1 dòng** | O(c) | O(rc)  Vì phải duyệt toàn bộ sách | O(c)  Vì chỉ phải duyệt DSLK của dòng tương ứng | O(c)  Vì chỉ phải duyệt DSLK của dòng tương ứng |
| **Truy xuất 1 cột** | O(r) | O(rc)  Vì phải duyệt toàn bộ sách | O(rc) vì phải duyệt hết các dòng | O(r)  Vì chỉ phải duyệt DSLK của cột tương ứng |
| **Tìm phần tử dòng i cột j** | O(1) | O(rc) | O(c) | Max(O(r), O(c |

Như vậy, ta có thể kết luận cách 4 là tốt nhất trong 4 cách tiếp cận với hệ số sử dụng bộ nhớ chấp nhận được và hiệu suất về thời gian truy xuất khả thi.

1. **Hiện thực**

Chọn cách 3 để hiện thực.

**Bước 1:** Xây dựng lớp mô tả tổng quát cho một phần tử trong ma trận thưa (lớp **SparseMatrixNode** trong phần minh hoạ).

**Bước 2**: Xây dựng lớp mô tả cho ma trận thưa tổng quát (lớp **SparseMatrix** trong phần minh hoạ).

**Bước 3**: Xây dựng lớp cụ thể cho info của một nút nếu cần ( nếu info của nút chỉ có kiểu cơ bàn như String, Number) thì bỏ qua bước này.

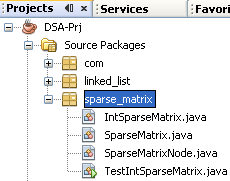
**Bước 4**: Xây dựng lớp con của lớp DSLK đã mô tả trong bước 2 cùng với những hành vi đặc thù của bài toán (lớp **IntSparseMatrix** trong phần minh hoạ để mô tả ma trận thưa các số nguyên cùng với hai phép toán cộng, nhân hai ma trận).

**Bước 5**: Xây dựng lớp có hàm main để thành chương trình (lớp **TestIntSparseMatrix** trong phần minh hoạ)..

1. **Một cách dùng**

Hiện thực hai phép toán cộng và nhân hai ma trận thưa chứa các số nguyên rồi chạy thử chương trình với hai ma trận được để trong phần chú thích của chương trình.

**Kiến trúc của lới giải**



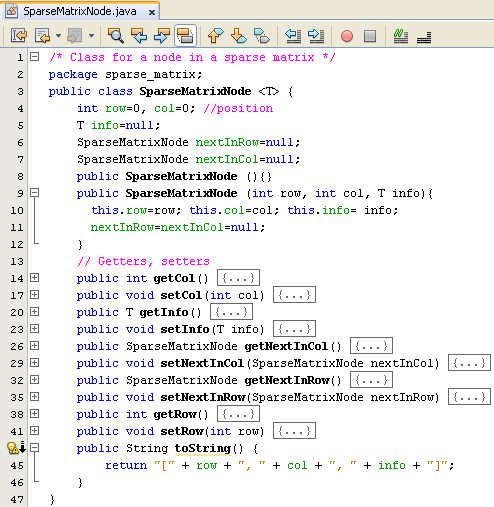
**SparseMatricNode**: Lớp cho một nút tổng quát trong ma trận thưa.

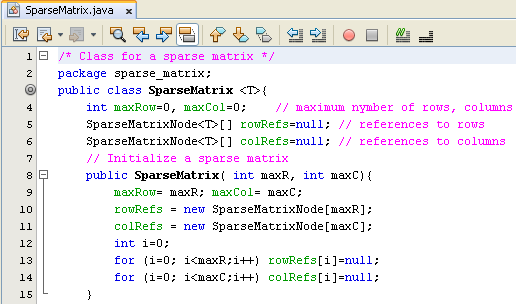
**SparseMatrix**: Lớp mô tả tổng quát cho 1 ma trận thưa

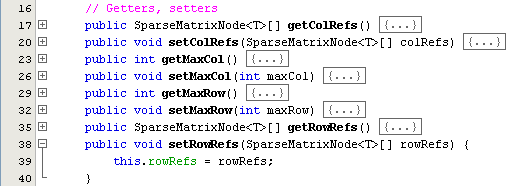
**IntSparseMatric**: Lớp mô tả cho một ma trận thưa các số nguyên có tác vụ cộng/nhân hai ma trận

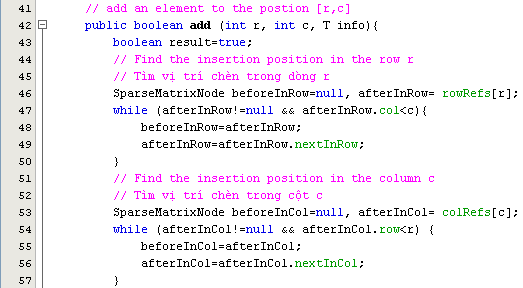
**TestIntSparseMatrix**: Lớp có hàm main trong đó nhập 2 ma trận, xuất hai ma trận này cùng hai ma trận tổng tìch của chúng

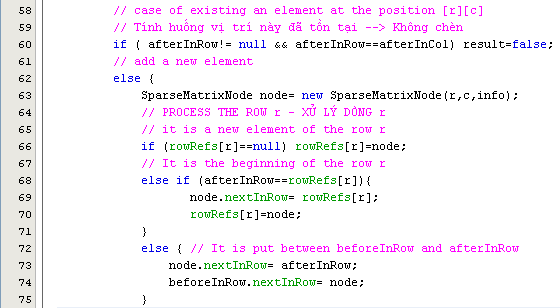
**Codes**

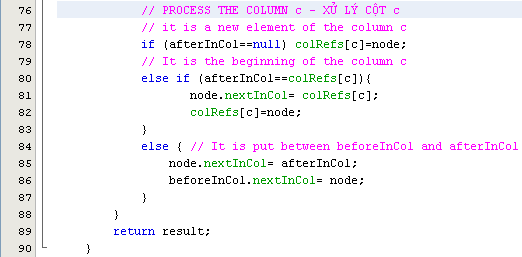


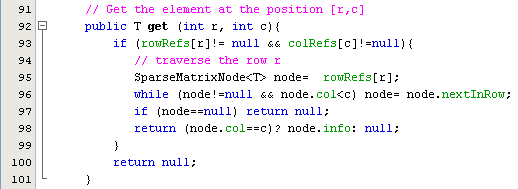


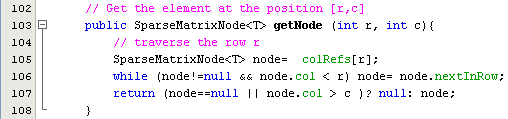


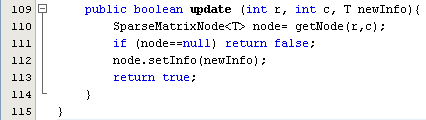


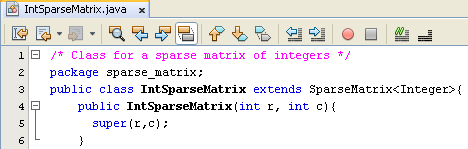


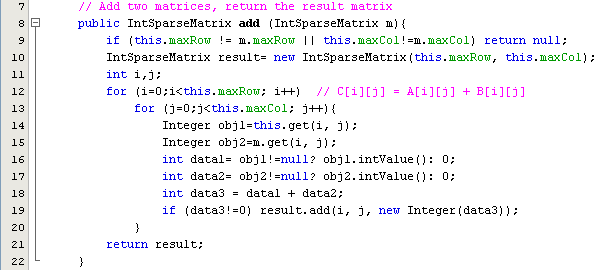


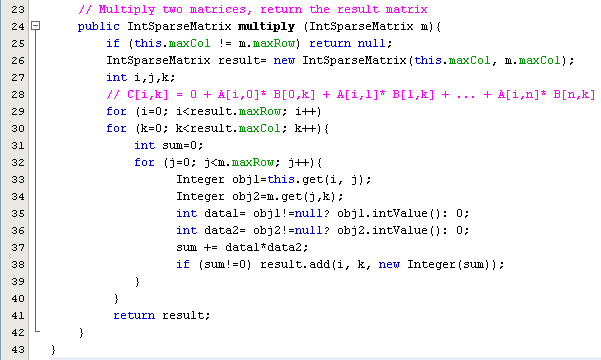




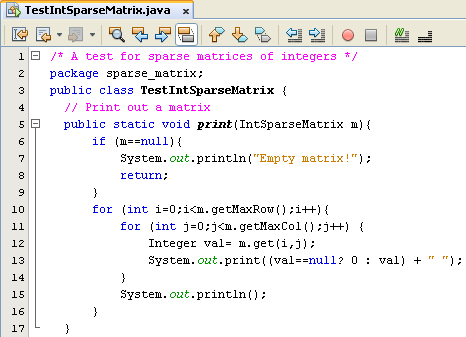


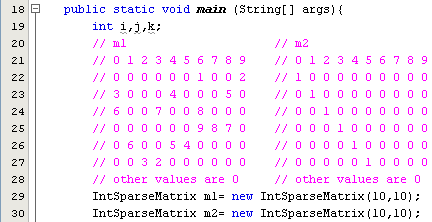


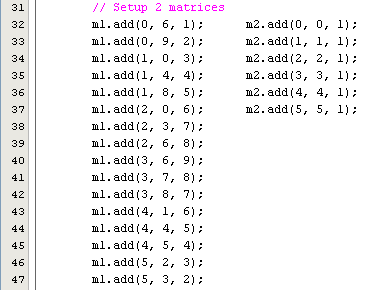


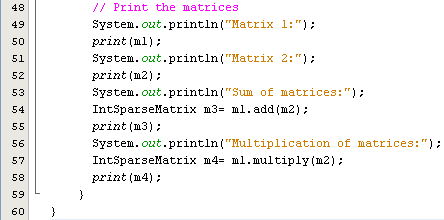


this.maxRow

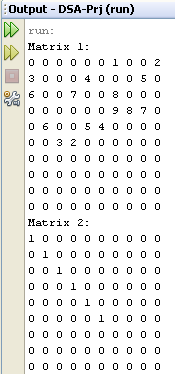
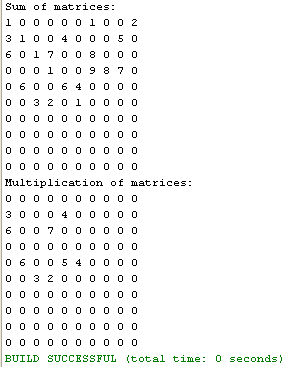








**Kết quả**

1. **Bài tập**

Áp dụng ma trận thưa để viết chương trình quản lý bảng điểm cho học sinh có menu đơn giản.

Có nhiều môn học, mỗi môn học chỉ cần được biểu diễn bằng một số nguyên ( mảng chỉ số hàng). Có nhiều sinh viên. Mỗi sinh viên chỉ cần được biểu diễn bằng 1 số nguyên ( mảng chỉ số cột). Thông tin bảng điểm mộ tả: môn học có chỉ số nào, được sinh viên chỉ số nào học, kết quả thi là bao nhiêu điểm. Như vậy, ma trận thưa các số nguyên sẽ được dùng để quản lý bảng điểm.

**Menu của chương trình:**

1. Nhận điểm cho một môn học với thuật toán được đề nghị như sau:

* Nhập chỉ số môn học;
* Lặp nhập < chỉ số sinh viên, kết quả thi>. Hỏi user nhập nữa không? Kết thúc nhập khi user chọn N

1. In bảng điểm của một môn học. Thuật toán được đề nghị:

* Nhập chỉ số môn học
* Chương trình in ra bảng điểm gồm <chỉ số sinh viên, kết quả>

1. In bảng điểm của sinh viên. Thuật toán được đề nghị:

* Nhập chỉ số sinh viên
* Chương trình in ra bảng điểm gồm <chỉ số môn học. kết quả>

1. Sửa điểm cho sinh viên. Thuật toán được đề nghị:

* Nhập chỉ số sinh viên
* Nhập chỉ số môn học
* Xuất điểm cũ
* Nhập điểm mới
* Hỏi user có cập nhật không?
* Nếu user trả lời Y thì cập nhật điểm mới cho sinh viên.