Sparse Matrix Multiplication

What the interviewer is expecting when this problem is asked in an interview...

retrouvailles 300 Nov 02, 2019

For those who may now or in the future be interviewing with Facebook, I found many recourses online but most of them failed to give a clear explaination what is expecting when this problem is asked in an interview, which inspires me to write down a summary for this problem.

I guess when an interviewer askes this question in an interview, he/she may want to see if we can convert mathmatical equation to codes(how to do matrix multiplication using a programming language). Moreover, he/she may want to see how we handle huge space waste when we write a program. This would be the first follow up, What if the two input matrix
are large and we can not save all of them in memory?

In my opinion, the ideal answer is sparse matrix compression. We can use three vectors(array) to store the index and value of all non-zero elements in one original matrix. This is a common solution in the real world when we deal with large sparse matrix I think. (For those who may concern that we still need to loop throught the whole matrix to do the conversion, I think this follow-up question is asking about how we store these matrix can we get a better run time when do matrix multiplication, (which means the input matrix has been changed to compressed form).

We use two pointer to loop through the index array (This is 1D vector case, not 2D matrix multiplication.) and the run time is O(m + n), m and n are number of non-zero elements in the original vectors.

```
int doc_product(vector<int>& n1, vector<int>& n2, vector<int>& idx1, vector<int>& idx2) {
    int p1 = 0, p2 = 0;
    int ret = 0;
    while (p1 < idx1.size() && p2 < idx2.size()) {
        if (idx1[p1] == idx2[p2]) {
            ret += n1[idx1[p1]] * n2[idx2[p2]];
            p1++;
            p2++;
        } else if (idx1[p1] < idx2[p2]) {
            p1++;
        } else {
            p2++;
        }
    }
    return ret;
}
```

Then the interviewer might keep follow up: What if one vector is significantly longer than the other one? We could say we can traverse the shorter one and use binary search to find the matched index in the other vector. Then the run time would be O(m*logn). m is the length of the shorter vector while n is the length of the longer vector.

This gives us an idea that we could traverse one index vector while doing binary search in another index vector. When we use binary search, if we find some matched index, we can use this index as the begin for next search. If we can not find a matched one, we return the lower bound, for next turn.

eg: idx1 = [1, 89, 100], we are searching for 90 in this iteration, we can not find a 90, so we return 89 for the next search. (We can use lower_bound() in c++ to do this)
The complete code written in Java to do binary search is in https://github.com/SCIN/Facebook-Interview-Coding-1/blob/master/Sparce%20Matrix%20Multiplication.java

If my explaination is confusing, this problem uses binary search to find a index as well which could be helpful.
https://leetcode.com/problems/shortest-way-to-form-string/

I got lots of help from this https://github.com/SCIN/Facebook-Interview-Coding-1/blob/master/Sparce Matrix Multiplication.java.

Please correct me if I were wrong. Thank you!

261 Share Favorite

Next
Easiest JAVA solution

Comments (8) Sort by: Best

Type comment here...(Markdown supported)

Preview Comment

jiashi1994 Nov 26, 2021

Share my Java version solution with HashMap, applied above compression idea. runs 2ms

```
class Solution {
    public int[][] multiply(int[][] mat1, int[][] mat2) {
        int row1 = mat1.length;
        int col1 = mat1[0].length;
        int row2 = mat2.length;
        int col2 = mat2[0].length;

        // compress matrix
        // key is row number, value is the list of pairs (index, value)
        Map<Integer, List<int[]>> map1 = new HashMap<>();
        for(int i=0; i<row1; i++) {
            List<int[]> tmp = new ArrayList<>();
            for(int j=0; j<col1; j++) {
                if(mat1[i][j] != 0) {
                    tmp.add(new int[]{j, mat1[i][j]});
                }
            }
            map1.put(i, tmp);
        }

        // key is col number, value is the list of pairs (index, value)
        Map<Integer, List<int[]>> map2 = new HashMap<>();
        for(int j=0; j<col2; j++) {
            List<int[]> tmp = new ArrayList<>();
            for(int i=0; i<row2; i++) {
                if(mat2[i][j] != 0) {
                    tmp.add(new int[]{i, mat2[i][j]});
                }
            }
            map2.put(j, tmp);
        }

        int[][] res = new int[row1][col2];
        for(int i=0; i<row1; i++) {
            for(int j=0; j<col2; j++) {
                res[i][j] = dotOperation(map1.get(i), map2.get(j));
            }
        }
        return res;
    }

    private int dotOperation(List<int[]> row, List<int[]> col) {
        int p1 = 0, p2 = 0;
        int res = 0;
        while(p1 < row.size() && p2 < col.size()) {
            int idx1 = row.get(p1)[0];
            int idx2 = col.get(p2)[0];
            if(idx1 == idx2) {
                res += row.get(p1)[1] * col.get(p2)[1];
                p1++;
                p2++;
            } else if(idx1 < idx2) {
                p1++;
            } else {
                p2++;
            }
        }
        return res;
    }
}
```

17 Reply

leet22991 Apr 14, 2020

best ans. thanks a ton!

10 Reply

Accepted **bakaiti002** Feb 02, 2021

Read more

6 Reply

DyXrLxSTAOadoD Jul 08, 2020

Read more

5 Reply

bchary Jan 18, 2022

Here is my 2 cents:
The "smart"/Easiest solution" by @yavinci is smart because it allows accessing both matrices in row by row manner which enables sequential access of data from memory as matrix is stored as row1 row2 row3 ...
With the brute force approach, we have to jump from one memory location to another for each column. Basically, we have to make more data fetches from memory (a lot of cache misses if the second matrix is huge, not fitting to memory)

1 Reply

zhenzhongluo Oct 26, 2021

For LC 1570. Dot Product of Two Sparse Vectors,
someone mentioned that the interviewer wanted to see the Merge Interval way of solution, and said using hashtable has too much overhead.
I could not find anywhere with this solution. Can anyone ping me some resources (links) if you have any please?

1 Reply

leet22991 Apr 14, 2020

Hey, I hope you are still monitoring this post. Would really help me.
I am trying to understand how in your code, you are talking about just 2 vectors. we need at least 3 vectors, isn't it?
how can one vector of index index into an array?
Or are you suggesting, that in case of large sets, we read data one row from A and one col from B at a time ?

1 Show 2 Replies Reply

sarahzz Oct 13, 2020

This should be voted as best answer.

0 Reply

1

261