2. Problem 1: Suggesting Similar Papers

Bookmark this page

Analysis due Oct 27, 2021 17:29 IST Completed

A citation network is a directed network where the vertices are academic papers and there is a directed edge from paper A to paper B if paper A cites paper B in its bibliography. **Google Scholar** performs automated citation indexing and has a useful feature that allows users to find similar papers. In the following, we analyze two approaches for measuring similarity between papers.

Part (a): Co-citation network

Two papers are said to be cocited if they are both cited by the same third paper. The edge weights in the cocitation network correspond to the number of cocitations. In this part, we will discover how to compute the (weighted) adjacency matrix of the cocitation network from the adjacency matrix of the citation network.

- **Problem setup:** In order to derive the cocitation matrix, we need to derive it as a function of the original adjacency matrix.

- **Problem notation:** If there is an edge from paper i to paper j, it means that paper i cites paper j. We will denote by A the corresponding adjacency matrix, such that $A_{ij} = 1$ means there is a directed edge from i to j. Let us denote by C the cocitation network matrix.

Question 1

2.0/2.0 points (graded)

In attempting to derive the cocitation matrix, your friend came up with the following algorithm:

Assuming the row indices of the matrix mean that the paper is citing others, and the column indices that the paper is being cited, then the algorithm's steps would be:

- Construct an empty matrix for C.

- Go through the rows of A one by one.

- For each row r of A, if the row sum is strictly greater than 1, then do: for each pair $((r,a),(r,b))$ in row r that are non-zero (meaning that there is an existing relationship), add 1 to C at the location (a,b). Note that by following this rule, you will naturally also add 1 to C at location (b,a) as the pair $((r,b),(r,a))$ must also be present.

After reading carefully through the proposed steps, please answer the following:

Does this generate the cocitation weighted adjacency matrix?

- [x] Yes
- [ ] No

What is the big-O complexity, $\mathcal{O}$, of the proposed algorithm, in terms of n, the number of nodes in the graph?

$\mathcal{O}(\ldots)$ is n^3 ✔ **Answer:** n^3

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Question 2

5.0/5.0 points (graded)

Write the cocitation weighted adjacency matrix, C, in terms of A using matrix operations. Use `A^T` for A^T and `*` for matrix multiplication. The diagonals in your answer need not match the diagonals generated by the definition in Question 1, the off-diagonals should match Question 1.

$C =$ A^T*A ✔ **Answer:** A^T * A

Hint: How can you use A_{ki} and A_{kj} to represent a logical AND for edge (k, i) and (k, j) being present in the graph? Use this to write down C in terms of C_{ij} then find the corresponding matrix operations.

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Part (b): Bibliographic coupling

5.0/5.0 points (graded)

Two papers are said to be bibliographically coupled if they cite the same other papers. The edge weights in a bibliographic coupling correspond to the number of common citations between two papers.

How do you compute the (weighted) adjacency matrix of the bibliographic coupling, B, from the adjacency matrix of the citation network, A? Write your answer in terms of matrix operations.

$B =$ A*A^T ✔ **Answer:** A * A^T

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Part (c): (2 points) Include your answer to this question in your written report. (100 word limit.)

How does the time complexity of your solution involving matrix multiplication in part (a) compare to your friend's algorithm?

Grading rubric:

Award one of the following:

- **(1 point)**: Identifies that matrix multiplication algorithms are $\mathcal{O}(n^a)$ where $2 < a \leq 3$.
- **(2 points)**: Recognizes that in practice, matrix multiplication will be no faster than the friend's algorithm.

Example solution:

Although matrix multiplication algorithms exist which have time complexities between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, in practice, multiplication algorithms with time complexity of $\mathcal{O}(n^3)$ are used by linear algebra software libraries. The theoretically faster algorithms either can't be run on any realistic computer (as the memory requirements are astronomical) or they have huge constant multipliers to their runtime which make them far slower than $O(n^3)$ algorithms for all realistic matrix problems. Thus, the time complexity of the matrix multiplication is no faster than the friend's algorithm in practice. Note that this is true even when considering further optimisations that can be performed due to matrix sparsity.

Part (d): (3 points) Include your answer to this question in your written report. (200 word limit.)

Bibliographic coupling and cocitation can both be taken as an indicator that papers deal with related material. However, they can in practice give noticeably different results. Why? Which measure is more appropriate as an indicator for similarity between papers?

Grading rubric:

Award one of the following:

- **(1 point)**: States that bibliographic coupling is a better metric without justification.
- **(2 points)**: Gives a convincing argument for either metric, **but does not** recognize that papers include a wide variety of references.
- **(3 points)**: Gives a convincing argument for either metric, **and** recognizes that papers include a wide variety of references.

Example solution:

First, note that papers typically include a wide variety of references. For example: a scientific result paper may cite both a scientific review on the topic at hand, and a mathematical method paper to justify a particular procedure. Therefore, we might expect that the cocitation metric will be a poor indicator of paper similarity. On the other hand, if two papers cite the same third paper, then presumably they have some overlap in content. They may be papers in two different scientific fields, but use the same mathematical method, and thus cite the same reference on that method. Therefore, we can expect that bibliographic coupling will be better measure of paper similarity.

Discussion

Hide Discussion

Topic: Module 3: Network Analysis:Networks: Written Analysis, Peer Review and Discussion / 2. Problem 1: Suggesting Similar Papers

Add a Post

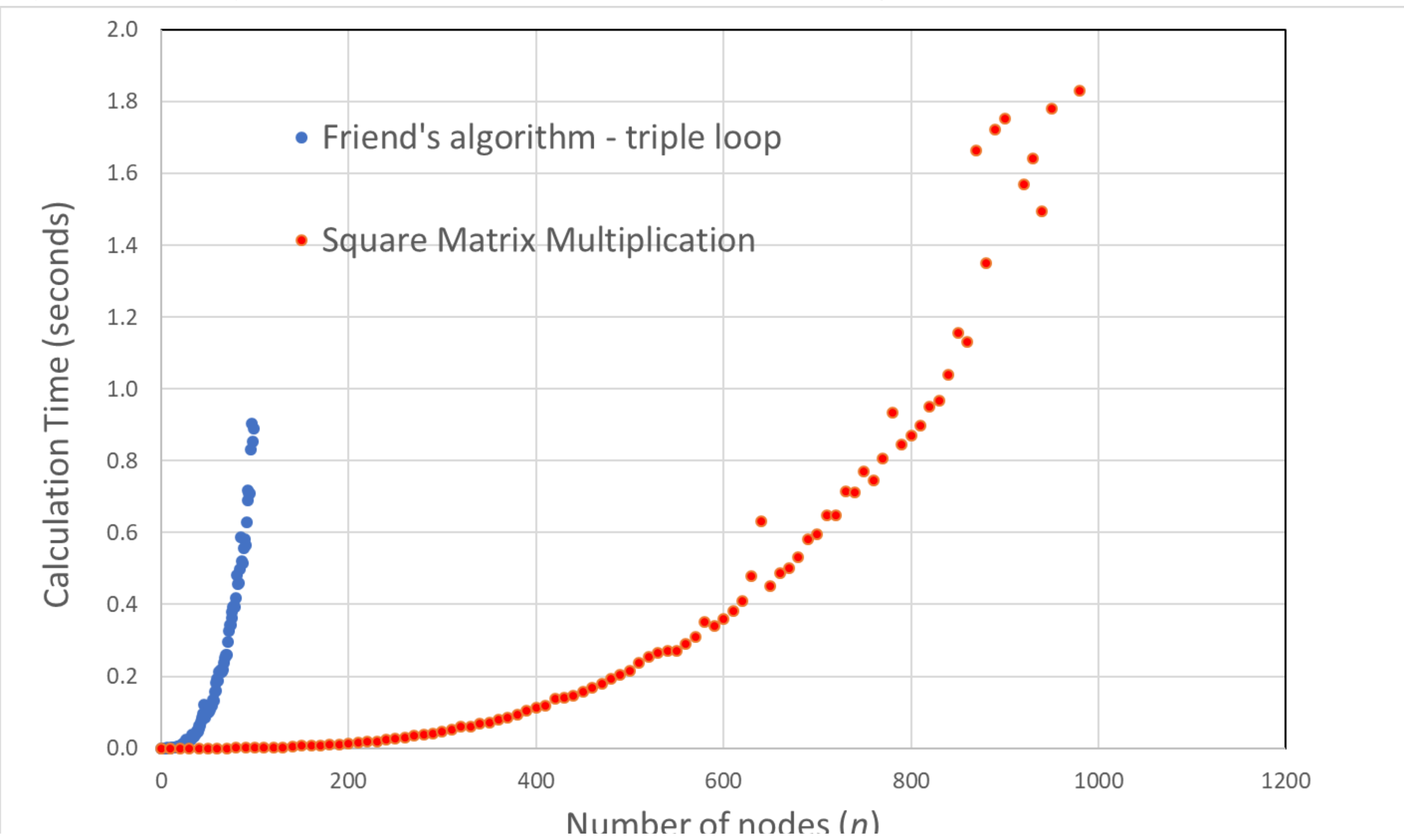
< All Posts

Matrix multiplication vs Triple loop

discussion posted 2 months ago by **FlorenGS**

I see that a triple loop is order $\mathcal{O}(n^3)$ and that the matrix multiplication is $\mathcal{O}(n^a)$ with 2 < a < 3 However, when I ran the triple loop algorithm vs a matrix multiplication the latter is much faster than the other and therefore preferable (see plot and code below). I found this happens not only on Python but also with R and Octave.

Anyone knows why this is? And wouldn't this mean that the second part of the rubric is not correct?



```
import numpy as np
import time

runs = 100
steps = 1 #steps for n
time_1 = np.zeros(runs) # time measurements for triple loop
time_2 = np.zeros(runs) # time measurements for matrix mult
count = 0

for dim in range(0,runs*steps,steps):
    #First we generate a random adjacency matrix
    nums = np.random.choice([0, 1], size=dim*dim, p=[.5, .5])
    nums = np.reshape(nums, (dim, dim))
    C = np.zeros((dim,dim))

    # We calculate C with my friends algorithm
    t1 = time.time()
    for row in range(dim):
        for i in range(dim):
            for j in range(dim):
                if nums[row,i]*nums[row,j] == 1:
                    C[i,j] =  C[i,j] + 1
    time_1[count] = time.time() - t1
    print(dim, time_1[count])

    #Now we use matrix multiplication instead
    t2 = time.time()
    C2 = nums.T@nums
    time_2[count] = time.time() - t2
    print(dim, time_2[count])

    count += 1
```

This post is visible to everyone.

Add a Response

2 responses

alenaaa
2 months ago

Well, the question is talking about time complexity. In other words, how the time of execution is expected to grow depending on the size on the input. Which doesn't translate directly to time in seconds of *any* implementation of the same theoretical time complexity. There are some other details of implementation that matter (memory access optimization etc.). So, you cannot compare your naive implementation with numpy's optimized implementation. But you can indeed see that the complexity of numpy's implementation is $O\left(n^3\right)$ even from your plot. multiplying n = 400 takes roughly 0.1 seconds. And then multiplying $2 * n = 800$ should take 8 times more. Which it does according to the plot.

Having said all that, I disagree with the grading rubric:

> How does the time complexity of your solution involving matrix multiplication in part (a) compare to your friend's algorithm?

The question doesn't state anything about being practical or not. Existing theoretical matrix multiplication algorithms are faster than $O\left(n^3\right)$, one can't argue with that.

Hi Alenaaa,

My posting was referring to the rubric. It is true the question does not ask about being practical, but the rubric does: "(2 points): Recognizes that **in practice**, matrix multiplication will be no faster than the friend's algorithm."

If they said than in theory there is no difference I would agree. But **in practice** there is a (significant) difference and therefore the rubric seems to be wrong. Of course you can do better than the triple loop algorithm, but that's what they are asking us to compare, so that's what I compare.

I found multiple times that in practice, you are better off using matrix algebra than loops.

Best,

```
Floren
```

posted 2 months ago by **FlorenGS**

...

Let me try again, Floren.

We've been ask to compare algorithms not implementations. Hence, we can compare only time complexity, but not time.

And you've actually compared them in practice. What your plot shows is that both naive implementation and numpy implementation have $O(n^3)$ complexity **in practice**. How you can see it? Both sets of points are approximated by a cubic function (of the length of the input). Are the two functions different? Yes, they are. And that's why the **time** (but not time complexity) for both implementations is different. But nevertheless the time is cubic in both cases. Meaning that even in practice the **time complexity** of both algorithms is the same.

What's confusing in the answer formulation, in my opinion, is that the word "faster" is used. I wouldn't use it in combination with "time complexity" cause it's very misleading (again, two different implementations of one algorithm can have significantly different running times but would have the same time complexity, obviously), I'd rather use the word "better" maybe or try making clearer statements.

Also, **time complexity** disappears from the grading at all (which probably was the main cause of your question, and I understand, it's indeed very misleading, but it still kinda indicates that we're comparing algorithms not implementations and therefore we can only compare time complexity not time)

> (2 points): Recognizes that in practice, matrix multiplication will be no faster than the friend's algorithm.

but it re-appears in the example solution

> Thus, the time complexity of the matrix multiplication is no faster than the friend's algorithm in practice.

Hopefully, this answers your question about the nature of things.

As for the grading rubric I think that it doesn't have the best wording (+ it answers what wasn't really asked, as I said earlier) but one can't say that it's incorrect, and your plot actually proves that it's correct, as explained above.

posted 2 months ago by **alenaaa**

...

Let's agree that the wording they used is far from ideal and that the literal meaning of this is incorrect: "[...] in practice, matrix multiplication will be no faster than the friend's algorithm." because in the real world, using matrix multiplication will save you a lot of time compared to a triple nested loop.

And I do agree with you that the time complexity of both approaches is, effectively, the same: $O(n^3)$.

Cheers,

```
Floren
```

posted 2 months ago by **FlorenGS**

...

Totally!

posted 2 months ago by **alenaaa**

Add a comment

Marc_Svensson
2 months ago

+

...

Strassen's algorithm is both non galactic, asymptotically $\mathcal{O}(n^{2.807})$ and non asymptotic proven more efficient than the naive approach for n as small as 512

< Previous Next >

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

Connect

Blog
Contact Us
Help Center
Media Kit
Donate

GET IT ON Google play
Download on the App Store