Discussion Forums

Week 4

← Week 4

Problem: Integrated Circuit Design

Alexander S. Kulikov Instructor Week 4 · 4 years ago

Please use this thread to discuss Integrated Circuit Design problem (make sure to review forum rules before posting).

0 Upvotes Reply Follow this discussion

Earliest **Top** **Most Recent**

T **Timofii** · a month ago

Tip for you my fellow programmers: sort pairs of post numbers and vertices in advance before processing them in order to find all the scc (so like not do [find the next vertex with the largest post number which is not processed yet] at each step but have a sorted array of {post number, vertex} and just take the next not processed yet vertex at each step)

0 Upvotes Reply

JA **Jatin Agarwala** · 2 months ago

Failed case #14/36: Wrong answer

(Time used: 0.00/1.00, memory used: 31748096/1073741824.)

I am using C++, DFS followed by Kosaraju's algorithm to find SCC and then taking them in reverse order. I have tried all sorts of corner cases and have no idea how to proceed.

0 Upvotes Reply

CY **Chenyun Yang** · 2 months ago

I have tried all cases showed in week 4 forum but still don't know why I stuck at case 5(wrong answer). Could anyone give me some test numbers to do that? Thanks!

0 Upvotes Hide 1 Reply

JA **Jatin Agarwala** · 2 months ago

Same here.

Tell me if you figure it out. Thanks!

0 Upvotes

Reply

Reply

AJM · 2 months ago

Using Python3, implemented Kosaraju's algorithm. Through all these courses, I've been doing BFS and DFS via stacks (iteratively) instead of recursively. I thought that was better performance wise (isn't that what people do in competitive programming?). This problem either proved that wrong, or provided an interesting exception.

With iterative DFS, my solution failed:

Failed case #8/36: time limit exceeded (Time used: 31.98/16.00, memory used: 31756288/1073741824.)

The only change I made was to make my two DFS calls recursive, and then add the lines suggested in the forums (code from plan_party.py: threading / recursion limit / thread stack size). And all the sudden:

Good job! (Max time used: 1.99/16.00, max memory used: 156225536/1073741824.)

0 Upvotes Reply

Christopher Walker · 3 months ago

I hate this. I've re-written this so many times. I've taken all these suggestions. I've been screwing with this all weekend and still exceeds the time limit. I don't know how to make it any faster. I'm down to micro optimizations now, but I'm like double the time limit. I've tried so many things. This is horrible. This is too hard or I'm too dumb. This little box popped up and said "the average student took 2 and a half hours to do this. Was this helpful?" Heck no that isn't helpful! I've got like 20 hours into this and no closer to an answer than when I started. I won't be able to sleep. This is the worst assignment I've ever had. I hate programming. I'm too dumb to do this. I give up. I will sell pencils on the street corner. Anyone want to buy a pencil?

0 Upvotes Reply

Zuhaib Ul Zamann · 5 months ago

SUBFORUMS

All

Assignment: Programming Assignment 4

I Don't know who this might help but here are some of my suggestions in this assignment for python language:-

1) Don't use Iterative implementation for Finding the topological ordering as the implementation will be twice slower than recursive implementation. You might fall in some TLE errors in TC #16

2) Increase the stack size and the recursion depths:-

```
import sys

import resource

resource.setrlimit(resource.RLIMIT_STACK, (2**29,-1))

sys.setrecursionlimit(10**6)
```

3) Check the time consumption of each of the subparts using time.perf_counter().

4) Use sets for saving elements in each component. Lookups and finding an element in sets are on average O(1) whereas it is O(1) and O(n) in lists respectively.

Best of Luck

1 Upvote Reply

Greg G. · 6 months ago · Edited

```
Good job! (Max time used: 3.18/18.00, max memory used:
298340352/1073741824.)
```

Don't do this in Java! It's a total, incredible pain in the keyboard.

Edit: you can possibly do a recursive DFS using the starter code from the fun party assignment. (Thread's stackSize parameter is platform dependent but may work!) Try that first!

It took me >2 days, because I did the graph assignments in Python and thought... I don't know what I thought!

Recursive DFS won't (edit: may or may not) work with Java, you'll get a stack overflow on large datasets. To extend stack limits, you'd need a JVM command-line switch, but that's unavailable here. In Python, you can simply issue a command.

Also, stress testing is a must here, you'll have to generate lots of test data with 2 variables and 3 clauses. That will contain most edge cases and you can still 'debug' them with pen & paper so then you can later focus on performance on large datasets. Fortunately, the naive algorithm is already included for Java.

(I know stress testing is good, but I usually only do it as a last resort since it takes hell a lot of time. It feels good when it runs though!)

So we need iterative DFS. I looked around and found some solutions involving 2 stacks, but that algorithm also broke down when encountering cycles and whatnot - all the examples were for trees.

After several tries to tweak it, I gave up and rather recreated the "function stack" in a stack variable. So here you push the complete state to the stack - a tuple of a vertex number and the number of neighbors already processed.

Here's the iterative pseudocode that follows the algorithm explained by Dan:

```
explore(root)
    create stack
    add (root, 0) to stack
    visited = empty list // a list of visited vertices for finding SCCs

    while stack is not empty
        (vertex, processedNeighbors) ← pop from stack

        if !isProcessed(vertex)
            // essentially previsit
            add vertex to visited
            isProcessed(vertex) ← true

        if processedNeighbors < number of vertex's neighbors
            processedNeighbors++
            push (vertex, processedNeighbors) to stack
            nextNeighbor ← 'processedNeighbors' neighbor of vertex

            if !isProcessed(nextNeighbor)
                push (nextNeighbor, 0) to stack
        else
            // postvisit
            add vertex to postOrder list

depthFirstSearch()
    set isProcessed to all false
    for each vertex
        if !isProcessed(vertex)
            explore(vertex)
```

So this worked pretty fine, but took hell of a long time to implement all pieces properly. There are lots of moving parts, and it's especially hard to debug this, because you'll need to manually draw the graph and follow SCC search.

The silver lining was that once I got rid of all the failing cases with small datasets, it already ran fine for large inputs, too, and the submission was accepted. Basically you just need to avoid hidden O(n^2) iterations or algorithms.

I also created this SO answer to share this useful piece of preudocode with others.

1 Upvote Hide 2 Replies

SP **Savvas Pitsillos** · 6 months ago

Greg I am having the same issues with you. I will try to use iterative DFS with Java because a working code does not pass case 19

0 Upvotes

Greg G. · 6 months ago

At least sharing my code was not completely useless. :)

0 Upvotes

Reply

Reply

RR **Ravishankar Rajagopalan** · 10 months ago · Edited

Was doing in Python. Stuck with test case failing test case #31. Finally the following did the trick:

```
import sys
import threading
import resource

resource.setrlimit(resource.RLIMIT_STACK, (2**29,-1))
sys.setrecursionlimit(10**6)

```

Thanks to https://stackoverflow.com/questions/5061582/setting-stacksize-in-a-python-script

Must add though, that the quality of videos and support needs a BIG upgrade.

1 Upvote Reply

Andrei Burgasov · a year ago

Hi to all!

I got stuck on #22/36. And have no idea what's problem is?.. Can somebody share his experience if you had problems with #22?

History is (C++):

#4/36 WA — Kosaraju with recursion and containers like maps and sets (probably recursion overflow)

#16/36 TLE — Kosaraju without recursion but with containers (map and sets insertions etc. are slow)

#22/36 WA — Kosaraju w/out recursion and only vectors used with direct access

Checked shared #23 by my program and with online web solver — same results (UNSAT), so I think #23 should be passed. But can't figure how to deal with #22...

0 Upvotes Hide 1 Reply

Andrei Burgasov · a year ago

UPD: solved. Just rewritten from scratch again.

0 Upvotes

Reply

Reply

KK **Козлов Никита Константинович** · a year ago · Edited

Hi, I also got problems with test #16 (time limit). Problem arose due to the fact that when compiling a DAG from the original graph, I ran over all the edges and searched for their vertices in SCC, which required O (E*V). Instead, I've created an array of length V and assigned to each vertex it's SCC number (only O(V)). And the algorithm passed all the tests.

0 Upvotes Reply

MA **Maksadbek Akxmedov** · a year ago · Edited

Good job! (Max time used: 1.97/16.00, max memory used: 216498176/1073741824.)

Finally, passed it. Stuck on on the #16 test case with "wrong answer" verdict. But setting the threading and recursion limits to greater values solved it:

```
import sys
import threading


sys.setrecursionlimit(10**6)
threading.stack_size(2**26)

...

threading.Thread(target=main).start()
```

1 Upvote Reply

Hidetake Takahashi · a year ago

For computing SCC, the algorithm from "Algorithm on Graphs" course is enough to pass. But need to be careful when assigning Literals. Make sure that the code doesn't have O(n2) loop.

0 Upvotes Reply

MZ **Mark Zakharov** · a year ago · Edited

I am having troubles understanding what I am doing wrong in my Python 3 implementation. I am using recursive Kosaraju's algorithm (a sequence of recursive DFS's on normal graph while building up finish time stack and then a sequence of recursive DFS's on transposed graph while checking for containment of v and -v in the resultant SCC) and the complexity of my algorithm is linear, yet it times out at TC#16, taking 32 seconds instead of 16 allowed. I have read every single comment on this thread and I am really stuck. Can anybody tell me how to determine where the bottleneck is? Is my situation even possible?

0 Upvotes Hide 1 Reply

MZ **Mark Zakharov** · a year ago

Now it's fine on time scale, but I get a "wrong answer" on TC#16, even though the recursion limit and stack size are set :(

0 Upvotes

Reply

D **Dmitry** · a year ago · Edited

Hello! I am struggling with case 16. Initially, I had "wrong answer" but after trying to improve my solution using advises related to this problem from this discussion I came to a "time limit". However, case 17 which is rather big doing well locally (less than 3 seconds).

0 Upvotes Reply

Mahmoud Magdy · 2 years ago

I can't pass test case #23

I'm using Java with Tarjan's SCCs algorithm (recursive), any piece of advice will be appreciated

0 Upvotes Hide 3 Replies

Alexander S. Kulikov Instructor · 2 years ago

Mahmoud, let me share the test 23 with you:
https://www.dropbox.com/s/ouqjbsi3fyxk8w4/23?dl=0

1 Upvote

Mahmoud Magdy · a year ago

I passed the problem, and the course

0 Upvotes

Alexander S. Kulikov Instructor · a year ago

Mahmoud, congratulations!

0 Upvotes

Reply

Reply

Reply

Reply