Class 14: Concurrency

6.102 — Software Construction
Spring 2024

Infinite factorials

Exercise:  yellkey.com/**recently**

Nanoquiz:  yellkey.com/**back**

Open `factorial.py` and `factorial.ts` side by side
They are both supposed to compute an infinite sequence of factorials starting from 9000!

Implement the TODOs in:

- `range_to_infinity` (Python)
- `rangeToInfinity` (TS)

Do not use `range()` or `inf` or `POSITIVE_INFINITY` in your implementations

Run both versions to confirm that they now compute factorials starting from 9000!

- `python3 src/factorial.py`
- `npm run factorial`

Using threads and workers

Now open `threads.py` and `workers.ts` side by side, and run the code to see what it does

- `python3 src/threads.py`
- `npm run workers`

Notice that `computeFactorials(4000)` runs first, computing 4000!, 4001!, 4002!, …
then `computeFactorials(7000)` runs next, computing 7000!, 7001!, 7002!, …

Change the **Python code** first, so that `computeFactorials(4000)` and `computeFactorials(7000)` run **concurrently**

- **don't** change `computeFactorials` or any functions it depends on
- hint: what is the fewest number of new threads you need?

Using threads and workers

Now open `threads.py` and `workers.ts` side by side, and run the code to see what it does

- `python3 src/threads.py`
- `npm run workers`

Change the **Python code** first, so that `computeFactorials(4000)` and `computeFactorials(7000)` run **concurrently**

- **don't** change `computeFactorials` or any functions it depends on
- hint: what is the fewest number of new threads you need?

Now change the **TS code**, so that `computeFactorials(4000)` and `computeFactorials(7000)` run **concurrently**

- hint: create a `Worker` that runs `./dist/workers.js` (yes, `.js`)
- is it interleaving or not?
  -

Using threads and workers

Now open `threads.py` and `workers.ts` side by side, and run the code to see what it does

- `python3 src/threads.py`
- `npm run workers`

Change the **Python code** first, so that `computeFactorials(4000)` and `computeFactorials(7000)` run **concurrently**

- **don't** change `computeFactorials` or any functions it depends on
- hint: what is the fewest number of new threads you need?

Now change the **TS code**, so that `computeFactorials(4000)` and `computeFactorials(7000)` run **concurrently**

- hint: create a `Worker` that runs `./dist/workers.js` (yes, `.js`)
- is it interleaving or not?
  - hint: try sorting the messages by timestamp, using `npm run workers | sort`

Race conditions

`bank-main.ts`, `bank-cash-machine.ts`, and `bank-account.ts`
implement the filesystem shared-memory cash machine example from the reading

- currently uses just one worker (`NUMBER_OF_CASH_MACHINES` = 1)
- run the code using `npm run bank` and understand what it does

Now change `NUMBER_OF_CASH_MACHINES` to 2

- first predict: what do you expect might happen?
- then run the code

What is the actual problem?
→ Take notes at the bottom of `bank-account.ts`

What is the race condition?
What does a bad interleaving look like?

Try to fix… or at least work around… the problem