There are several limitations to the number of processes we can fork. According to the man page of the `fork()` call the following errors may occur due to various limits: Our `fork()` call was failing due to errno 11 which corresponds to EAGAIN.

```
EAGAIN A system-imposed limit on the number of threads was
encountered.  There are a number of limits that may trigger this
error:

*  the RLIMIT_NPROC soft resource limit (set via setrlimit(2)), which
limits the number of processes and threads for a real user ID, was
reached;

*  the kernel's system-wide limit on the number of processes and
threads, /proc/sys/kernel/threads-max, was reached (see proc(5));

*  the maximum number of PIDs, /proc/sys/kernel/pid_max, was reached
(see proc(5)); or

*  the PID limit (pids.max) imposed by the cgroup "process number"
(PIDs) controller was reached.
```

The most relevant to us are the *pid_max*, *RLIMIT_NPROC* and *pids.max*. They can be found using the following commands:

1. `cat /proc/sys/kernel/pid_max`
2. `cat /proc/self/limits`
3. `cat /sys/fs/cgroup/pids/user.slice/user-$(id -u).slice/pids.max`

Theoretically we cannot create more processes than these limits. So as an upper bound we must have ***r1*c2 <= min(pid_max, RLIMIT_NPROC, pids.max)***.

On the system we tested the code on these limits were 32768, 62780 and 10813 respectively. So while it should be possible to create as many as 10813 processes we observed that the maximum forks could not go above 9000. The `fork()` call failed whenever *r1*c2* went roughly above 9000. This is primarily because of other processes already running on the system. There is another parameter pids.current which we monitored while stress testing our program. Before the start of the program, already around 1800 processes were running and the value peaked to 10813 before the `fork()` call failed. So we were roughly able to fork around 10800 - 1800 = 9000 processes.

So depending on the pids.current at any time we can multiply matrices with dimensions *r1,c1* and *r2,c2* as long as ***r1*c2 <= pids.max - pids.current***.

References:

- https://stackoverflow.com/questions/29605502/maximum-number-of-children-processes-on-linux
- https://www.kernel.org/doc/man-pages
- https://www.kernel.org/doc/Documentation/cgroup-v1/pids.txt