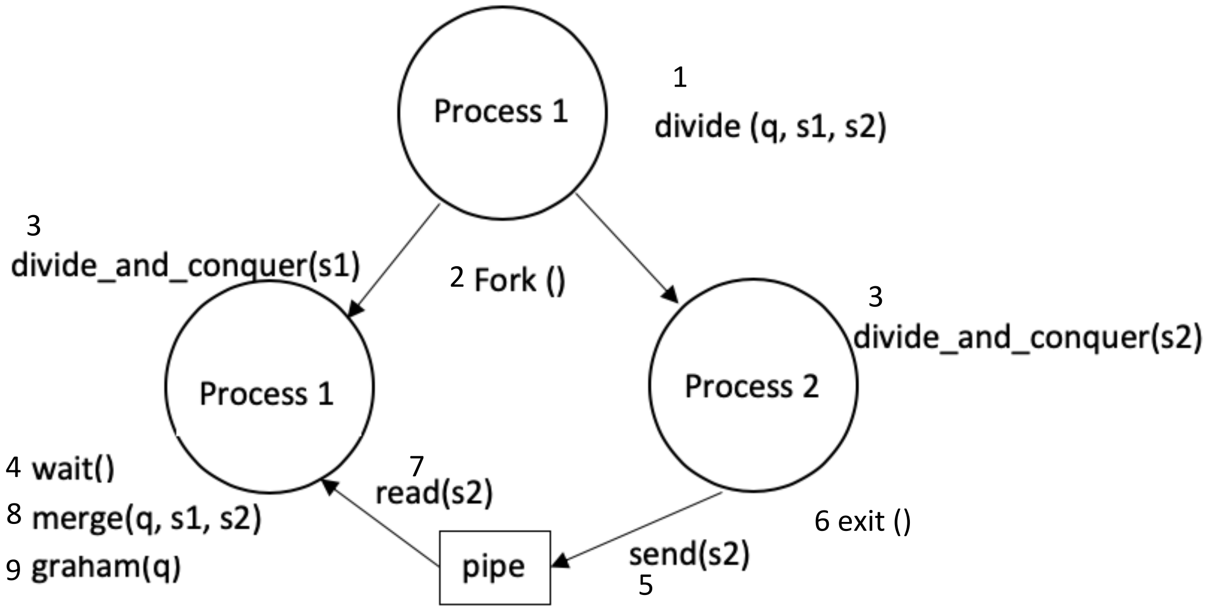
Program 1: Convex Hull Report

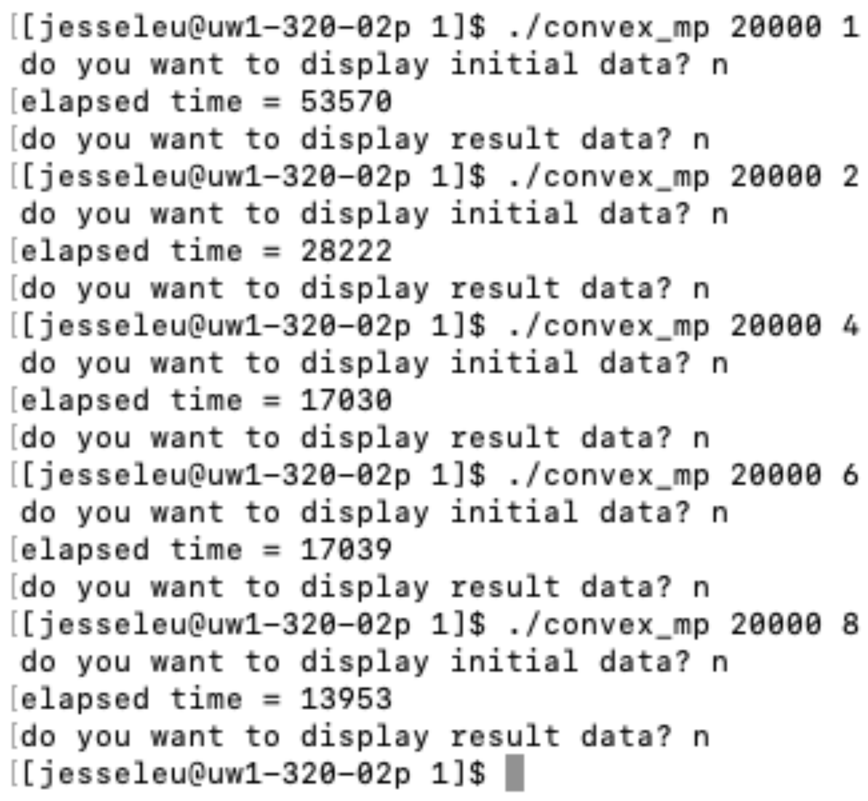
Design document:

The main process will first divide queue into two sub-queues then create a pipe and fork a child process. After forking a child process, the parent process will handle (call the divide_and_conquer recursively) the left side, and the child process will handle (call the divide_and_conquer recursively) the right side parallelly. After the child process finishes working, it will send the result through the pipe and exit. The parent process will wait for the child process to finish, then read the result of the child process from the pipe. The parent process will merge the left result and right result in a queue and call the graham algorithm.



Discuss Results:

1. **Analysis of the effectiveness of your parallelization.**



```
[[jesseleu@uw1-320-02p 1]$ ./convex_mp 20000 1
 do you want to display initial data? n
[elapsed time = 53570
[do you want to display result data? n
[[jesseleu@uw1-320-02p 1]$ ./convex_mp 20000 2
 do you want to display initial data? n
[elapsed time = 28222
[do you want to display result data? n
[[jesseleu@uw1-320-02p 1]$ ./convex_mp 20000 4
 do you want to display initial data? n
[elapsed time = 17030
[do you want to display result data? n
[[jesseleu@uw1-320-02p 1]$ ./convex_mp 20000 6
 do you want to display initial data? n
[elapsed time = 17039
[do you want to display result data? n
[[jesseleu@uw1-320-02p 1]$ ./convex_mp 20000 8
 do you want to display initial data? n
[elapsed time = 13953
[do you want to display result data? n
[[jesseleu@uw1-320-02p 1]$ █
```

The performance improvement with **two** processes applied to 20,000 points

53570 / 28222 = 1.8981…

The performance improvement with **four** processes applied to 20,000 points

53570 / 17030 = 3.1456…

The performance improvement with **eight** processes applied to 20,000 points

53570 / 13953 = 3.8393…

The lab has 8 CPU, so if I run ./convex_mp 20000 16, the performance does not improve. Also, if I run with the number which is not 2^n, the performance improvement is not obvious. For example, the performance improvement with **six** processes applied to 20,000 points:

53570/17039 = 3.1439…

The performance is almost the same as four processes.

2. Possible performance improvement of your program

The non 2^n process does not improve much compared to the previous 2^n number. I checked the pids, and it seems the program creates the correct number of processes. It is one possible performance improvement for me to think. Another possible performance improvement is maybe I can delete and combine some of the codes to make the code shorter because there is some repeated code. Maybe the program will be faster if I try to refactor more.

3. **Limitations of your program.**

1. The non 2^n process does not improve much compared to the previous 2^n number.
2. If the leaves variable is too big for the number of points, the program will not create the exact number of processes. For example, if I call ./convex_mp 10 10, the program will only create 9 processes total(includes the parent process) instead of 10 processes.
3. The performance also depends on the number of CPUs. My computer only has 2, so the performance does not improve as the leaves increasing.
4. The computer might get stuck if the number of points is too big depends on the computer.
5. if the number of points is not big enough, the more process the program will be slower. For example, convex_mp 100 4 is slower than convex_mp 100 1.