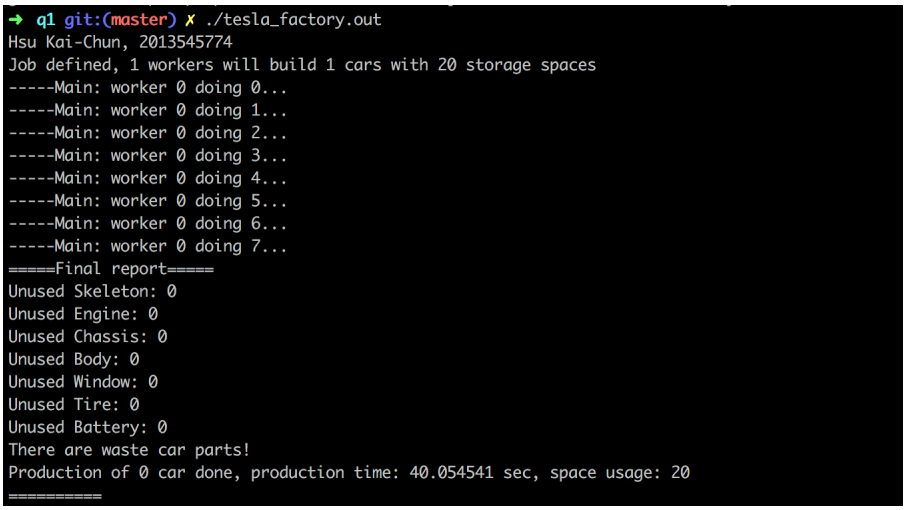
Q1.



Q2.

I use an array to store the threads (i.e. threads[num_workers]), and another array to store the work packs (i.e. wpack[num_cars * 8]) representing all the tasks to be done by threads.

Then, I have a while loop that keeps producing cars if there are not enough of them. In each while loop, there is another for loop to loop through available workers/threads and take up 8 of the available threads each time to produce a new car (i.e. after 8 loops in the for loop, car number++). Stop both the while loop and for loop if the amount of cars are already enough in any given entry point.

And at the end of the while loop, pthread_join to wait for all the threads to complete in a while loop.

Please kindly refer to the code submitted, I have also made some comments along the code.

And here is the screenshot of the output of `run.sh`:

```
h1354577@workbench q2> ./run.sh
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 1 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 1 car done, production time: 15.000556 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 4 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 4 cars done, production time: 60.003243 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 16 workers will build 4 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 4 cars done, production time: 30.001314 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 32 workers will build 4 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 4 cars done, production time: 15.002677 sec, space usage: 40
==========
h1354577@workbench q2>
```

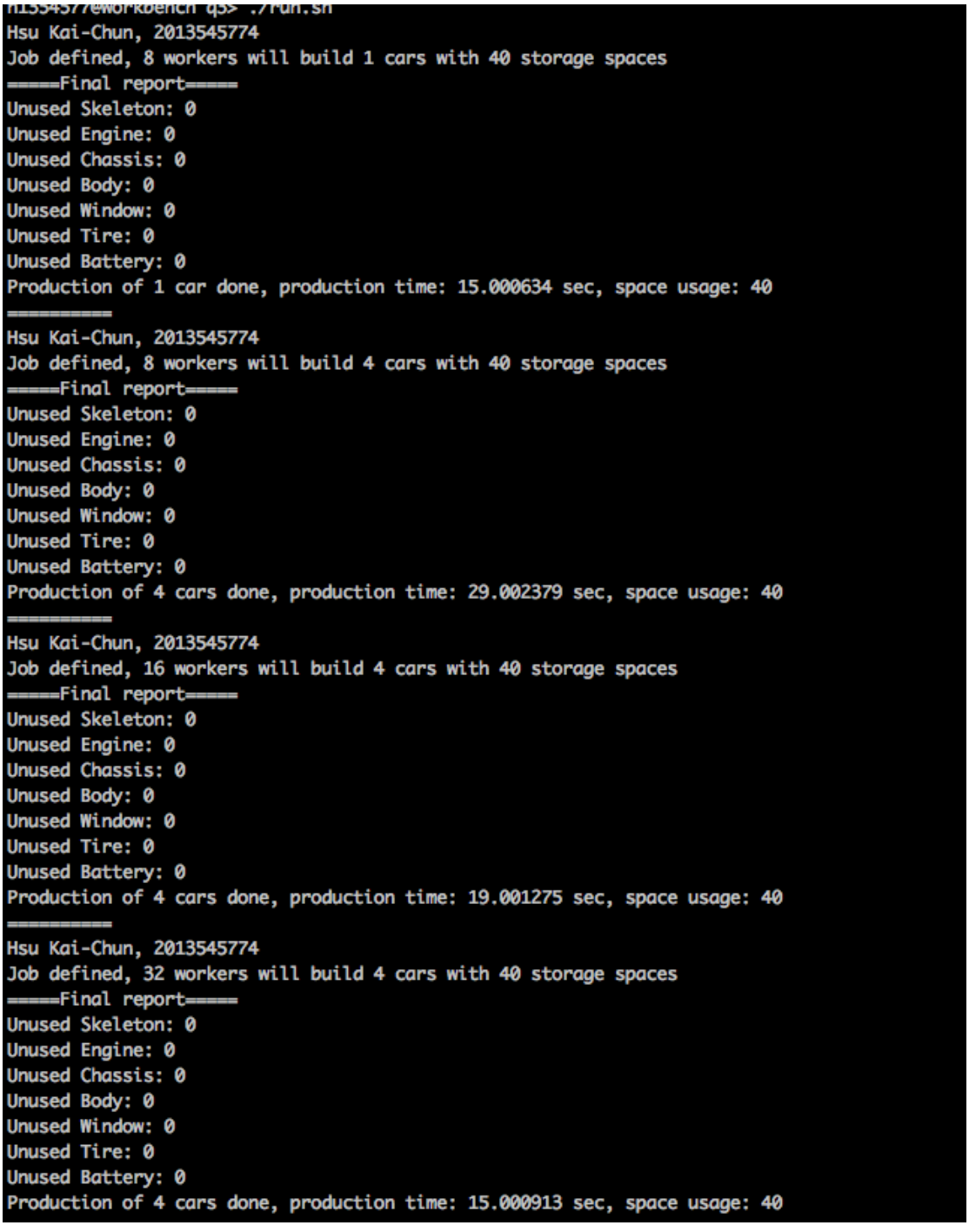
Q3.

The problem in Q2's approach was that it traverse through jobs to make car by car (i.e. job 0 -> job 7 for car 1 then job 0 -> job 7 for car 2 again and so on) and it is not the optimized order to execute the threads. This also causes the problem of producing unused units of car parts.

Therefore, in Q3, I try to prioritize the independent production processes earlier in the scheduling prior to the execution of multi-threading programs, i.e. let the

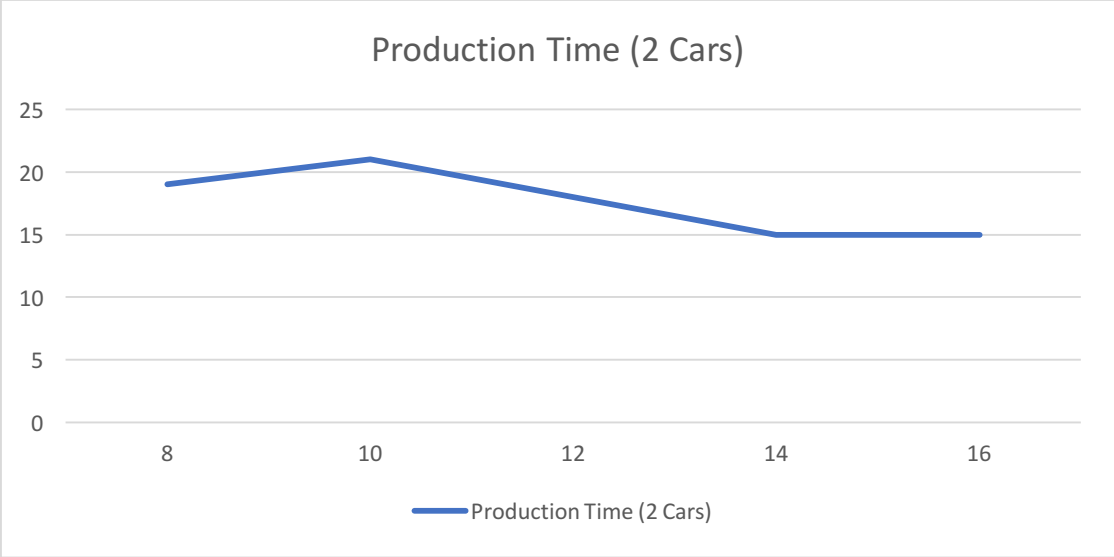
available workers to first produce skeletons of all cars, then engines of all cars, then chassis of all cars and so on, and leave the dependent jobs to later to better make use of multiple threads.

The result of the same ./run.sh is as shown below in the screenshot. Except for the 15 sec being already the optimal case, other cases all have significant improvement.



scalability

number of workers - run time (2 cars and given enough space)

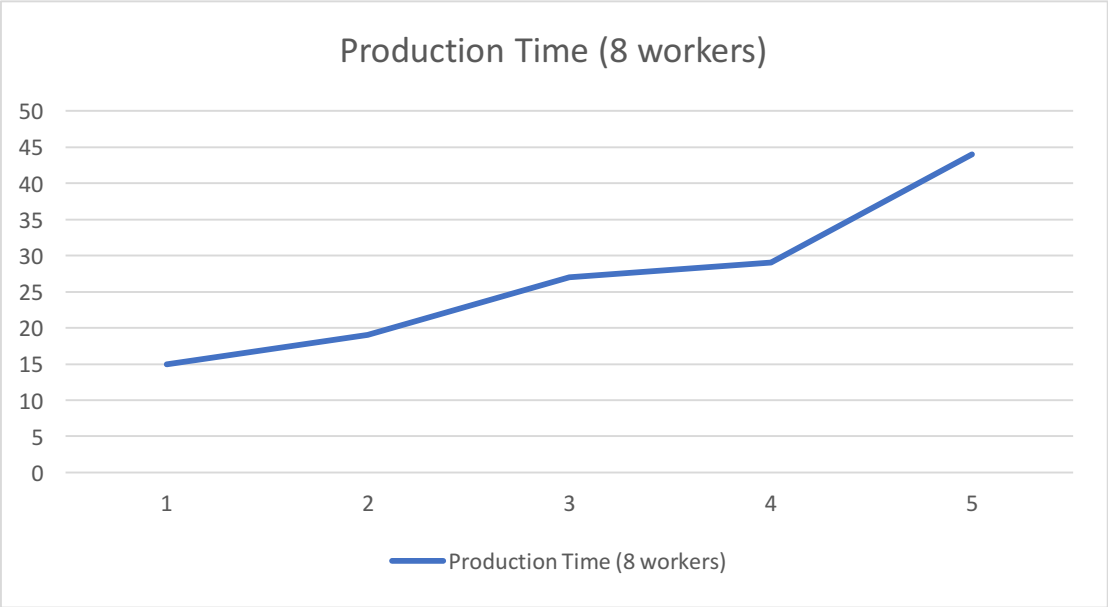


As also shown in the previous screenshot, the time spent for making two cars with 8 workers decreased from 30 sec to 19 sec. But it is also very interesting that when the number of workers increase to 10, there is actually a small drop in efficiency. That is because when there are 8 workers, in the first round they will do 2 times of job 0, 1, 2 and 6 (time spent: 5 sec) and in the second round they will do 2 times of job 3, 4, 5 and 7 (time spent: 14 sec), whereas when there are 10 workers available, in the first round, they will do 2 times of job 0, 1, 2, 4 and 6 (10 jobs, time spent: 5 -> 7 sec) and although they only need to do 6 jobs: two times of job 3, 5 and 7, the time spent is actually still 14 second, which means the total time spent just went from 19 to 21 in this case. This also means, we can consider holding some workers although they are available, i.e. although we have 10 workers, we might still want to run 8 threads in two rounds.

But other than this case, all other scenario has a positive correlation of the number of workers available and the production efficiency.

For the screenshot of data source of the graph, please kindly refer to Appendix 1

number of cars - run time (8 workers)



As shown in this graph, the correlation between cars to make and the execution time is no longer linear (i.e. making 1 car needs 15 sec, so making 2 cars needs 30 sec) and that is because of the new scheduling that is arranged even before jobs or threads are started.

For example, in Q2, making two cars with 8 workers would require all 8 workers to work 2 symmetry rounds:
Round1: {job 0, job 1, job 2, job 3, job 4, job 5, job 6, job 7} = 15 sec
Round2: {job 0, job 1, job 2, job 3, job 4, job 5, job 6, job 7} = 15 sec
⇨ Total 30 sec

But now, like aforementioned, making two cars with 8 workers would be working on two non-symmetry rounds:
Round 1: {job 0, job 0, job 1, job 1, job 2, job 2, job 6, job 6} = 5 sec
Round2 :{job 4, job 4, job 5, job 5, job 3, job 3, job 7, job 7} = 8 + 6 = 14 sec
⇨ Total 19 sec
For the screenshot of data source of the graph, please kindly refer to Appendix 2

Handle Deadlock

However, this way deadlock is also possible as I will be leaving a lot of car parts unassembled until job 3 or job 7 is finally processed. Therefore, in my implementation, I also handle these potential deadlocks by preventing them in the first place. In the scheduling part of my code, I also keep track of available_space before and after putting in next job. If there is no enough space, I will insert the assembling jobs first until enough spaces are released.

For example, in the case of making 4 cars with 40 spaces and 8 workers.

In the first round, according to my scheduling, the outcome will be:
{job 0, job 0, job 0, job 0, job 1, job 1, job 1, job 1}
⇨ Space left: 32

In the second round:
{job 2, job 2, job 2, job 2, job 6, job 6, job 6, job 6}
⇨ Space left: 24

In the third round:
Worker1 – job 4
Worker2 – job 4
Worker3 – job 4 – space left: 3
Worker4 therefore can't do job4 again, as it requires 7 spaces. And it will instead, insert job3 here to release spaces, until job 4 can be added again and so on.

Appendix 1

```
Job defined, 8 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 19.000977 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 10 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 21.002141 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 12 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 18.001217 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 14 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 15.001226 sec, space usage: 40
==========
Hsu Kai-Chun, 2013545774
Job defined, 16 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 15.000749 sec, space usage: 40
```

Appendix 2

```
Job defined, 8 workers will build 1 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 1 car done, production time: 15.000735 sec, space usage: 40
=========
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 2 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 2 cars done, production time: 19.000943 sec, space usage: 40
=========
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 3 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 3 cars done, production time: 27.002771 sec, space usage: 40
=========
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 4 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 4 cars done, production time: 29.002647 sec, space usage: 40
=========
Hsu Kai-Chun, 2013545774
Job defined, 8 workers will build 5 cars with 40 storage spaces
=====Final report=====
Unused Skeleton: 0
Unused Engine: 0
Unused Chassis: 0
Unused Body: 0
Unused Window: 0
Unused Tire: 0
Unused Battery: 0
Production of 5 cars done, production time: 44.008462 sec, space usage: 40
```