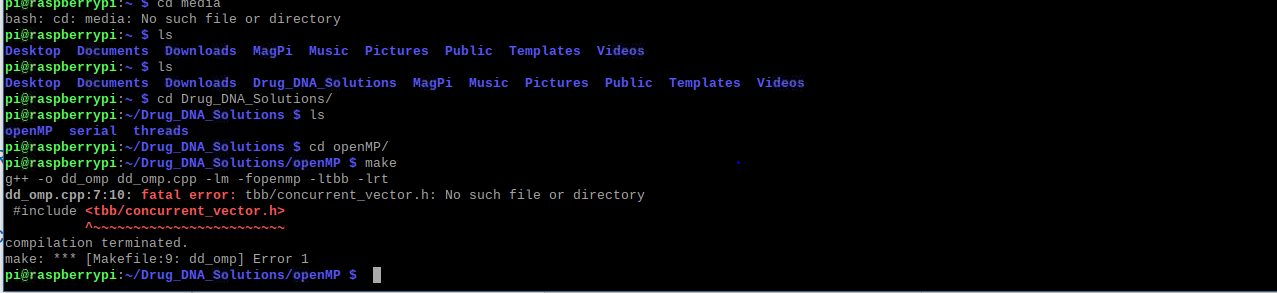
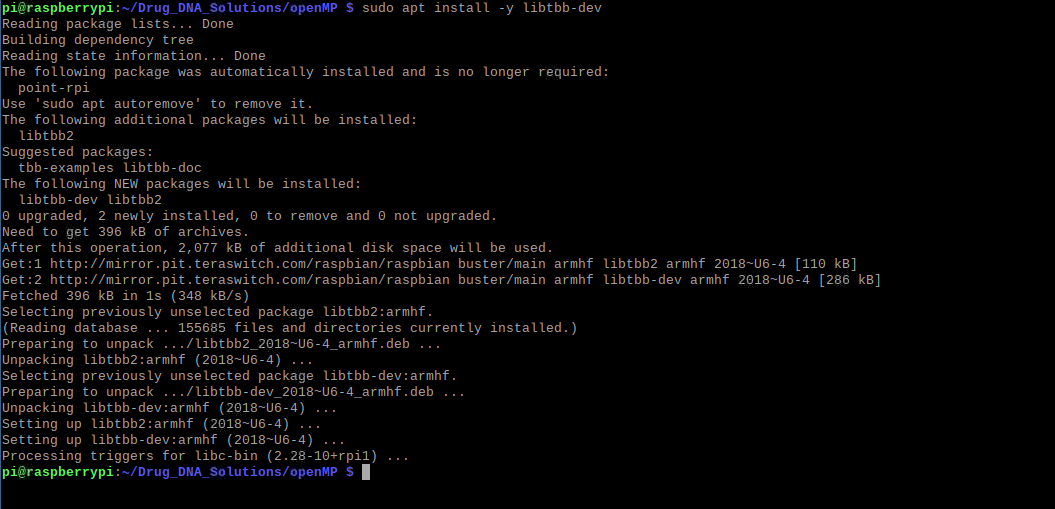
The serial program compiles alright, but the OpenMp had some issues.



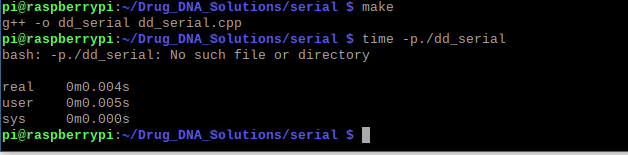
Issue ran into while trying to compile the openMP file. But according to the issue it occurs that the we need to install the file/directory because it cant find it.

So we install the libtbb-dev library.

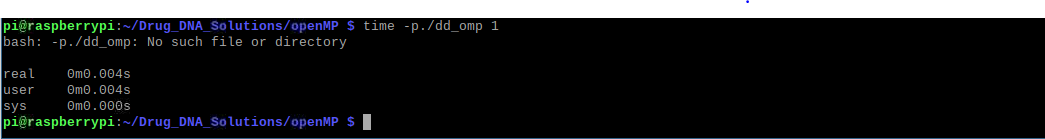


And we can finally compile the openMP program.

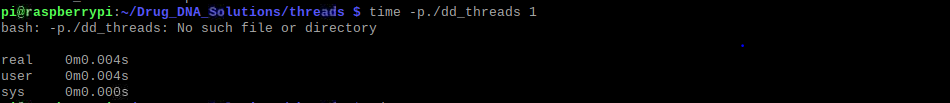
Measure Run-Time



Time -p ./dd\_omp 1

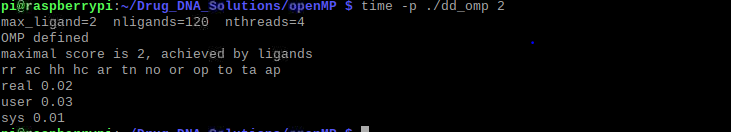


Time -p ./dd\_threads 1

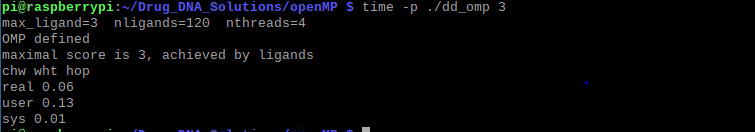


Use for real time:

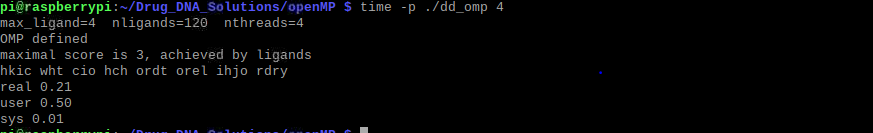
Omp 2



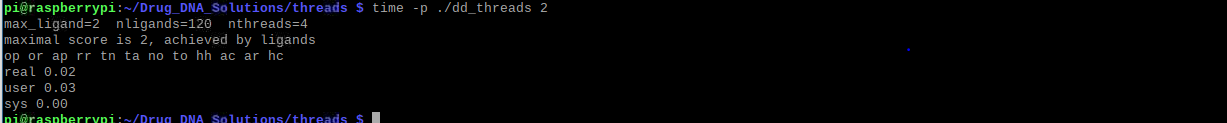
Omp 3

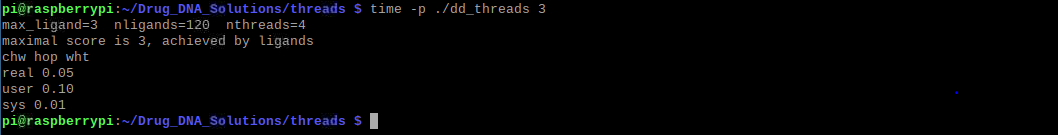


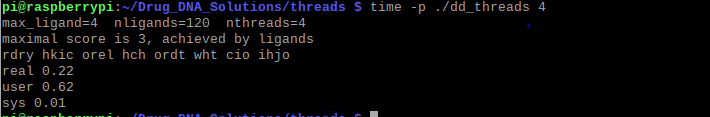
Omp 4



Threads 2,3,and 4







What approach is fastest?

* According to our results, the executions have similar results, but with the openMP, we are getting a little bit faster result than the reset.
* Perhaps, this is because of the structure of the openMP and its ability to divide executions between cores.

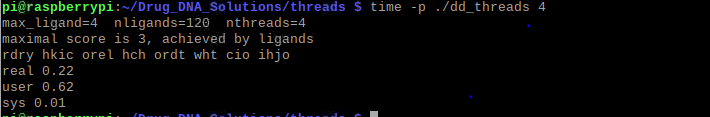
Determine the number of lines in each file. How does the C++ implementations compare to the openMP implementations?

* We can say that the C++ implementation is more detailed and simplified, but we have to understand the purpose of openMP is minimizing code while reducing run time complexity at the same time. But since the C++ implementation is more explanatory, in a way that other programmers can look into your program and suggest what you are writing.

Increasing threads to 5. What is the un time?

* Real – 0. 45
* User – 1.00
* Sys - 0. 3

Increase the maximum lingand and return each program



I don’t know if right, but run time is about the same as the rest.