# CS342 - Project 1

Çelik Köseoğlu - 21400196

I will be doing these tests on a  mobile grade dual-core Intel x86 processor. The computer is 6 years old and has a very slow processor. On a newer processor, 1000 lines are not enough to show a major difference, it is better for my test environment. There are some different scenarios to consider while writing this report. For example, the runtime for a one and two input files should be theoretically the same. There could be up to %20 error due to Intel's turbo boost technology which boosts the frequency of a single CPU core if a single-thread intensive application is being run. Also, keep in mind that different processors have different cache properties which may affect the result as well.

**Test A.** I will use the following scenarios to see how the number of processor cores affect the performance:

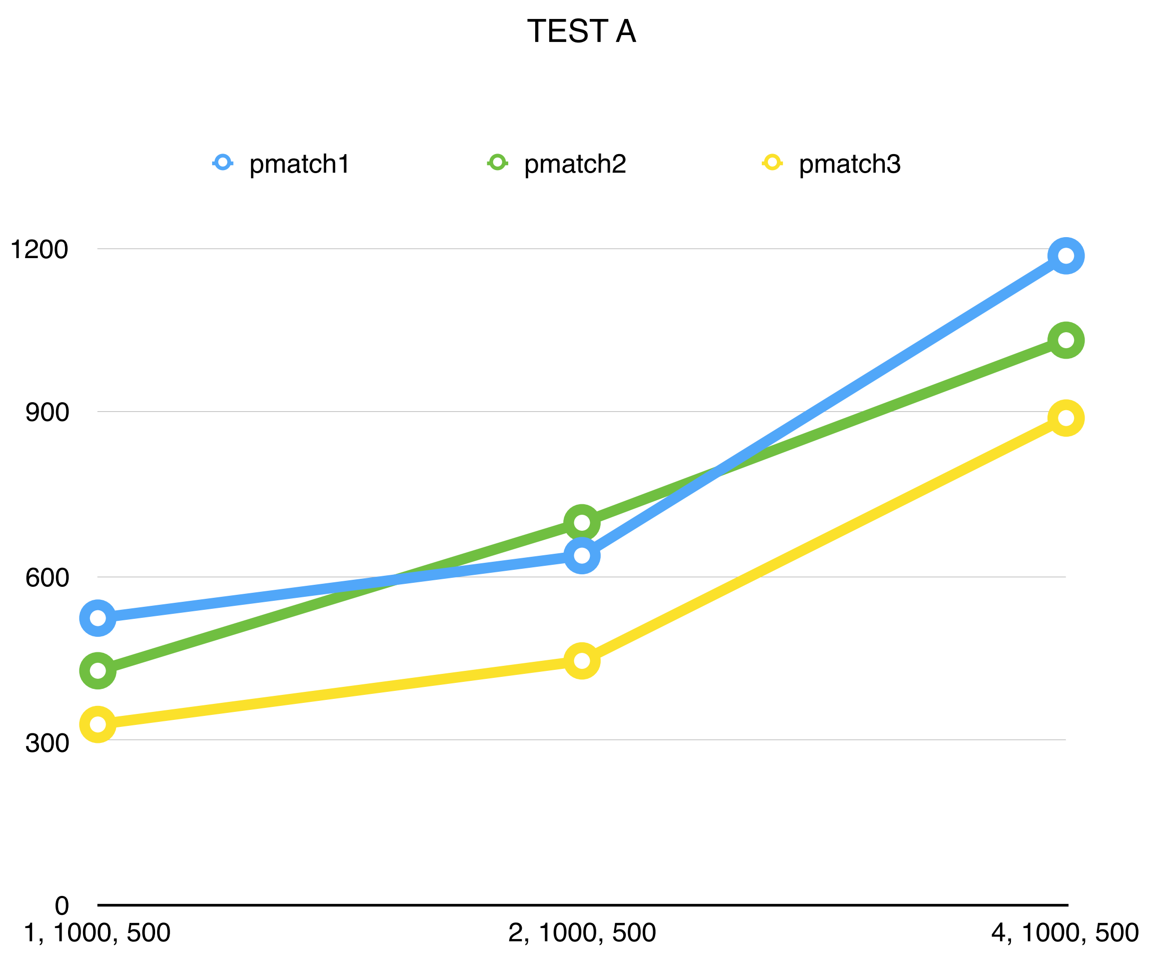
1) *Single input, 1000 lines, 500 matches*
2) *Two inputs, 1000 lines each, 500 matches each*
3) *Four inputs, 1000 lines each, 500 matches each*

**Test B.** And I will be using the following scenario to see how the number of matching lines affect the performance

1) *Single input, 1000 lines, 0 matches*
2) *Single input, 1000 lines, 500 matches*
3) *Single input, 1000 lines, 1000 matches*

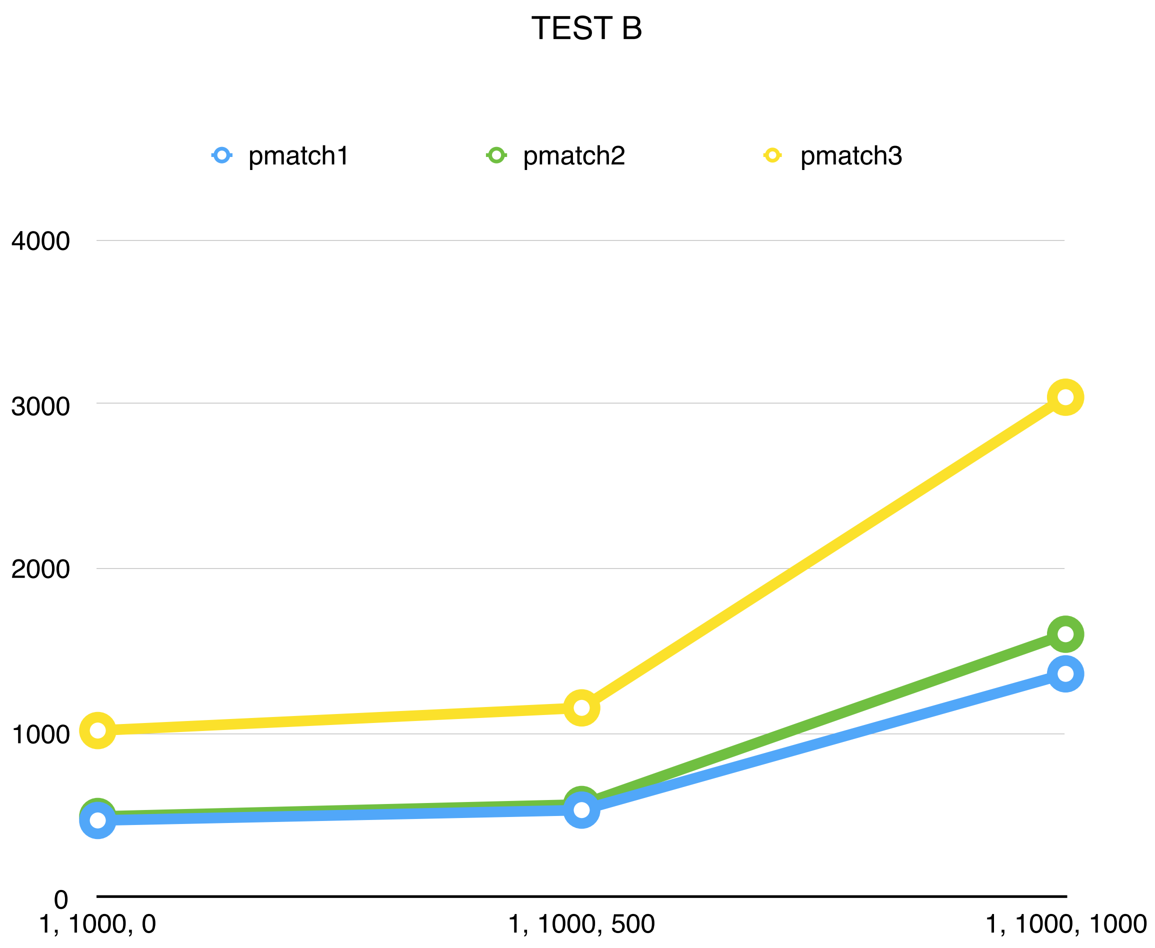
**Test C.** Now, I will change the input length and see how that affects the performance with the following tests:

1) *Single input, 1000 lines, 500 matches*
2) *Single input, 10000 lines, 5000 matches*
3) *Single input, 100000 lines, 50000 matches*



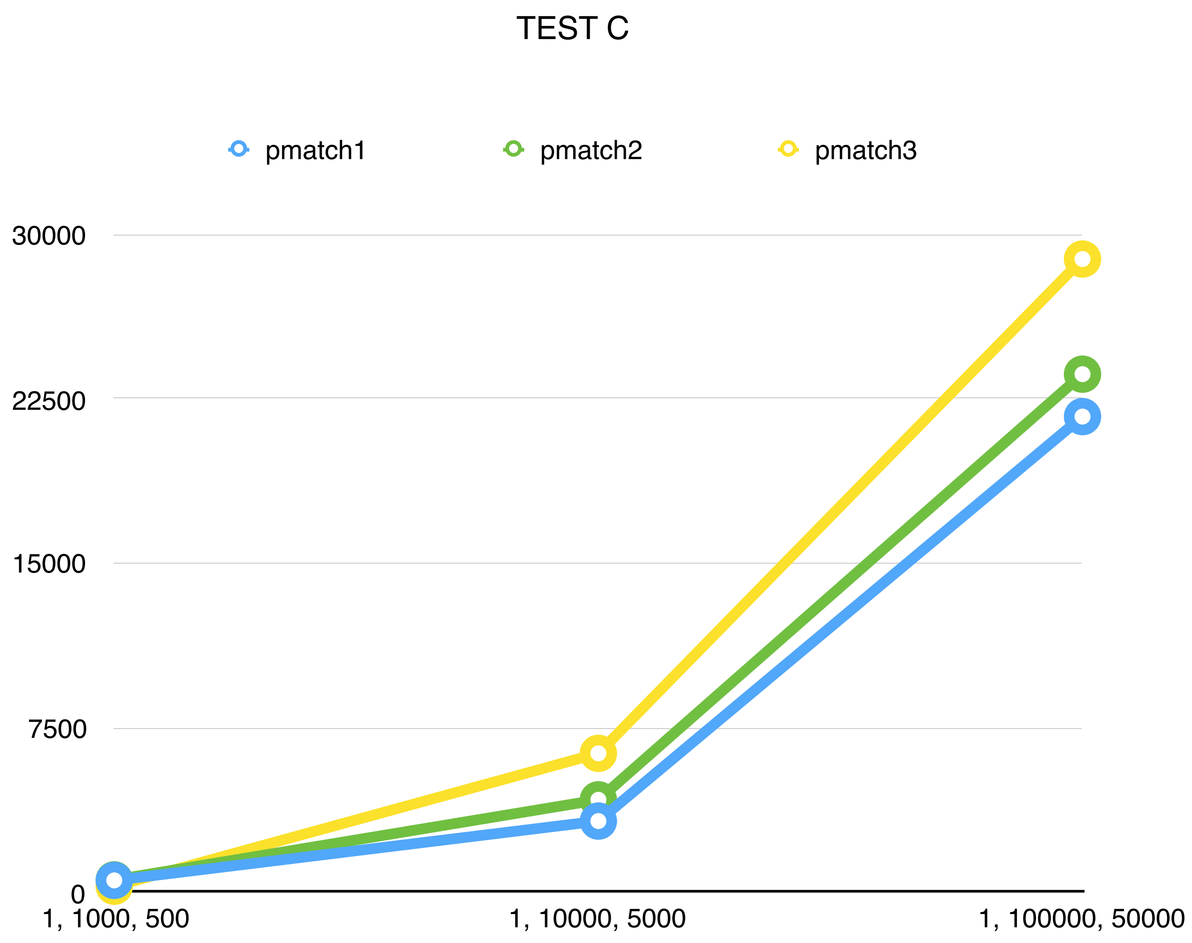
For my machine, these are the exact results that I expected. There is not a huge jump when we increase the input size from one to two. The small increase could be explained by Intel's Turbo Boost technology.

However, when we jump from two to four inputs, the jump is higher. Yes, again, Intel's HyperThreading technology helps in this case but since the CPU has two physical cores, this bigger jump is expected.



As we increase the number of matching lines in a single file, we need to add the matching results into our storage media. In the first program, this is a file. In the second and third programs, we write the intermediate values into the memory.  This brings an overhead. If no lines matched, we wouldn't need to do file or memory writes and this would increase performance as seen in the graph.

Another point about this graph is, pmatch3's runtime increases faster than the others when the matching input size increases. This is due to the linked list node creation overhead when a matching line is found.



In this test, we could see that increasing the number of inputs increase the runtime in a linear way. As the length of the input increases, we see that the runtime increases as well.

<u>Test Environment Specs:</u>
-Intel Core i5-2410M
-IntelHD 3000
-Corsair Force  LS SSD 120GB SATA 3