REPORT

A) Execution Time Experiment

I tested my program by using 5 different input files, each file has different number of lines and they are given as:

Input File 1	Input File 2	Input File 3	*Input File 4*	Input File 5
hi hi hi hello hi hi hi hi	hi hi hello helloankara alohellob	hi hello hi hi hello hi hello helloankara alohellob hellobro	babahellobaba helloanne hi hello hiiii hebebele hellobro hello	salala helloooo shellosss hebelee hellobro hi hello helloankara

Below chart shows the **Execution Times** of "psearch" program for different number of identical inputs

Data

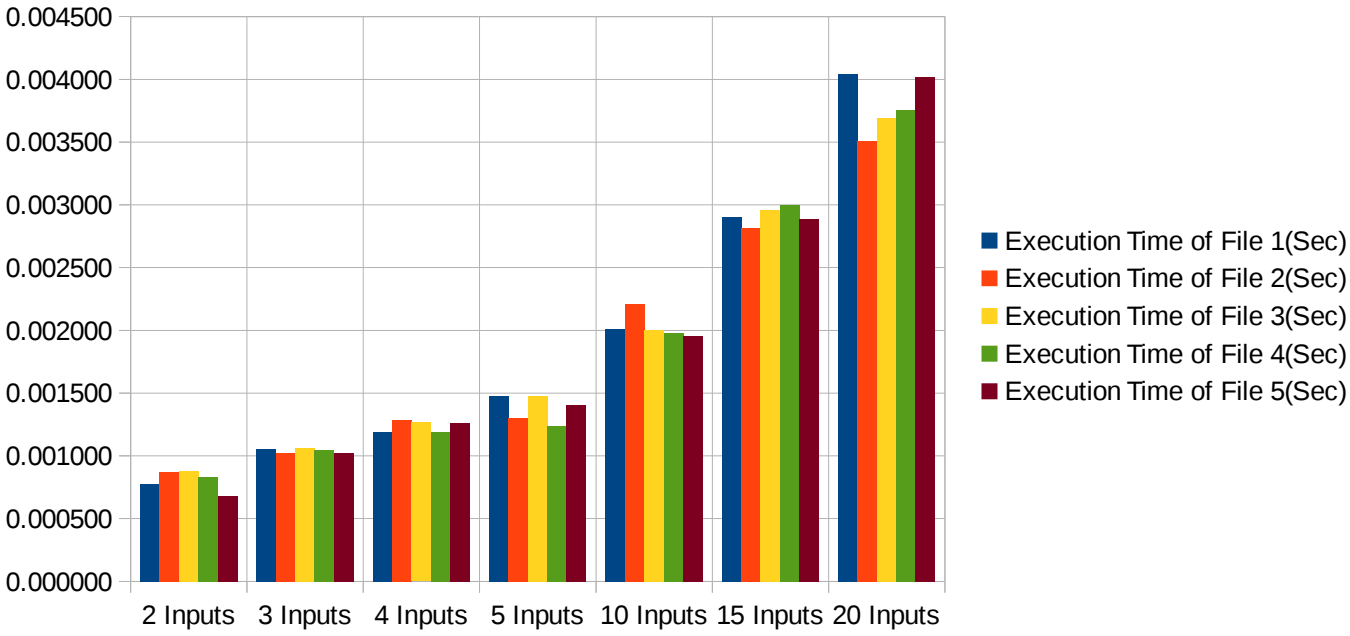


table of the above chart is given as:

Data Table

	Categories	Y-Values	Y-Values	Y-Values	Y-Values	Y-Values
		Execution	Execution	Execution	Execution	Execution
1	2 Inputs	0.000776	0.000869	0.000878	0.000830	0.000682
2	3 Inputs	0.001055	0.001024	0.001066	0.001045	0.001025
3	4 Inputs	0.001190	0.001282	0.001266	0.001192	0.001263
4	5 Inputs	0.001476	0.001302	0.001476	0.001235	0.001406
5	10 Inputs	0.002013	0.002207	0.002001	0.001975	0.001956
6	15 Inputs	0.002907	0.002818	0.002960	0.002999	0.002890
7	20 Inputs	0.004043	0.003513	0.003691	0.003753	0.004021

To find execution times, I used C <time.h> library and used clock() function.

I observed that; as the input number increases the execution time gets longer, as expected. Every time I re-run the same test code I got different execution time results but the results were close to each other so that I preferred to use the first result which I received from each test case.

B) Memory Usage Experiment

To find Memory Usage of the psearch program, I used valgrind's massif tool. An example:

```
batuhan@Batu-Lenovo:~/Desktop/HW2$ valgrind --tool=massif ./psearch hello 2 in5 in5 out
==5724== Massif, a heap profiler
==5724== Copyright (C) 2003-2013, and GNU GPL'd, by Nicholas Nethercote
==5724== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==5724== Command: ./psearch hello 2 in5 in5 out
==5724==
==5725==
==5726==
==5724==
batuhan@Batu-Lenovo:~/Desktop/HW2$
```

Then to see memory usage as I used massif's print command by referring the above result:

```
batuhan@Batu-Lenovo:~/Desktop/HW2$ ms_print massif.out.5724
--------------------------------------------------------------------------------
Command:            ./psearch hello 2 in5 in5 out
Massif arguments:   (none)
ms_print arguments: massif.out.5724
--------------------------------------------------------------------------------


    KB
2.164^                                                                      #
     |                                                                      #
     |                                                                    @:#
     |                                                                  ::@:#
     |                                                                  : @:#
     |                                                                 :: @:#
     |                                                                 :: @:#::
     |                                                                 :: @:# ::
     |                                                                 :: @:# ::
     |                                                                 :: @:# ::
     |                                                                ::: @:# :@
     |                                                                ::: @:# :@
     |                                                                ::: @:# :@
     |                                                                ::: @:# :@
     |                                                                ::: @:# :@
     |                                                              ::@::: @:# :@
     |                                                              : @::: @:# :@
     |                                                              : @::: @:# :@
     |                                                              : @::: @:# :@
     |                                                              : @::: @:# :@
   0 +-------------------------------------------------------------------------->ki
     0                                                                       139.1

Number of snapshots: 35
 Detailed snapshots: [3, 13, 15, 22 (peak), 32]

--------------------------------------------------------------------------------
  n        time(i)         total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
  0              0                0              0             0            0
  1        115,341                8              0             8            0
  2        118,633              592            568            24            0
```

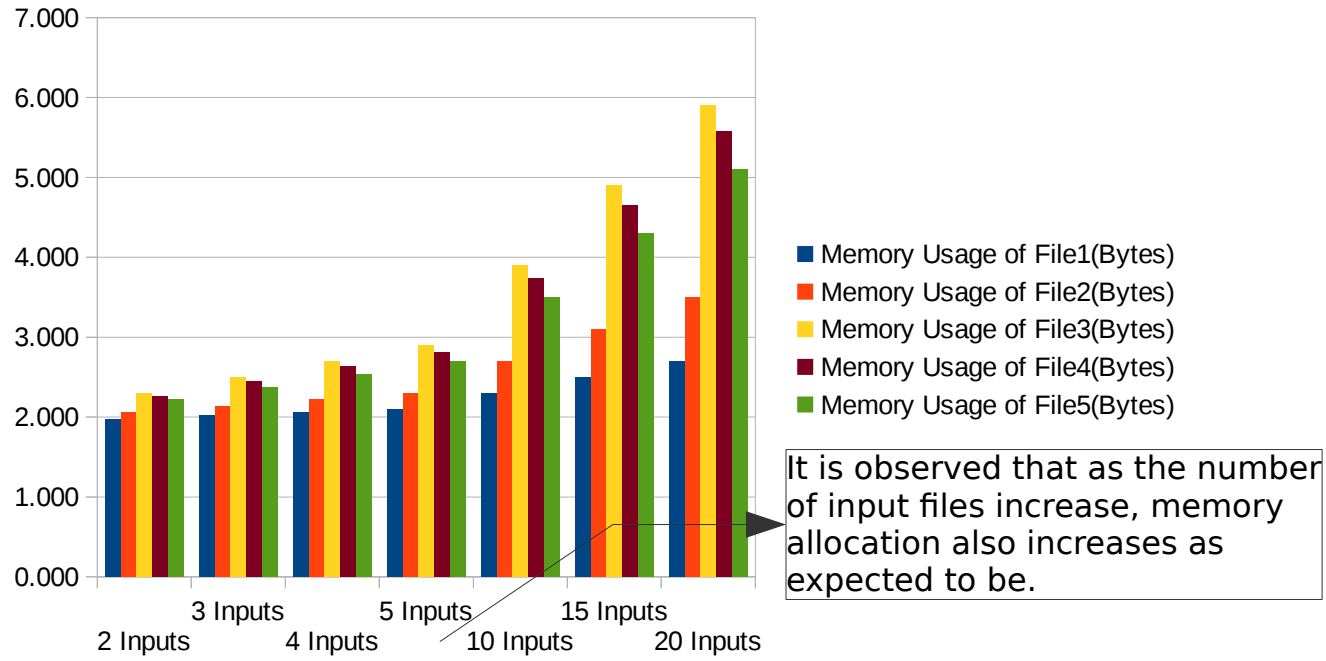
Above graph indicates that Detailed Snapshot No:22 is the peak which means that the program makes a memory consume peak at that location.

By looking n=22 we see that:

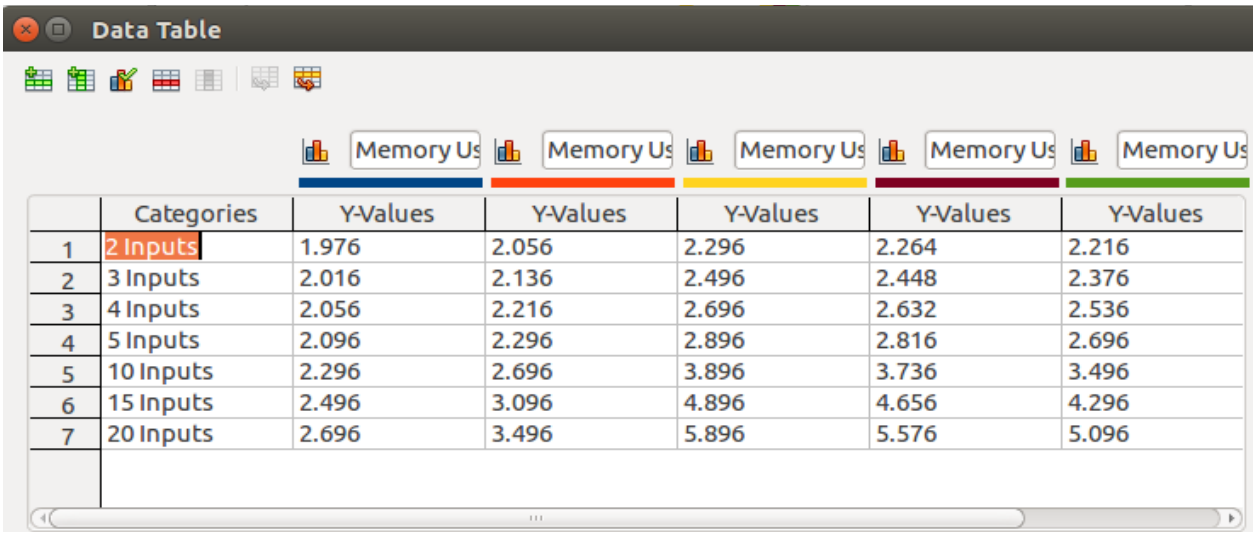
n	time(i)	total(B)	useful-heap(B)	extra-heap(B)	stacks(B)
16	132,503	1,472	1,341	131	0
17	133,394	2,056	1,909	147	0
18	134,129	2,096	1,927	169	0
19	134,590	2,136	1,945	191	0
20	135,072	2,176	1,975	201	0
21	135,557	2,216	1,994	222	0
22	136,333	2,216	1,994	222	0

Total Memory Consumption in 22 is 2,216 Bytes and this means that program used this much of Memory at most.

Below chart shows the **Memory Usage** of "psearch" program for different number of identical inputs



Data table of the above graph is this:



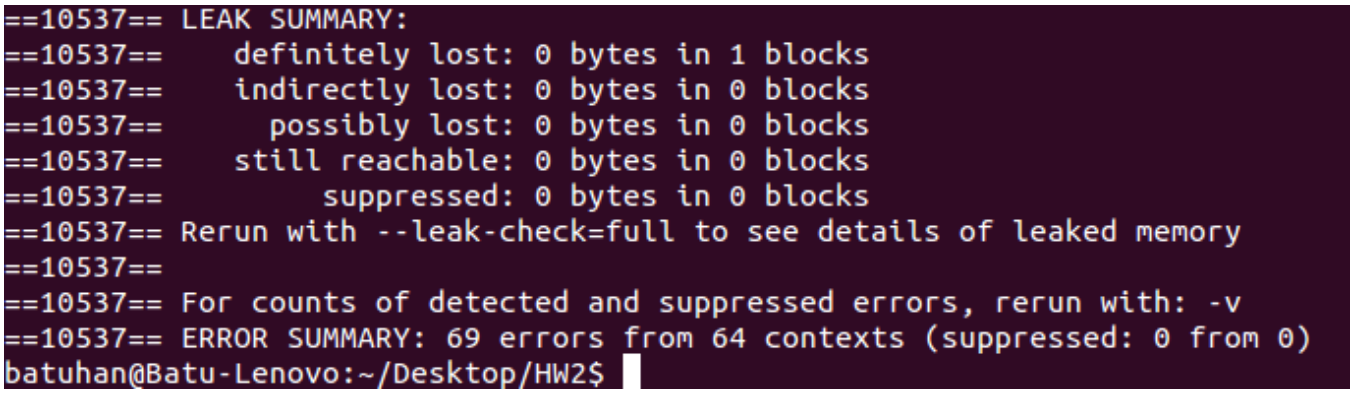
	Categories	Y-Values	Y-Values	Y-Values	Y-Values	Y-Values
1	2 Inputs	1.976	2.056	2.296	2.264	2.216
2	3 Inputs	2.016	2.136	2.496	2.448	2.376
3	4 Inputs	2.056	2.216	2.696	2.632	2.536
4	5 Inputs	2.096	2.296	2.896	2.816	2.696
5	10 Inputs	2.296	2.696	3.896	3.736	3.496
6	15 Inputs	2.496	3.096	4.896	4.656	4.296
7	20 Inputs	2.696	3.496	5.896	5.576	5.096

C) Memory Leak

I also checked my program with valgrind to see any kinds of memory leak if exists with this example command;

$ valgrind ./psearch hello 2 in1 in2 out

Result is below:



As seen in the picture, there is no memory leak at all