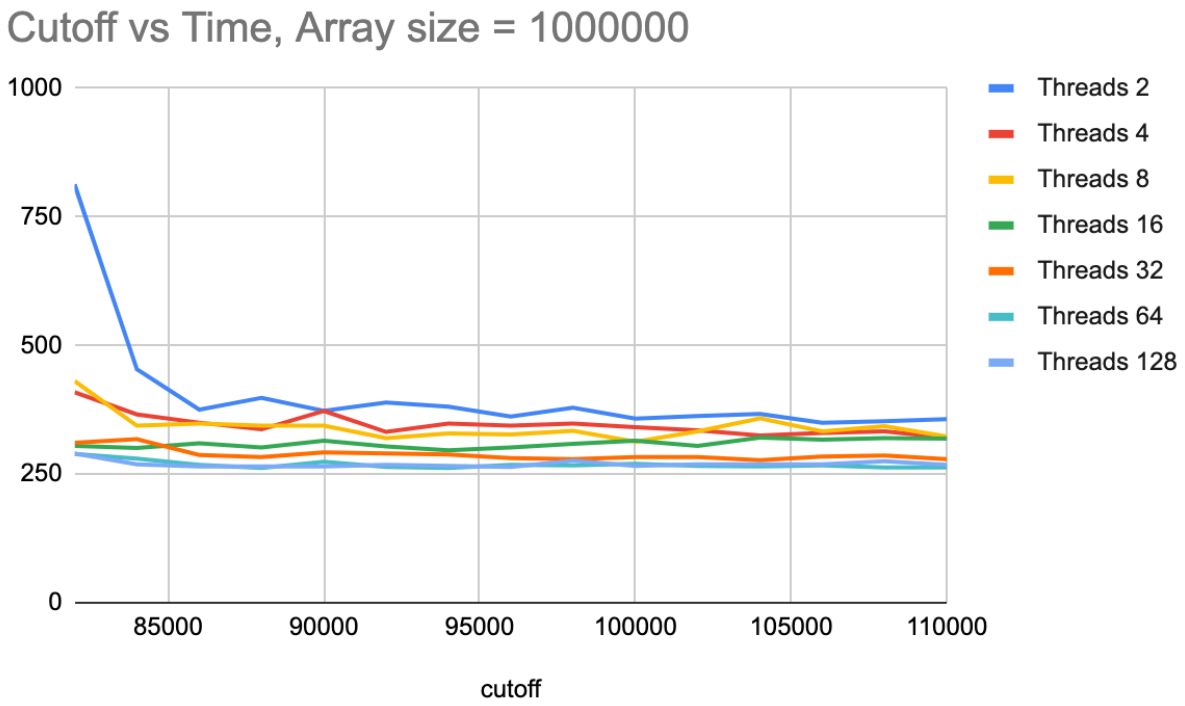
Harshal Jaiswal (1097734)

Program Structures & Algorithms
Fall 2021
Assignment 5
Parallel Sorting

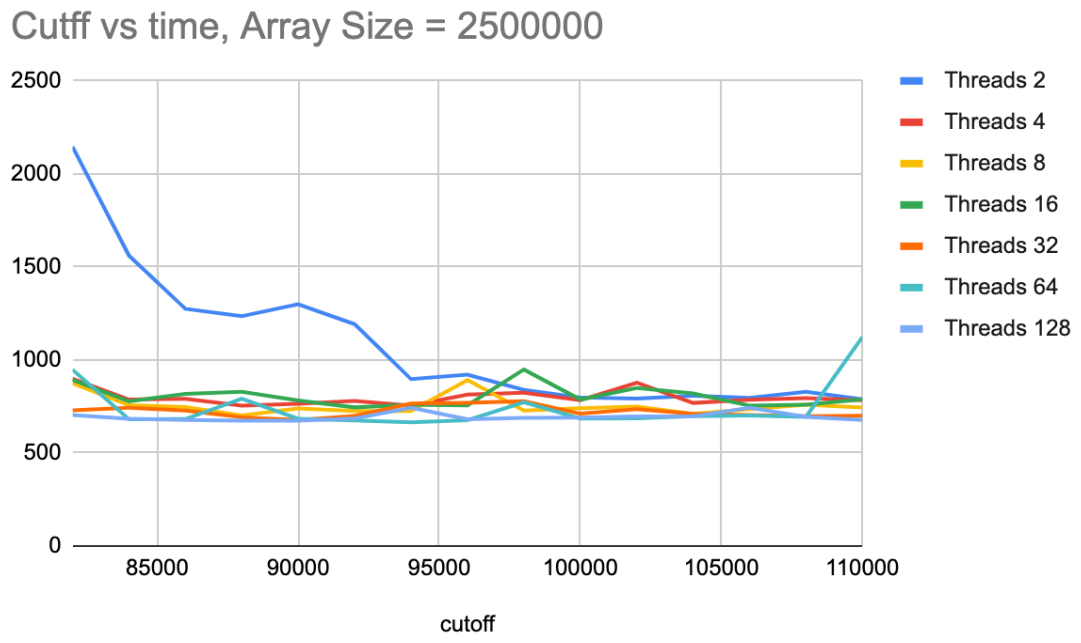
- **Task:**
    1. Experiment with different values of cutoff.
    2. Change the number of threads and experiment with it.
    3. Experiment with different size of Array.

- **Relationship Conclusion:**
    - From the graphs below it can be observed that increasing the number of threads after 4 does not have a huge impact on time. So it can be concluded that it is optimal to use the number of threads as 4.
    - Also, the value of the cut-off can be found almost not affecting the time after almost 20% of the array size.
    - So the optimal value for the combination of the thread and cut-off value should be 4 and 20% of array size.

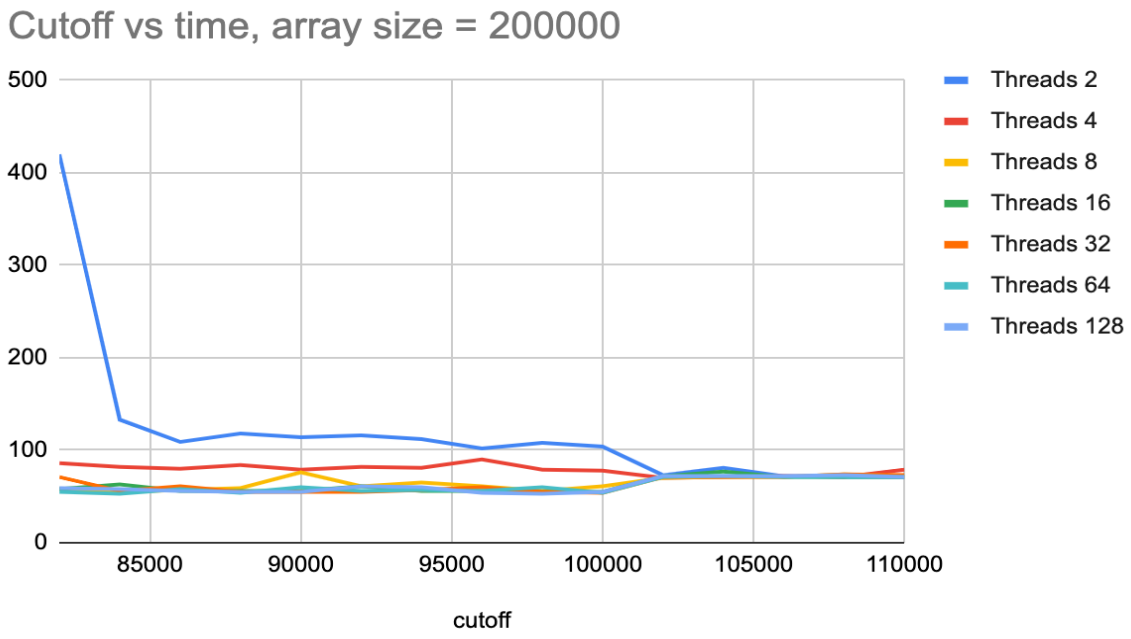
- **Output & Graphical Representation:**



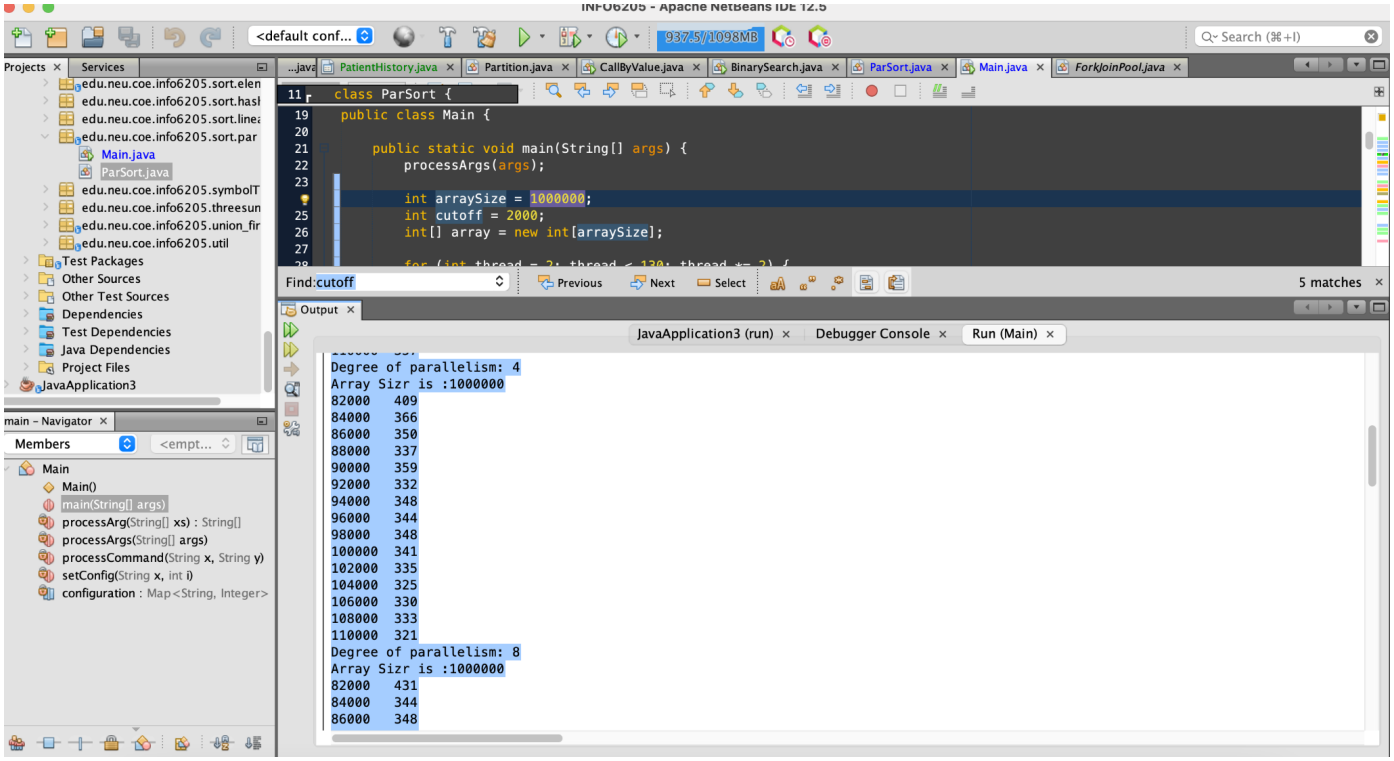
Array size = 1000000							
cutoff	Threads 2	Threads 4	Threads 8	Threads 16	Threads 32	Threads 64	Threads 128
82000	813	409	431	305	311	289	290
84000	454	366	344	301	318	280	269
86000	375	350	348	310	287	268	265
88000	398	337	344	302	283	262	265
90000	373	373	344	315	292	274	265
92000	389	332	320	304	290	264	268
94000	381	348	329	296	288	262	266
96000	362	344	327	302	281	268	264
98000	379	348	334	309	279	267	276
100000	358	341	313	315	283	270	266
102000	363	335	333	305	283	266	269
104000	367	325	358	321	277	265	269
106000	350	330	333	317	284	267	269
108000	353	333	343	320	286	263	275
110000	357	321	323	319	279	263	268



Array Size = 2500000							
cutoff	Threads 2	Threads 4	Threads 8	Threads 16	Threads 32	Threads 64	Threads 128
82000	2146	899	875	893	729	948	704
84000	1561	787	757	778	743	681	684
86000	1275	791	747	818	728	682	678
88000	1235	755	702	829	692	792	674
90000	1299	765	739	782	678	684	674
92000	1192	780	725	745	699	675	685
94000	897	755	726	760	766	665	743
96000	921	814	892	756	769	676	681
98000	839	824	728	949	778	773	689
100000	798	784	740	787	711	685	690
102000	792	877	750	849	736	686	696
104000	808	769	710	820	710	698	697
106000	796	786	735	754	704	702	743
108000	829	795	759	759	698	694	694
110000	788	783	744	788	700	1124	678



Array Size = 200000							
cutoff	Threads 2	Threads 4	Threads 8	Threads 16	Threads 32	Threads 64	Threads 128
82000	420	86	58	58	71	55	59
84000	133	82	55	63	56	53	58
86000	109	80	57	56	61	58	56
88000	118	84	59	56	55	54	55
90000	114	79	76	56	55	60	55
92000	116	82	61	61	55	56	61
94000	112	81	65	56	57	57	60
96000	102	90	61	56	60	56	54
98000	108	79	56	55	55	60	53
100000	104	78	61	55	54	54	55
102000	73	70	70	72	71	71	72
104000	81	71	71	77	71	72	72
106000	72	72	71	71	71	71	72
108000	71	71	74	71	74	71	73
110000	71	79	72	71	73	71	71


INFO6205 - Apache NetBeans IDE 12.5
<default conf...
937.5/1098MB
Search (⌘+I)
Projects
Services
edu.neu.coe.info6205.sort.elen
edu.neu.coe.info6205.sort.hasl
edu.neu.coe.info6205.sort.linea
edu.neu.coe.info6205.sort.par
Main.java
ParSort.java
edu.neu.coe.info6205.symbolT
edu.neu.coe.info6205.threesun
edu.neu.coe.info6205.union_fir
edu.neu.coe.info6205.util
Test Packages
Other Sources
Other Test Sources
Dependencies
Test Dependencies
Java Dependencies
Project Files
JavaApplication3
main – Navigator
Members
Main
Main()
main(String[] args)
processArg(String[] xs) : String[]
processArgs(String[] args)
processCommand(String x, String y)
setConfig(String x, int i)
configuration : Map<String, Integer>
PatientHistory.java
Partition.java
CallByValue.java
BinarySearch.java
ParSort.java
Main.java
ForkJoinPool.java
11 class ParSort {
19 public class Main {
21 public static void main(String[] args) {
22 processArgs(args);
25 int arraySize = 1000000;
26 int cutoff = 2000;
27 int[] array = new int[arraySize];
Find:cutoff
Previous
Next
Select
5 matches
Output
JavaApplication3 (run)
Debugger Console
Run (Main)
Degree of parallelism: 4
Array Sizr is :1000000
82000 409
84000 366
86000 350
88000 337
90000 359
92000 332
94000 348
96000 344
98000 348
100000 341
102000 335
104000 325
106000 330
108000 333
110000 321
Degree of parallelism: 8
Array Sizr is :1000000
82000 431
84000 344
86000 348