Vennela Reddy Karaddi

NU ID: 0010181643
Assignment 5

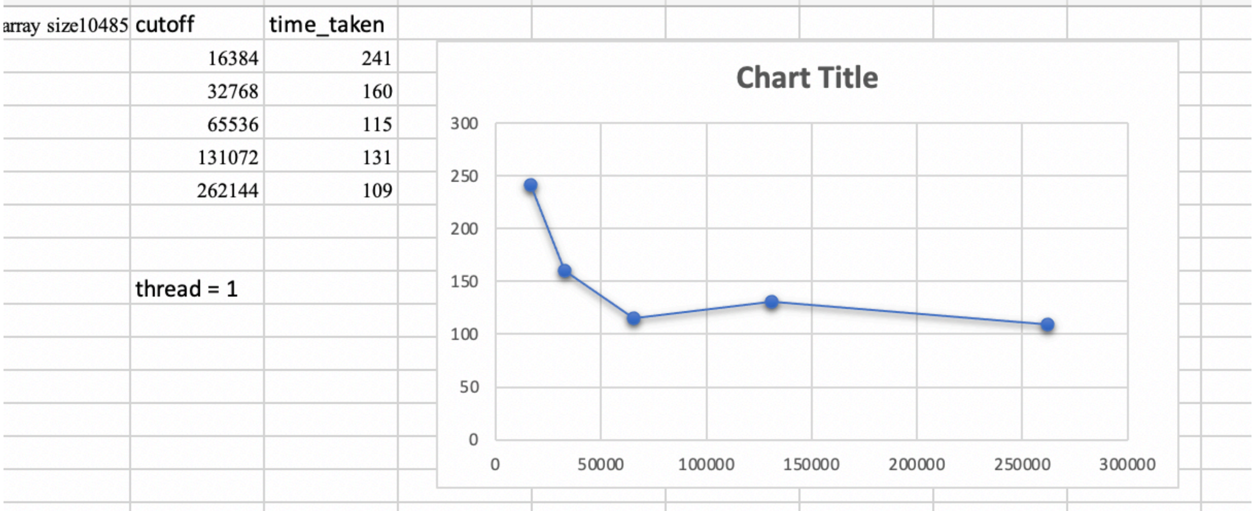
Parallel Sort algorithm

The program aims to sort an array leveraging the multithreading capability of the processor to improve performance. Merge Sort is a good candidate for this, as it consists of distinct segments that can be processed in parallel then merged together. As such, bottom up merge sort with multithreading using a fixed thread pool has been used.

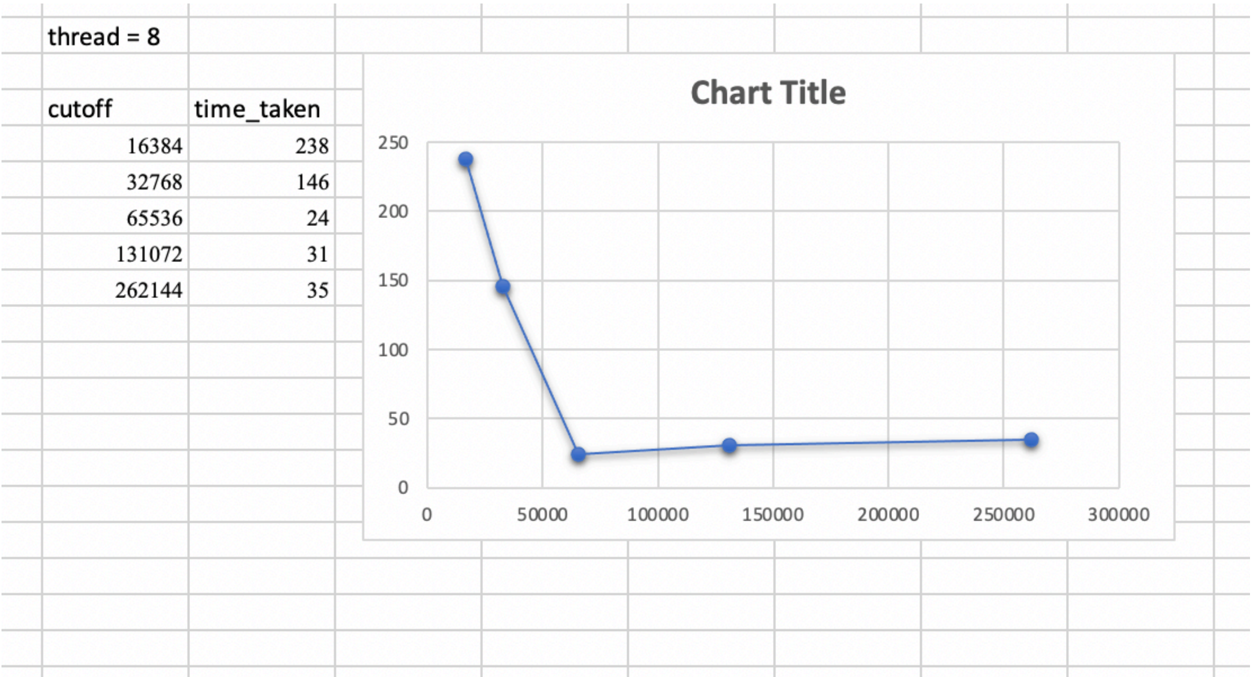
Measured the time taken to sort the array with respect to:

1. Size of the array
2. Cutoff size
3. Thread count in the pool

Graphs :



array size10485	cutoff	time_taken
	16384	241
	32768	160
	65536	115
	131072	131
	262144	109
	thread = 1	



thread = 8

cutoff	time_taken
16384	238
32768	146
65536	24
131072	31
262144	35

Conclusions –

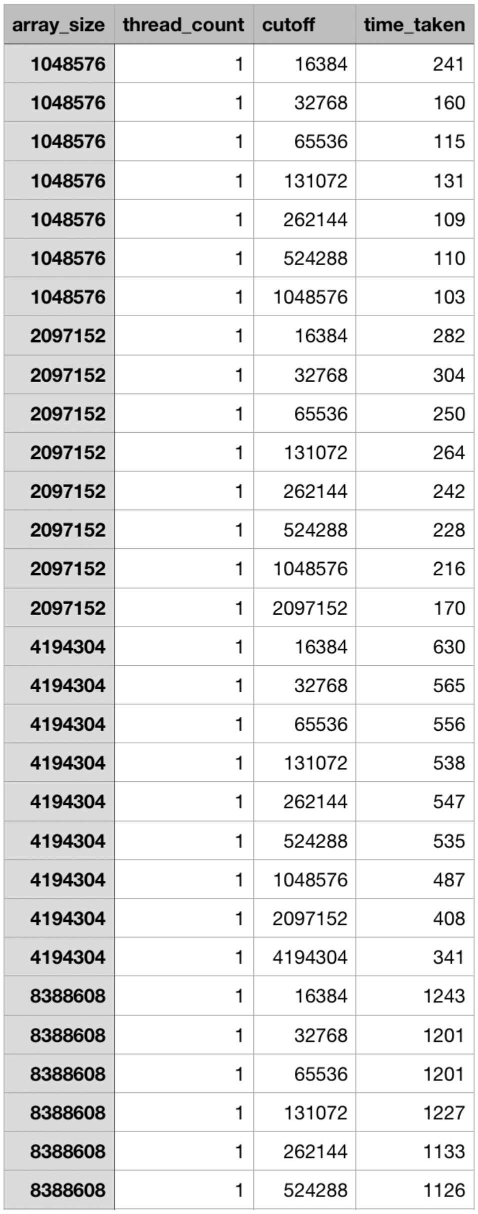
1. The time taken to sort on a single thread decreases as cutoff size increases.

2. When sorting on multiple threads, the smaller cutoff means more segments that can be processed in parallel improving the time.

3.In the experiments, when the cutoff and array size decreases, the sort time seems to be decreasing.

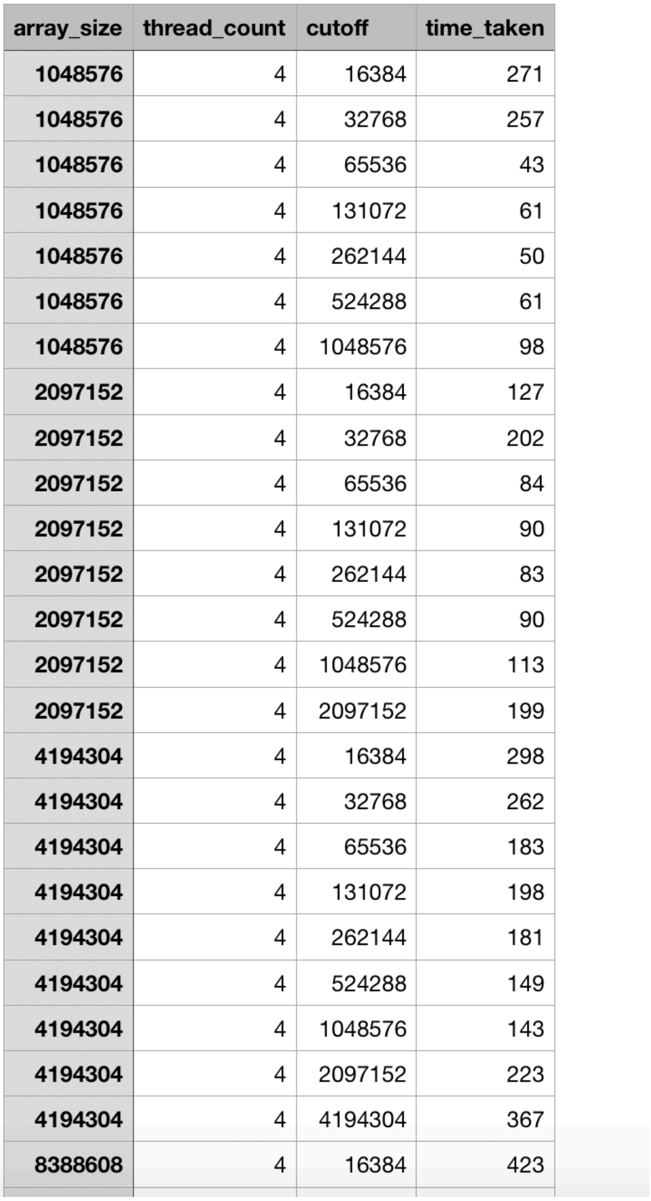
All the data respectively with thread_count is present in data1.csv, data2.csv, data3.csv, data4.cvs

data1



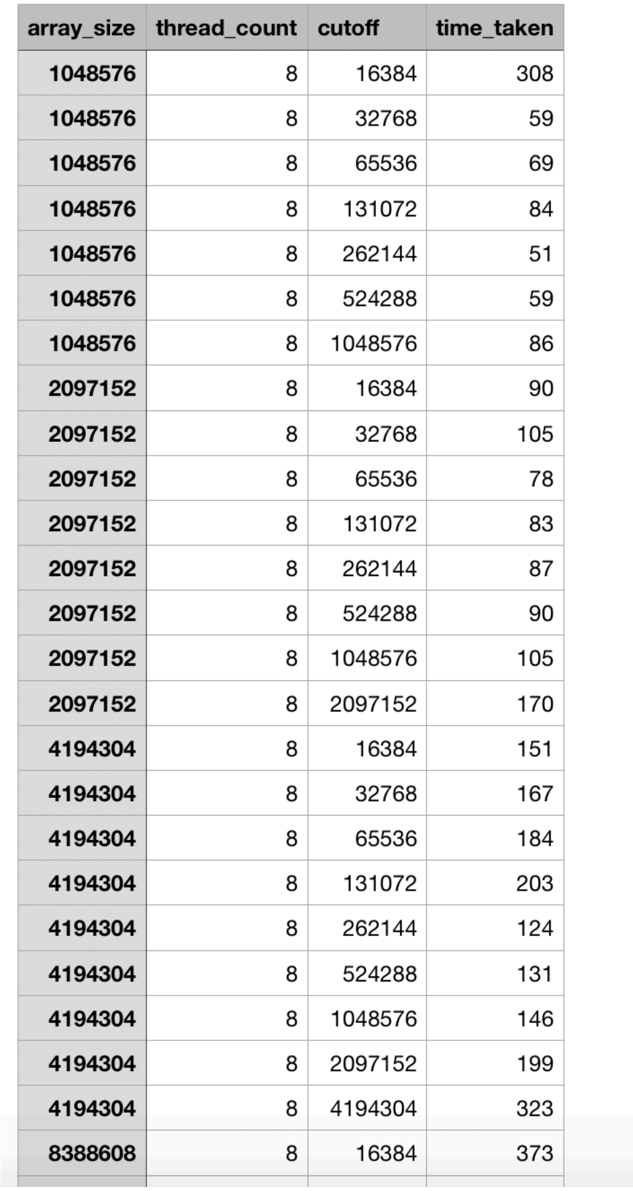
array_size	thread_count	cutoff	time_taken
1048576	1	16384	241
1048576	1	32768	160
1048576	1	65536	115
1048576	1	131072	131
1048576	1	262144	109
1048576	1	524288	110
1048576	1	1048576	103
2097152	1	16384	282
2097152	1	32768	304
2097152	1	65536	250
2097152	1	131072	264
2097152	1	262144	242
2097152	1	524288	228
2097152	1	1048576	216
2097152	1	2097152	170
4194304	1	16384	630
4194304	1	32768	565
4194304	1	65536	556
4194304	1	131072	538
4194304	1	262144	547
4194304	1	524288	535
4194304	1	1048576	487
4194304	1	2097152	408
4194304	1	4194304	341
8388608	1	16384	1243
8388608	1	32768	1201
8388608	1	65536	1201
8388608	1	131072	1227
8388608	1	262144	1133
8388608	1	524288	1126

data2



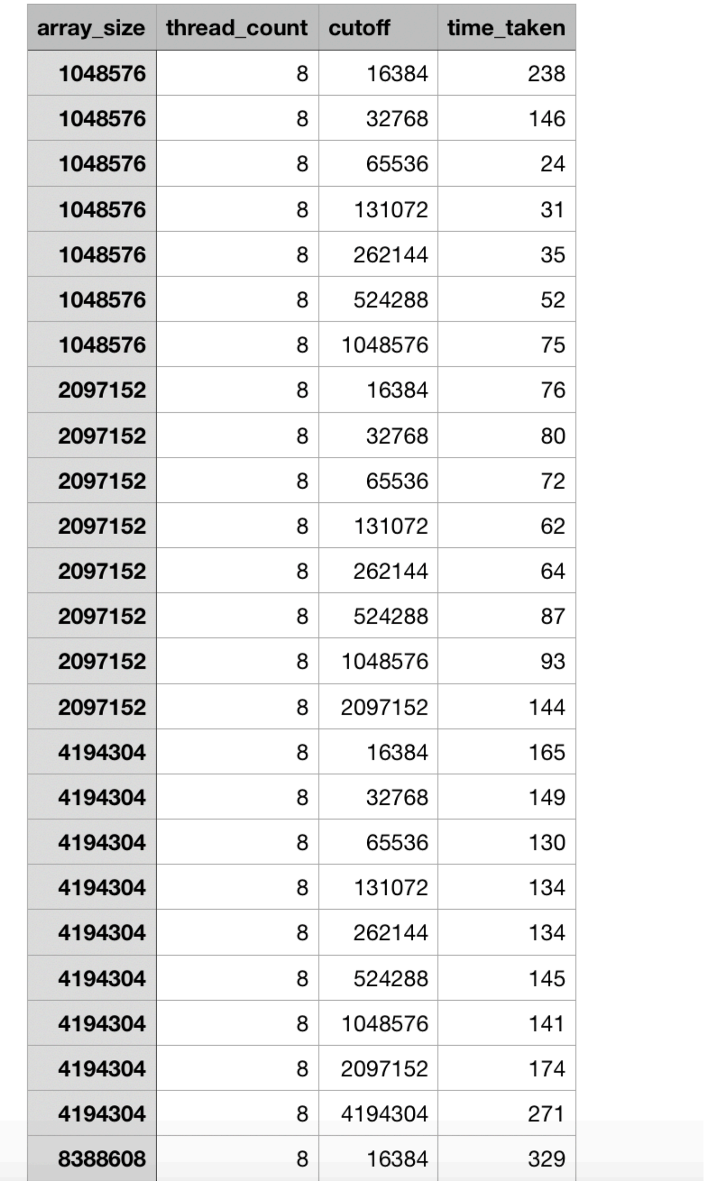
array_size	thread_count	cutoff	time_taken
1048576	4	16384	271
1048576	4	32768	257
1048576	4	65536	43
1048576	4	131072	61
1048576	4	262144	50
1048576	4	524288	61
1048576	4	1048576	98
2097152	4	16384	127
2097152	4	32768	202
2097152	4	65536	84
2097152	4	131072	90
2097152	4	262144	83
2097152	4	524288	90
2097152	4	1048576	113
2097152	4	2097152	199
4194304	4	16384	298
4194304	4	32768	262
4194304	4	65536	183
4194304	4	131072	198
4194304	4	262144	181
4194304	4	524288	149
4194304	4	1048576	143
4194304	4	2097152	223
4194304	4	4194304	367
8388608	4	16384	423

data3

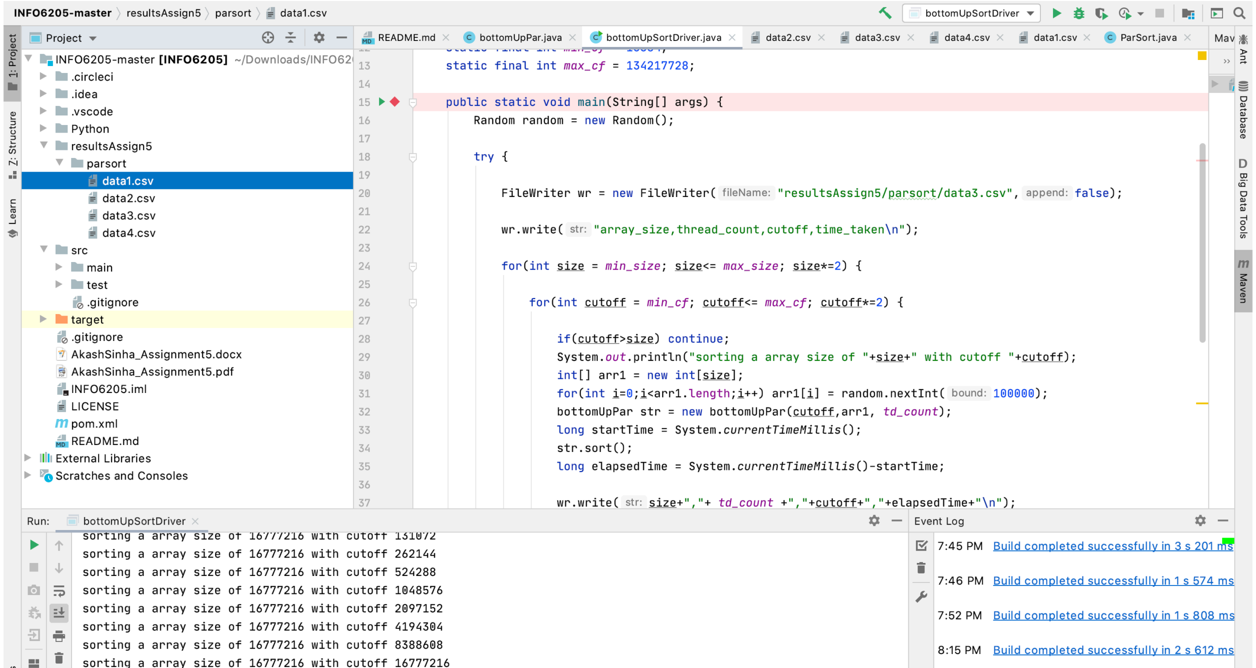


array_size	thread_count	cutoff	time_taken
1048576	8	16384	308
1048576	8	32768	59
1048576	8	65536	69
1048576	8	131072	84
1048576	8	262144	51
1048576	8	524288	59
1048576	8	1048576	86
2097152	8	16384	90
2097152	8	32768	105
2097152	8	65536	78
2097152	8	131072	83
2097152	8	262144	87
2097152	8	524288	90
2097152	8	1048576	105
2097152	8	2097152	170
4194304	8	16384	151
4194304	8	32768	167
4194304	8	65536	184
4194304	8	131072	203
4194304	8	262144	124
4194304	8	524288	131
4194304	8	1048576	146
4194304	8	2097152	199
4194304	8	4194304	323
8388608	8	16384	373

data4



array_size	thread_count	cutoff	time_taken
1048576	8	16384	238
1048576	8	32768	146
1048576	8	65536	24
1048576	8	131072	31
1048576	8	262144	35
1048576	8	524288	52
1048576	8	1048576	75
2097152	8	16384	76
2097152	8	32768	80
2097152	8	65536	72
2097152	8	131072	62
2097152	8	262144	64
2097152	8	524288	87
2097152	8	1048576	93
2097152	8	2097152	144
4194304	8	16384	165
4194304	8	32768	149
4194304	8	65536	130
4194304	8	131072	134
4194304	8	262144	134
4194304	8	524288	145
4194304	8	1048576	141
4194304	8	2097152	174
4194304	8	4194304	271
8388608	8	16384	329



```java
    static final int max_cf = 134217728;

    public static void main(String[] args) {
        Random random = new Random();

        try {

            FileWriter wr = new FileWriter( fileName: "resultsAssign5/parsort/data3.csv", append: false);

            wr.write( str: "array_size,thread_count,cutoff,time_taken\n");

            for(int size = min_size; size<= max_size; size*=2) {

                for(int cutoff = min_cf; cutoff<= max_cf; cutoff*=2) {

                    if(cutoff>size) continue;
                    System.out.println("sorting a array size of "+size+" with cutoff "+cutoff);
                    int[] arr1 = new int[size];
                    for(int i=0;i<arr1.length;i++) arr1[i] = random.nextInt( bound: 100000);
                    bottomUpPar str = new bottomUpPar(cutoff,arr1, td_count);
                    long startTime = System.currentTimeMillis();
                    str.sort();
                    long elapsedTime = System.currentTimeMillis()-startTime;

                    wr.write( str: size+","+ td_count +","+cutoff+","+elapsedTime+"\n");
```

```
sorting a array size of 16777216 with cutoff 131072
sorting a array size of 16777216 with cutoff 262144
sorting a array size of 16777216 with cutoff 524288
sorting a array size of 16777216 with cutoff 1048576
sorting a array size of 16777216 with cutoff 2097152
sorting a array size of 16777216 with cutoff 4194304
sorting a array size of 16777216 with cutoff 8388608
sorting a array size of 16777216 with cutoff 16777216
```

```
7:45 PM  Build completed successfully in 3 s 201 ms
7:46 PM  Build completed successfully in 1 s 574 ms
7:52 PM  Build completed successfully in 1 s 808 ms
8:15 PM  Build completed successfully in 2 s 612 ms
```