

COMP1921: PROGRAMMING PROGRAMMING
PROJECT: BENCHMARKING SORTING ALGORITHMS
REPORT

CONTENTS

1. DESIGN DOCUMENT

- 1. Purpose of the modules*
- 2. Functions used in each module*
- 3. Libraries used in modules*
- 4. References*

2. TEST PLAN

- 1. Include screen shots of results*
- 2. Include graphs screen shots of sorting*
- 3. Other functions and tools*

3. CONCLUSION

DESIGN DOCUMENT

1. Beginning with the minimal stage of creating the module files, I began with the module files because it will be easier for me to see what I will define later and will work on, this will also make it much easier to create my makefile. Below I have included the module file screen shots along with the fully created makefile:

In make file, I added the modules files and header file

```
# dependencies

main.o: main.c bubbleSort_header.h quickSort_header.h selectionSort_header.h
bubbleSortDynamic.o: bubbleSortDynamic.c bubbleSort_header.h
quickSortDynamic.o: quickSortDynamic.c quickSort_header.h
selectionSortDynamic.o: selectionSortDynamic.c selectionSort_header.h
mergeSortDynamic.o: mergeSortDynamic.c mergeSort_header.h
```

2. The next stage I have included is my first method of sorting, minimally done, this is the bubble sort method, not comprehensively complete however it works:

bubbleSortDynamic.c

```
1 //*****
2 //*****bubbleSortDynamic.c*****
3 //*****
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <time.h>
8 #include "bubbleSort_header.h"
9
10 //Create the function which test bubble sorting of the random numbers
11 double testBubbleSorting (int n)
12 {
13     //Declaring the variable, data is pointer to integer
14     int *data, *temp_array;
15
16     //Allocate space for 'numberOfElements' integers
17     data = calloc(n , sizeof(int));
18
19     //Using for loop
20     //to generate the random numbers
21     for (temp_array = data; temp_array <data + n; temp_array++)
22     {
23
24         //rand function returns a pseudo-random number.
25         *temp_array = rand() % n;
26     }
27
28     //printf("The %d random numbers before sorting :\\n", n);
29
30     //Call the function to print out the unsorted list
31     //printBubbleSorting(data,numberOfElements);
32
33     //Call the bubbleSorting() function to sort out the unsorted numbers
34     bubbleSort(data,n);
35
36     printf("The %d random numbers after sorting :\\n", n);
37
38     //Call the printBubbleSorting() to print out sorted list.
39     printBubbleSorting(data,n);
40
41     double time_bubbleSort = bubbleSort(data, n);
42     printf("\\nTime taken for bubble sort in seconds: %f\\n", time_bubbleSort );
43 }
```

bubbleSort_header.h

```
1  /*****
2  *****/bubbleSort_header.h*****/
3  *****/
4
5  double testBubbleSorting (int n );
6
7  void swapForBubbleSort ( int *p, int *q);
8
9  double bubbleSorting ( int *data, int numberOfElements );
10
11 void printBubbleSorting ( int *data, int numberOfElements );
12
```

3. This stage includes the quick sort stage, just like the bubble sort it works however remains fully complete, this is just for this stage as it requires for it to work, I have included this below:

quickSortDynamic.c

```
bubbleSortDynamic bubbleSort_header graph.out quickSort_header.h main.c mergeSortDynamic quickSortDynamic.c mergeSort_header selectionSort_header selectionSortDynamic graph.gnu makefile
1  /*****
2  *****/quickSort_header.h*****/
3  *****/
4
5  void swapForQuickSort(int *x,int *y);
6
7  int choose_pivot(int i,int j );
8
9  double quickSorting(int *list,int m,int n);
10
11 void printQuickSorting(int *list,const int n);
12
13 double testQuickSorting ( int n );
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614

```

4. This stage includes my selection sorting optional sort which is the selection sort, this stage is close to completed, it works, however not comprehensively done so:

selectionSortDynamic.c

```
bubbleSortDynamic.c | graph.out | quickSort_header.h | main.c | mergeSortDynamic.c | quickSortDynamic.c | mergeSort_header.h | selectionSort_header.h | selectionSortDynamic.c | graph.gnu | makefile
1  /*****
2  *****/
3  *****/
4
5  //Standard I/O Libraries
6  #include <stdlib.h>
7  #include <stdio.h>
8  #include <time.h>
9  #include "selectionSort_header.h"
10
11 //Create the function 'swapForBubbleSort' to swap the two different elements
12 void swapForSelectionSort(int *first_element, int *second_element)
13 {
14     //Create temporary array
15     //Initialise the variable and assign to pointer p
16     int temp = *first_element;
17     *first_element = *second_element
18     *second_element = temp;
19 }
20
21 double selectionSort(int *data, int n)
22 {
23     //Decalare time
24     //And initialise the start_t and end_t
25     clock_t start_t, end_t;
26     start_t = clock();
27
28     //Decalare the vairiables
29     int i, j, min_idx;
30
31     // Using for loop to move one by one every boundary of unsorted subarray
32     for (i = 0; i < n-1; i++)
33     {
34         //Find the minimum element in unsorted array
35         min_idx = i;
36         for (j = i+1; j < n; j++)
37         {
38             if (data[j] < data[min_idx])
39             {
40                 min_idx = j;
41             }
42         }
43         //Swap the found minimum element with the first element
44     }
45 }
selectionSortDynamic.c 16/28 UTF-8 C
```

selectionSort_header.h

```
bubbleSortDynamic.c | graph.out | quickSort_header.h | main.c | mergeSortDynamic.c | quickSortDynamic.c | mergeSort_header.h | selectionSort_header.h | selectionSortDynamic.c | graph.gnu | makefile
1  /*****
2  *****/
3  *****/
4
5  void swapForSelectionSort(int *xp, int *yp);
6
7  double selectionSort(int *data, int n);
8
9  void printSelectionSorting(int *data, int numberOfElements);
10
11 double testSelectionSorting(int n);
12
```

5. This stage includes my selection sorting optional sort which is the selection sort, this stage is close to completed, it works, however not comprehensively done so:

mergeSortDynamic.c

```
bubbleSortDynamic.c  graph.out  quickSort_header.h  main.c  mergeSortDynamic.c  quickSortDynamic.c  mergeSort_header.h  selectionSort_header  selectionSortDynamic  graph.gnu  makefile
1  /*****mergeSortDynamic.c*****/
2  /*****mergeSortDynamic.c*****/
3  /*****mergeSortDynamic.c*****/
4
5  //Standard I/O Libraries
6  #include <stdlib.h>
7  #include <stdio.h>
8  #include <time.h>
9  #include "mergeSort_header.h"
10
11 //Create a function to merge the two sub arrays or data
12 //Merges two subarrays of arr[]
13 void merge(int *data, int left, int middle, int right)
14 {
15     //Declaring the variables
16     /*First subarray is First_Array
17     Second subarray is Second_Array*/
18     int i, j, k;
19     int n1 = middle - left + 1;
20     int n2 = right - middle;
21
22     //create temporary Array
23     int First_Array[n1], Second_Array[n2];
24
25     //Copy data to temporary arrays
26     for (i = 0; i < n1; i++)
27     {
28         First_Array[i] = data[left + i];
29     }
30     for (j = 0; j < n2; j++)
31     {
32         Second_Array[j] = data[middle + 1 + j];
33     }
34
35     //Merge the temporary arrays back to the previous
36
37     //Initial index of first data (i.e sub arrays)
38     i = 0;
39     //Initial index of second data
40     j = 0;
41     // Initial index of merged data
42     k = left;
```

mergeSort_header.h

```
bubbleSortDynamic.c  graph.out  quickSort_header.h  main.c  mergeSortDynamic.c  quickSortDynamic.c  mergeSort_header.h  selectionSort_header  selectionSortDynamic  graph.gnu  makefile
1  /*****mergeSort_header.h*****/
2  /*****mergeSort_header.h*****/
3  /*****mergeSort_header.h*****/
4
5  void merge(int *data, int left, int middle, int right);
6
7  double mergeSort(int *data, int left, int right);
8
9  void printMergeSorting(int *list, int n);
10
11 double testMergeSorting ( int n );
12
```

6. This is the intermediate stage of my design, I have done the bubble sort, quick sort and selection sort to full detail, they all work and I have also began to add the commenting into my program, this also ensures that I have began to do things to near completion, below are the screen shots of all the 3 algorithms I have used for sorting. They all work and run, as compared to the previous stage, these are much more comprehensive.

main.c



```
10 *****Main.c*****
11 *****/
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include "quickSort_header.h"
16 #include "selectionSort_header.h"
17 #include "bubbleSort_header.h"
18 #include "mergeSort_header.h"
19
20
21 int main ( void )
22 {
23     FILE *fp = fopen("graph.out", "w+");
24
25     //Decalaring the variables
26     double time_mergeSort;
27     double time_selectionSort;
28     double time_bubbleSort;
29     double time_quickSort;
30
31     //Decalaring the variables
32     int counter = 1;
33     int n;
34     int choice;
35     srand(time(NULL));
36
37     printf("\nEnter the random numbrs: ");
38     scanf("%d", &n);
39
40     //Using while loop to keep selecting the number of elements
41     while ( n != 0)
42     {
43
44         //Using while loop to choose the option for sorting
45         while(1)
46         {
47             printf("\nChoose the sorting type:\n\n");
48             printf("1. Quick sort\n");
49             printf("2. Merge sort\n");
50             printf("3. Selection sort\n");
51             printf("4. Bubble sort\n");
```

7. I have now begun to near the final stages of my program, this shows that I have finished commenting and now also ran tests on my program to see whether it works or not, and now that they all work and commenting is finished/near completion, I will now begin to debug my program and do tests to see if it can break to be comprehensive with it.

graph.gnu

```
bubbleSortDynam  graph.out  bubbleSort_head  graph.gnu  header.h  main.c  makefile  mergeSortDynam  mergeSort_headr  quickSortDynam  quickSort_header  selectionSortDyn  selectionSort_he

1 #####
2 #
3 #          GNU FILE
4 #
5 #####
6
7
8
9 set title 'Graph for Sorting'  #set title
10 set xlabel 'Length'          #set x axis
11 set ylabel 'Time in seconds'  #set y axis
12 plot 'graph.out' u 1:2 w lp t 'Quick Sort', 'graph.out' u 1:3 w lp t 'Merge Sort', 'graph.out' u 1:4 w lp t 'Selection Sort', 'graph.out' u 1:5 w lp t 'Bubble Sort'
13
```

Below I have included additional screen shots for my final finished design of the program, this includes the modules and how the program is layed out, and my finalised touches that were added to it.

In main.c, I have provided the choice to select the sorting type

```
bubbleSortDynamic  graph.out  quickSort_header.h  main.c  mergeSortDynamic.c  quickSortDynamic.c  mergeSort_header.h  selectionSort_header  selectionSortDynamic  graph.gnu  makefile

44 //Using while loop to choose the option for sorting
45 while(1)
46 {
47     printf("\nChoose the sorting type:\n\n");
48     printf("1. Quick sort\n");
49     printf("2. Merge sort\n");
50     printf("3. Selection sort\n");
51     printf("4. Bubble sort\n");
52     printf("5. Exit\n");
53     printf("6. For another Number of Elements\n");
54
55     scanf("%d", &choice);
56     printf("Option: %d\n", choice);
57     if ( choice > 0 && choice <= 6)
58     {
59         switch (choice)
60         {
61             case 1:
62                 time_quickSort = testQuickSorting(n);
63                 break;
64             case 2:
65                 time_mergeSort = testMergeSorting(n);
66                 break;
67             case 3:
68                 time_quickSort = testSelectionSorting(n);
69                 break;
70             case 4:
71                 time_bubbleSort = testBubbleSorting(n);
72                 break;
73             case 5:
74                 printf("Exit!!!!!!!!!!!!\n");
75                 //system("PAUSE");
76                 exit(0);
77         }
78         fprintf(fp, "%d %f %f %f\n", n, time_quickSort, time_mergeSort, time_selectionSort, time_bubbleSort );
79     }
80     else if (choice == 6)
81     {
82         printf("\nGive the number of elements : ");
83         scanf("%d", &n);
84         counter++;
85     }
86 }
```

makefile

| | | | | | | | | | | |
|---------------------|-----------|--------------------|--------|--------------------|--------------------|--------------------|----------------------|----------------------|-----------|----------|
| bubbleSortDynamic.c | graph.out | quickSort_header.h | main.c | mergeSortDynamic.c | quickSortDynamic.c | mergeSort_header.h | selectionSort_header | selectionSortDynamic | graph.gnu | makefile |
|---------------------|-----------|--------------------|--------|--------------------|--------------------|--------------------|----------------------|----------------------|-----------|----------|

```
1
2 # code details
3
4 EXE_DIR = .
5 EXE = $(EXE_DIR)/sortingAlgorithms
6
7 SRC= main.c bubbleSortDynamic.c quickSortDynamic.c selectionSortDynamic.c mergeSortDynamic.c
8
9 # generic build details
10
11 CC= cc
12 COPT= -O
13 CFLAGS=
14
15 # compile to object code
16
17 OBJ= $(SRC:.c=.o)
18
19 .c.o:
20 | $(CC) $(COPT) -c -o $@ $<
21
22 # build executable
23
24 $(EXE): $(OBJ)
25 | $(CC) $(OBJ) $(CFLAGS) -lm -o $(EXE)
26
27 # clean up compilation
28
29 clean:
30 | rm -f $(OBJ) $(EXE)
31
32 # dependencies
33
34 main.o: main.c bubbleSort_header.h quickSort_header.h selectionSort_header.h
35 bubbleSortDynamic.o: bubbleSortDynamic.c bubbleSort_header.h
36 quickSortDynamic.o: quickSortDynamic.c quickSort_header.h
37 selectionSortDynamic.o: selectionSortDynamic.c selectionSort_header.h
38 mergeSortDynamic.o: mergeSortDynamic.c mergeSort_header.h
39
```

makefile 38:30 UTF-8 Makefile

PURPOSE OF THE MODULES

- **bubbleSortDynamic.c:** This is my bubble sort algorithm, in this I am using a dynamic array to sort out the random unsorted numbers.
 - **bubbleSort_header.h:** Declaration of the variables that used bubbleSortDynamic.c.
 - **quickSortDynamic.c:** This is my quick sort algorithm, in this I am also using dynamic array to sort out the random unsorted numbers.
 - **quickSort_header.h:** This is the declaration of the variables that I used in quickSortDynamic.c
 - **selectionSortDynamic.c:** This is my selection sorting algorithm, used for the same purpose as to sort unsorted numbers in an array.
 - **selectionSort_header.h:** This is the declaration of the variables being used in the selectionSortDynamic.c
 - **main.c:** This is the main file, in this I am calling all the sorting functions, allowing the user choice of control to choose an sorting algorithm to sort a list of numbers.
 - **graph.gnu:** Allows the plotting of the program data to the gnu plot graph, seen and represented in 2D plots.
-

FUNCTIONS USED IN EACH MODULE

- **bubbleSortDynamic.c:**

- swapForBubbleSort: I have created a function void swapForBubbleSort() with two arguments, allows to swap the two different elements or arrays.
- bubbleSorting: The function double bubbleSorting() with two arguments, allows to implement bubble sorting algorithm.
- printBubbleSorting: The function void printBubbleSorting() with two arguments, allows to print out the unsorted random numbers and after applying the bubble sorting function, it will sort out the randomly sorted numbers. However it is not important to use this to print out the whole list in the terminal.
- testBubbleSorting: The function double testBubbleSorting() with 1 argument, which is creating a unsorted random number, and then calling the bubble sorting function inside it, this is called to the main.c.

- **selectionSortDynamic.c:**

- swapForSelectionSort: The function is void swapForSelectionSort() has two arguments which allow to swap between two different elements.
- selectionSort: The function is void selectionSort() has two arguments which implements the selection sort, inside this function, swapForSelection was called into this.
- printSelectionSorting: The function void printSelectionSorting() has 2 arguments, which allow user to print out the unsorted list and then allow it to reprint to sorted list.
- testSelectionSorting: The function void testSelectionSorting() has 1 argument, this calls the selectionSort and this function is called inside the main.

- **quickSortDynamic.c:**

- swapForQuickSort : The function is void swapForQuickSort() has 2 arguments which allows to swap between two different elements.
- choosePivot : The function void choosePivot() has 2 arguments which takes last element as pivot, places the pivot element at its correct position in sorted array.

- quickSorting : The function double quickSorting() has 3 arguments which Implements the main method for quick sort. Inside this function, choosePivot() and swapForQuickSort() are used to implements the quick sorting algorithm.
 - printQuickSorting : The function void printQuickSorting() has 2 arguments, which allows user to print out the unsorted list and then allow it to reprint to reprint to sorted list.
 - testQuickSorting: The function void testQuickSorting() has 1 argument, this calls the quickSorting() and printQuickSorting(). This function is called inside the main file.
 - **MergeSortDynamic.c** :
 - merge : The function void merge() has 3 arguments which allow to merge two sub arrays of a array. This function implements the method for merge sorting
 - mergeSort : The function double mergeSort() has 3 arguments which sort the first array and then second half and then it calls the merge() function to implement the sorting.
 - printMergeSorting : The function void printMergeSorting() has 2 arguments, which allows user to print out the unsorted list and then allow it to reprint to reprint to sorted list.
 - testMergeSorting: The function void testMergeSorting() has 1 argument, this calls the quickSorting() and printQuickSorting(). This function is called inside the main file.
 - **main.c**:
 - Main calls all the functions inside it and also allows for the user input to call the specific sorting algorithms and also allows the user to then run and test the sorting algorithms through the terminal.
 - **graph.gnu** :
 - graph.gnu allow to print out the result as graph on gnuplot, where in graph x-axis is Length and y-axis is Time in seconds.
-

The libraries I have used within this program include:

- **#include <stdio.h>** : The stdio.h header defines three variable types, several macros, and various functions for performing input and output.
 - **#include <stdlib.h>** : stdlib.h is the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others.
 - **#include <time.h>** : The time.h header defines four variable types, two macro and various functions for manipulating date and time.
 - **#include "bubbleSort_header.h"** : In this header file I declared the variables which are used in bubbleSortDynamic.c
 - **#include "quickSort_header.h"** : In this header file I declared the variables which are used in bubbleSortDynamic.c
 - **#include "mergeSort_header.h"** : In this header file I declared the variables which are used in mergeSortDynamic.c
 - **#include "selectionSort_header.h"** : In this header file I declared the variables which are used in selectionSortDynamic.c
-

External code that I have used from public online repositories includes:

References:

- <http://www.geeksforgeeks.org/>
 - <http://www.mzan.com/article/26128092-i-am-getting-a-segmentation-fault-core-dumped-error-with-dynamically-allocated.shtml>
 - <https://www.youtube.com/watch?v=Jdtq5uKz-w4>
 - http://gnuplot.sourceforge.net/docs_4.2/node184.html
 - <http://alvinalexander.com/technology/gnuplot-charts-graphs-examples>
-

TEST PLAN

My initial thoughts on how the tests would run was, minor faults at the start however it should run fine and perfectly, I have planned to test this from the beginning as if it would work step by step, so I would be able to sort out the errors one by one, this would leave room for much less errors and help me find the errors much more efficiently and debug my code this way.

Running through the test phase one of my program this is what happened with Bubble Sort:

Bubble Sort 100 Words:

After execution the bubble sort algorithm displays that the time taken to process 100 random numbers was 0.000 seconds; this is also dependant on the computer and their process.

```
The 100 random numbers after sorting :
0      0      1      2      2      3      3      3      4      4      5      5      7      10     10     11     12     13     14     15     15     17     17     17     18
1      23     24     27     27     28     28     29     30     30     31     32     32     33     34     35     37     39     40     40     40     41     41     42     43
2      48     50     51     51     54     55     56     57     58     59     59     60     61     62     63     64     65     65     66     67     70     71     71     72     74
4      75     75     76     77     78     83     83     84     85     85     87     89     91     97     97     98     98     98

100 random numbers sorted by using bubble sorting algorithms.

Time taken for bubble sort in seconds: 0.000000
```

Bubble Sort 10,000 Words:

After execution the bubble sort algorithm displays that the time taken to process 10,000 random numbers was 0.02000 seconds.

```
5      9366   9367   9368   9369   9369   9371   9372   9373   9374   9375   9377   9378   9379   9381   9381   9381   9382   9383   9384   9384   9390   9390   9391   9392   9393
3      9393   9396   9399   9401   9402   9404   9404   9405   9405   9406   9406   9407   9411   9412   9412   9414   9415   9416   9417   9417   9417   9418   9418   9419   9420   942
1      9421   9422   9422   9423   9424   9424   9426   9427   9427   9428   9428   9428   9430   9430   9431   9431   9432   9432   9433   9433   9433   9434   9436   9437   9438   944
0      9443   9443   9444   9445   9446   9447   9449   9449   9451   9452   9454   9458   9458   9458   9459   9464   9464   9465   9466   9467   9468   9468   9469   9471   9471   947
2      9473   9474   9474   9474   9474   9475   9476   9476   9476   9477   9478   9479   9479   9479   9480   9481   9490   9491   9491   9491   9492   9494   9494   9494   9495   949
6      9497   9497   9497   9498   9498   9499   9501   9501   9503   9504   9504   9504   9507   9508   9509   9510   9511   9511   9511   9513   9515   9515   9516   9517   9517   952
0      9520   9520   9521   9522   9523   9524   9524   9525   9525   9527   9528   9529   9531   9532   9532   9533   9535   9536   9539   9540   9542   9543   9544   9544   954
7      9547   9547   9548   9555   9555   9559   9559   9561   9561   9562   9562   9563   9566   9566   9566   9567   9567   9567   9568   9569   9571   9571   9572   9573   9574   957
5      9575   9575   9576   9578   9579   9583   9583   9584   9584   9584   9587   9590   9592   9593   9593   9594   9595   9595   9595   9596   9598   9599   9600   9600   9600   960
1      9602   9606   9607   9607   9607   9608   9609   9610   9611   9611   9614   9615   9615   9617   9617   9618   9618   9622   9624   9624   9625   9625   9626   9626   9626   962
6      9628   9628   9632   9633   9633   9634   9634   9635   9635   9636   9637   9637   9638   9641   9642   9643   9645   9645   9646   9646   9646   9648   9648   9648   9648   964
8      9648   9649   9649   9649   9650   9650   9651   9652   9655   9655   9657   9658   9660   9660   9661   9661   9662   9663   9663   9664   9665   9665   9666   9667   9667   966
8      9670   9673   9673   9674   9675   9676   9676   9677   9679   9680   9681   9681   9683   9683   9684   9684   9684   9685   9688   9688   9689   9690   9691   9695   9695
6      9697   9699   9699   9700   9701   9702   9702   9702   9705   9705   9705   9707   9708   9710   9710   9711   9711   9713   9716   9716   9719   9720   9722   9722   9725   972
6      9727   9728   9728   9729   9729   9730   9730   9732   9734   9734   9736   9737   9737   9737   9738   9739   9740   9740   9741   9741   9742   9742   9743   9745   9746   974
8      9749   9750   9752   9752   9755   9757   9757   9757   9759   9760   9761   9765   9765   9766   9766   9766   9768   9769   9769   9772   9774   9777   9778   9778   9778   977
9      9781   9782   9783   9783   9787   9788   9789   9789   9789   9789   9790   9792   9794   9796   9796   9797   9797   9800   9801   9803   9804   9805   9806   9806   9807   980
9      9811   9812   9812   9812   9814   9814   9815   9819   9819   9820   9821   9823   9824   9826   9826   9827   9827   9827   9827   9830   9830   9833   9834   9839   9839   984
3      9843   9844   9844   9846   9846   9846   9846   9847   9847   9848   9853   9854   9854   9856   9860   9861   9861   9865   9869   9870   9870   9873   9879   9879   9880   988
5      9887   9888   9889   9889   9889   9889   9890   9890   9891   9891   9893   9893   9894   9894   9895   9896   9896   9897   9898   9899   9899   9900   9900   9901   9901   990
3      9903   9904   9904   9905   9905   9905   9906   9907   9908   9910   9911   9911   9911   9911   9912   9913   9913   9914   9914   9914   9914   9915   9915   9918   991
8      9921   9921   9921   9922   9925   9925   9927   9927   9927   9927   9928   9929   9933   9934   9934   9934   9935   9935   9935   9935   9938   9938   9938   9938   9939   993
9      9940   9940   9942   9942   9945   9943   9944   9944   9945   9946   9948   9951   9951   9951   9952   9952   9953   9955   9955   9957   9958   9959   9961   9961   9961   996
3      9964   9964   9966   9967   9969   9970   9971   9973   9974   9974   9975   9976   9977   9979   9979   9979   9982   9982   9983   9985   9987   9994   9994   9995   9996   9996
7      9998   9999   9999   9999

10000 random numbers sorted by using bubble sorting algorithms.

Time taken for bubble sort in seconds: 0.020000
```

Bubble Sort 100,000 Words:

After execution the bubble sort algorithm displays that the time taken to process 10,000 random numbers was 2.8800 seconds.

```
99263 99263 99265 99266 99267 99267 99267 99268 99269 99269 99270 99271 99273 99278 99281 99281 99282 99282 99282 99287 99287 99288 99288 99289 992
99290 99291 99294 99295 99295 99296 99297 99299 99299 99300 99301 99301 99302 99302 99303 99305 99306 99306 99307 99308 99308 99309 99310 993
99312 99314 99315 99317 99318 99318 99321 99321 99322 99324 99324 99330 99332 99332 99333 99335 99337 99338 99338 99339 99339 99341 99343 993
43 99353 99354 99354 99354 99356 99358 99358 99358 99359 99362 99362 99364 99364 99364 99365 99367 99368 99368 99369 99370 99371 99372 99372 993
73 99373 99374 99374 99375 99375 99377 99378 99380 99381 99381 99382 99382 99383 99383 99385 99385 99386 99387 99388 99389 99389 99391 99391 993
91 99393 99393 99393 99394 99396 99396 99397 99398 99398 99401 99402 99403 99403 99404 99406 99407 99407 99408 99409 99410 99411 99413 99413 99415 994
20 99421 99422 99422 99425 99426 99426 99426 99431 99431 99431 99432 99433 99434 99436 99436 99438 99438 99439 99440 99441 99441 99442 99443 99446 994
47 99447 99449 99450 99452 99455 99456 99457 99458 99459 99461 99463 99465 99466 99466 99467 99468 99468 99469 99470 99471 99472 99476 99478 99479 994
79 99479 99480 99480 99482 99483 99483 99483 99484 99485 99485 99486 99486 99487 99488 99489 99491 99491 99492 99493 99495 99496 99497 99497 995
00 99501 99502 99502 99502 99503 99508 99509 99510 99510 99510 99511 99512 99514 99514 99514 99515 99516 99516 99518 99518 99519 99520 995
20 99522 99522 99522 99523 99526 99527 99528 99529 99529 99531 99534 99534 99535 99536 99537 99538 99539 99540 99541 99545 99545 99546 99547 995
48 99549 99549 99550 99550 99552 99553 99553 99554 99555 99557 99558 99558 99562 99562 99563 99566 99566 99568 99569 99570 99571 99571 99574 995
74 99576 99576 99577 99578 99580 99582 99586 99593 99595 99596 99596 99597 99600 99600 99601 99603 99605 99606 99607 99608 99608 99611 996
14 99615 99615 99619 99619 99620 99621 99621 99621 99621 99623 99624 99624 99627 99627 99628 99630 99630 99631 99631 99632 99633 996
33 99633 99635 99638 99639 99641 99642 99642 99643 99646 99646 99647 99647 99649 99650 99651 99652 99655 99657 99658 99659 99660 99662 996
63 99663 99664 99666 99668 99671 99671 99672 99673 99674 99676 99677 99679 99680 99681 99684 99689 99689 99690 99691 99692 99693 99695 99698 996
98 99699 99702 99702 99703 99703 99705 99705 99706 99707 99712 99712 99713 99716 99719 99719 99719 99720 99720 99721 99722 99723 99725 99726 997
26 99726 99727 99727 99728 99729 99730 99730 99731 99732 99733 99737 99738 99739 99741 99743 99745 99745 99747 99748 99748 99751 99754 99754 997
55 99756 99757 99758 99759 99762 99766 99769 99770 99774 99778 99779 99782 99784 99785 99785 99787 99788 99790 99792 99794 99794 99795 997
97 99797 99797 99798 99798 99799 99801 99801 99801 99803 99806 99808 99808 99808 99814 99814 99815 99816 99817 99817 99818 99819 99821 99822 998
22 99823 99823 99824 99824 99826 99827 99827 99828 99832 99832 99834 99836 99837 99839 99839 99840 99840 99841 99843 99845 99845 99846 99847 99848 998
48 99852 99852 99853 99854 99854 99855 99856 99857 99858 99860 99860 99860 99860 99861 99863 99865 99865 99866 99869 99870 99871 99873 99874 998
75 99875 99875 99876 99877 99879 99880 99881 99882 99882 99883 99883 99883 99884 99885 99885 99886 99888 99891 99891 99896 99896 99897 998
97 99897 99898 99901 99905 99905 99905 99908 99909 99910 99911 99912 99912 99915 99916 99918 99920 99920 99922 99922 99927 99927 99928 99931 99932 999
33 99934 99934 99935 99936 99936 99937 99939 99940 99941 99941 99943 99946 99947 99947 99948 99949 99950 99950 99951 99952 99952 99953 99953 999
54 99954 99956 99956 99957 99958 99960 99961 99963 99964 99964 99965 99965 99966 99967 99968 99969 99974 99975 99975 99975 99979 99979 99980 999
80 99982 99983 99984 99985 99985 99985 99987 99987 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 99988 999
100000 random numbers sorted by using bubble sorting algorithms.

Time taken for bubble sort in seconds: 2.880000
```

Running through the test phase of the program with Quick Sort, the results are as shown

Quick Sort 100 Words:

After execution the quick sort algorithm displays that the time taken to process 100 random numbers was 0.000 seconds; this is due to the algorithm for quick sort is too quick, this is also dependant on the computer.

```
*****
*      Benchmarking Algorithms      *
*      Harshit Verma                *
*      ID: 200978548                *
*****

Enter the random numbrs: 100

Choose the sorting type:

1. Quick sort
2. Merge sort
3. Selection sort
4. Bubble sort
5. Exit
6. For another Number of Elements
1
Option: 1

The 100 random numbers after sorting :
0      0      0      3      3      3      5      5      56      6      8      8      11      12      12      13      14      14      14      16      18      19      19
0      21      22      22      23      25      26      27      28      29      30      30      30      33      34      34      35      36      37      37      38      38      40      41
43      45      47      49      50      51      51      52      56      57      58      59      63      64      64      65      67      67      68      69      71      72      72
75      78      78      80      81      84      84      84      85      85      85      88      88      91      93      93      95      96      96      97      97      97      97

100 random numbers sorted by using quick sorting algorithms.

Time taken for quick sort in seconds: 0.000000
```

Quick Sort 1000 Words:

After execution the quick sort algorithm displays that the time taken to process 1000 random numbers was 0.000 seconds as same as for 100 numbers; this is due to the algorithm for quick sort is too quick, this is also dependant on the computer.

```
Option: 1

The 1000 random numbers after sorting :
0 1 2 3 3 6 6 6 7 7 7 8 9 11 12 12 13 14 14 14 15 15 16 17
8 18 20 21 22 24 26 29 30 30 31 32 32 33 34 36 38 39 39 40 41 43 46 46
6 49 51 52 53 53 53 55 58 58 60 60 61 61 61 62 62 63 64 68 68 70 70 71
8 78 79 79 81 81 81 84 85 86 87 87 88 89 89 90 91 91 92 92 93 93 97 97
01 101 102 104 106 108 109 110 113 114 115 116 119 120 120 120 121 121 122 122 123 125 125 125
121 132 133 134 136 138 139 140 141 143 144 149 149 152 152 153 153 154 154 154 155 155 156
158 159 161 161 163 163 166 166 166 167 167 169 169 170 171 173 173 174 174 175 176 178 178
180 180 181 181 182 183 184 185 186 187 189 190 191 192 193 194 194 195 195 195 196 196 198 198
285 205 206 206 206 207 207 208 209 213 213 215 215 216 216 218 218 220 221 221 222 223 224 225
229 233 234 236 236 237 238 238 239 239 241 242 242 243 245 250 251 255 255 261 263 264 264
270 270 272 274 275 275 276 276 278 279 279 283 283 284 285 286 287 288 290 290 290 295 299 299
300 305 307 308 308 310 310 310 314 315 315 317 317 317 318 322 322 322 323 328 332 334 335 335
344 344 349 349 351 351 352 353 353 354 354 356 356 358 359 359 360 360 360 361 362 363 363 363
367 368 371 372 373 374 374 375 376 376 376 379 379 381 381 382 384 386 387 389 390 392 393 393
394 395 397 398 398 399 399 401 405 406 409 410 410 411 412 412 412 413 414 415 416 416 416
424 429 430 431 431 431 432 432 436 436 437 438 438 440 440 443 444 446 447 448 449 449 451 454
456 458 458 458 458 459 460 460 460 461 461 462 463 463 463 464 464 464 465 465 465 465 467 469
471 472 473 473 474 475 475 475 476 476 477 478 478 479 480 482 482 483 486 487 487 488 490 490
493 494 495 495 495 496 500 500 502 503 504 505 506 507 510 514 514 515 515 516 516 516 517 517 517
525 527 530 530 532 532 535 536 536 536 537 537 541 542 544 545 545 546 547 548 550 550 550 552 552
556 556 557 557 558 560 560 562 562 562 563 564 564 567 568 571 571 573 574 576 576 577 577 577 577
580 581 582 583 583 584 586 586 586 587 587 587 588 588 589 589 591 591 591 591 591 591 591 591 591
598 599 599 599 604 604 605 606 607 607 607 610 611 612 612 614 616 617 618 618 620 620 621 621 621
625 626 626 627 627 630 632 633 633 633 634 635 635 636 636 637 637 637 638 638 639 639 640 641 641
643 644 644 645 648 648 651 651 651 651 651 652 654 654 655 656 657 657 657 658 658 661 662 663 664
667 668 671 671 671 672 672 674 676 678 679 679 679 680 681 683 683 686 687 687 687 690 691 692 692 692
697 700 700 706 706 706 708 709 709 709 710 715 715 720 721 722 722 726 726 727 727 730 730 730 731 731
738 739 739 741 741 742 742 743 744 744 745 745 746 747 748 748 748 748 749 751 751 751 754 755 755
757 758 759 760 763 764 765 766 766 767 768 770 772 773 776 777 778 778 779 781 783 783 784 784 784
789 790 792 793 793 794 796 801 801 802 802 803 806 807 808 808 810 810 811 812 812 812 812 813 813
818 819 820 821 823 823 823 824 825 826 827 827 828 829 829 829 829 830 831 831 832 832 832 834 834
839 839 839 840 841 841 842 842 843 843 843 843 844 845 845 846 847 849 849 852 853 854 854 856 856
859 863 864 864 864 865 865 865 866 868 869 869 870 871 872 873 873 874 879 879 880 881 881 881 881
885 886 887 887 889 890 890 891 894 894 895 896 896 897 898 900 900 901 902 904 904 905 905 908 908
911 912 913 914 915 917 917 919 919 921 922 923 923 926 927 928 930 930 936 937 938 940 940 940 940
944 945 946 950 950 951 953 954 959 959 959 960 960 960 961 961 962 963 964 965 965 970 971 971 971
973 974 975 977 978 979 981 981 982 982 982 982 983 984 985 986 986 987 987 991 992 993 993 994 995 995
998
1000 random numbers sorted by using quick sorting algorithms.

Time taken for quick sort in seconds: 0.000000
```

Quick Sort 1,000,000 Words:

After execution the quick sort algorithm displays that the time taken to process 1,000,000 random numbers was 0.39000 seconds.

```
378 999378 999379 999380 999381 999381 999383 999383 999383 999384 999384 999386 999388 999388 999388 999392 999392 999393 999394 999397 999398 999398 999398 999399 999399 999399 999
400 999400 999401 999401 999404 999405 999406 999407 999407 999408 999409 999409 999411 999413 999417 999420 999420 999422 999423 999424 999427 999428 999428 999428 999429 999
430 999431 999432 999432 999432 999433 999434 999434 999436 999437 999439 999440 999441 999441 999441 999442 999442 999442 999443 999445 999446 999446 999446 999447 999447 999
449 999452 999452 999454 999454 999455 999456 999459 999460 999461 999465 999465 999466 999466 999466 999466 999468 999468 999469 999469 999470 999470 999471 999471 999
471 999473 999474 999476 999479 999479 999479 999482 999484 999484 999485 999485 999486 999488 999489 999490 999492 999492 999492 999492 999495 999496 999496 999496 999
496 999498 999498 999498 999501 999502 999504 999504 999504 999505 999507 999509 999500 999510 999510 999511 999512 999515 999520 999521 999523 999523 999523 999523 999
523 999523 999526 999526 999527 999528 999529 999530 999531 999532 999533 999534 999535 999536 999536 999537 999538 999539 999541 999541 999543 999544 999544 999544 999544 999
544 999544 999544 999546 999546 999550 999552 999553 999556 999557 999558 999559 999560 999564 999565 999566 999566 999566 999570 999571 999572 999573 999574 999576 999576 999
578 999580 999580 999581 999583 999584 999584 999584 999584 999585 999585 999588 999592 999594 999596 999597 999598 999599 999602 999604 999604 999606 999606 999606 999606 999
606 999607 999608 999608 999609 999609 999610 999610 999611 999612 999612 999613 999614 999615 999617 999620 999623 999623 999624 999624 999625 999627 999628 999629 999631 999
631 999631 999631 999632 999632 999633 999635 999635 999636 999638 999639 999640 999641 999642 999643 999644 999644 999644 999645 999649 999650 999651 999651 999652 999652 999
654 999654 999659 999660 999661 999661 999662 999663 999664 999667 999667 999669 999670 999673 999674 999674 999674 999677 999678 999679 999679 999680 999680 999680 999680 999
681 999687 999688 999689 999691 999695 999696 999696 999696 999696 999698 999703 999703 999707 999707 999708 999710 999712 999712 999714 999714 999715 999715 999715 999
715 999715 999717 999718 999719 999720 999722 999726 999727 999729 999730 999730 999730 999731 999733 999734 999736 999738 999740 999742 999742 999743 999744 999745 999
745 999745 999746 999747 999748 999748 999748 999749 999750 999750 999752 999755 999757 999758 999761 999762 999766 999767 999767 999768 999771 999771 999773 999773 999774 999774 999
774 999775 999775 999775 999775 999776 999776 999777 999779 999779 999780 999781 999783 999784 999784 999784 999789 999789 999790 999791 999792 999793 999793 999794 999794 999
795 999796 999798 999798 999799 999799 999800 999801 999802 999805 999806 999806 999807 999807 999808 999812 999812 999813 999814 999814 999819 999820 999820 999820 999820 999
821 999822 999822 999823 999824 999827 999828 999828 999829 999831 999832 999834 999835 999835 999837 999838 999839 999839 999840 999841 999841 999841 999842 999842 999842 999
845 999849 999850 999851 999853 999855 999857 999858 999859 999860 999861 999862 999863 999864 999865 999866 999867 999868 999868 999869 999870 999870 999870 999870 999870 999
870 999870 999872 999872 999873 999873 999874 999874 999876 999877 999878 999879 999880 999882 999883 999884 999886 999887 999888 999888 999888 999889 999889 999890 999890 999890 999
893 999894 999894 999899 999899 999899 999899 999899 999900 999900 999900 999902 999903 999903 999904 999904 999905 999906 999907 999908 999908 999909 999909 999910 999912 999912 999
912 999912 999913 999913 999913 999914 999914 999915 999915 999918 999920 999920 999921 999922 999922 999922 999923 999924 999924 999925 999926 999927 999927 999927 999927 999927 999
928 999929 999932 999932 999932 999933 999934 999934 999934 999937 999939 999939 999940 999941 999942 999944 999945 999948 999949 999950 999952 999953 999953 999953 999953 999953 999
953 999953 999953 999954 999955 999955 999956 999956 999959 999961 999961 999963 999964 999965 999965 999967 999969 999969 999971 999972 999973 999973 999974 999974 999974 999974 999
974 999976 999979 999979 999980 999983 999983 999984 999984 999985 999986 999986 999986 999989 999990 999990 999991 999991 999992 999992 999993 999995 999995 999996 999996 999996 999
997 999997 999997 999998 999998
1000000 random numbers sorted by using quick sorting algorithms.

Time taken for quick sort in seconds: 0.390000
```


Quick Sort 5,000,000 Words:

After execution the quick sort algorithm displays that the time taken to process 1,000,000 random numbers was 1.98000 seconds.

| | | | | | | | | | | | | | | | | | | | | | | |
|--------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 939258 | 4999259 | 4999261 | 4999261 | 4999262 | 4999262 | 4999264 | 4999265 | 4999267 | 4999268 | 4999268 | 4999268 | 4999271 | 4999271 | 4999271 | 4999271 | 4999271 | 4999271 | 4999271 | 4999275 | 4999275 | 4999277 | 4999277 |
| 939282 | 4999283 | 4999283 | 4999283 | 4999285 | 4999285 | 4999286 | 4999287 | 4999287 | 4999288 | 4999288 | 4999288 | 4999289 | 4999291 | 4999295 | 4999296 | 4999296 | 4999299 | 4999299 | 4999301 | 4999302 | 4999305 | 4999306 |
| 939306 | 4999307 | 4999307 | 4999308 | 4999308 | 4999308 | 4999309 | 4999310 | 4999310 | 4999311 | 4999311 | 4999314 | 4999315 | 4999316 | 4999317 | 4999318 | 4999318 | 4999321 | 4999321 | 4999323 | 4999323 | 4999326 | 4999328 |
| 939329 | 4999330 | 4999332 | 4999332 | 4999333 | 4999333 | 4999334 | 4999334 | 4999335 | 4999335 | 4999337 | 4999339 | 4999342 | 4999343 | 4999345 | 4999348 | 4999351 | 4999352 | 4999353 | 4999358 | 4999358 | 4999359 | 4999361 |
| 939362 | 4999363 | 4999364 | 4999364 | 4999365 | 4999365 | 4999366 | 4999368 | 4999368 | 4999370 | 4999371 | 4999372 | 4999373 | 4999374 | 4999374 | 4999374 | 4999374 | 4999377 | 4999378 | 4999378 | 4999378 | 4999380 | 4999381 |
| 939381 | 4999382 | 4999383 | 4999383 | 4999385 | 4999385 | 4999386 | 4999388 | 4999388 | 4999389 | 4999390 | 4999392 | 4999393 | 4999393 | 4999393 | 4999396 | 4999396 | 4999397 | 4999398 | 4999399 | 4999401 | 4999401 | 4999403 |
| 939404 | 4999405 | 4999406 | 4999406 | 4999408 | 4999408 | 4999409 | 4999410 | 4999411 | 4999412 | 4999413 | 4999414 | 4999415 | 4999416 | 4999417 | 4999418 | 4999419 | 4999420 | 4999421 | 4999422 | 4999423 | 4999424 | 4999425 |
| 939434 | 4999434 | 4999437 | 4999438 | 4999439 | 4999443 | 4999444 | 4999446 | 4999447 | 4999447 | 4999447 | 4999448 | 4999449 | 4999450 | 4999452 | 4999452 | 4999453 | 4999453 | 4999454 | 4999457 | 4999458 | 4999459 | 4999460 |
| 939461 | 4999462 | 4999462 | 4999463 | 4999465 | 4999465 | 4999465 | 4999468 | 4999468 | 4999470 | 4999472 | 4999472 | 4999473 | 4999474 | 4999474 | 4999481 | 4999482 | 4999484 | 4999484 | 4999485 | 4999486 | 4999487 | 4999488 |
| 939491 | 4999491 | 4999491 | 4999491 | 4999492 | 4999492 | 4999492 | 4999494 | 4999494 | 4999498 | 4999499 | 4999501 | 4999504 | 4999504 | 4999505 | 4999505 | 4999506 | 4999508 | 4999508 | 4999509 | 4999509 | 4999510 | 4999511 |
| 95112 | 4999513 | 4999515 | 4999515 | 4999516 | 4999517 | 4999518 | 4999519 | 4999520 | 4999521 | 4999521 | 4999522 | 4999523 | 4999524 | 4999525 | 4999527 | 4999528 | 4999528 | 4999529 | 4999531 | 4999531 | 4999532 | 4999532 |
| 95333 | 4999534 | 4999535 | 4999535 | 4999537 | 4999538 | 4999539 | 4999539 | 4999540 | 4999540 | 4999542 | 4999543 | 4999543 | 4999544 | 4999544 | 4999548 | 4999549 | 4999552 | 4999553 | 4999555 | 4999555 | 4999557 | 4999558 |
| 95559 | 4999562 | 4999563 | 4999567 | 4999567 | 4999567 | 4999568 | 4999570 | 4999570 | 4999571 | 4999571 | 4999571 | 4999572 | 4999573 | 4999575 | 4999575 | 4999578 | 4999579 | 4999580 | 4999580 | 4999581 | 4999582 | 4999584 |
| 95861 | 4999587 | 4999587 | 4999589 | 4999591 | 4999591 | 4999591 | 4999594 | 4999594 | 4999597 | 4999597 | 4999600 | 4999601 | 4999602 | 4999603 | 4999605 | 4999607 | 4999608 | 4999610 | 4999613 | 4999614 | 4999616 | 4999617 |
| 96181 | 4999619 | 4999619 | 4999619 | 4999620 | 4999621 | 4999621 | 4999622 | 4999622 | 4999624 | 4999626 | 4999626 | 4999629 | 4999630 | 4 | | | | | | | | |

```
Time taken for quick sort in seconds: 1.980000
```

Quick Sort 10,000,000 Words:

After execution the quick sort algorithm displays that the time taken to process 1,000,000 random numbers was 3.94000 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| 9311 | 999311 | 999311 | 999312 | 999313 | 999315 | 999316 | 999317 | 999319 | 999321 | 999324 | 999326 | 999328 | 999330 | 999332 | 999334 | 999336 | 999338 | 999340 | 999342 | 999344 | 999346 | 999348 | 999350 | 999352 | 999354 | 999356 | 999358 | 999360 | 999362 | 999364 | 999366 | 999368 | 999370 | 999372 | 999374 | 999376 | 999378 | 999380 | 999382 | 999384 | 999386 | 999388 | 999390 | 999392 | 999394 | 999396 | 999398 | 999400 | 999402 | 999404 | 999406 | 999408 | 999410 | 999412 | 999414 | 999416 | 999418 | 999420 | 999422 | 999424 | 999426 | 999428 | 999430 | 999432 | 999434 | 999436 | 999438 | 999440 | 999442 | 999444 | 999446 | 999448 | 999450 | 999452 | 999454 | 999456 | 999458 | 999460 | 999462 | 999464 | 999466 | 999468 | 999470 | 999472 | 999474 | 999476 | 999478 | 999480 | 999482 | 999484 | 999486 | 999488 | 999490 | 999492 | 999494 | 999496 | 999498 | 999500 | 999502 | 999504 | 999506 | 999508 | 999510 | 999512 | 999514 | 999516 | 999518 | 999520 | 999522 | 999524 | 999526 | 999528 | 999530 | 999532 | 999534 | 999536 | 999538 | 999540 | 999542 | 999544 | 999546 | 999548 | 999550 | 999552 | 999554 | 999556 | 999558 | 999560 | 999562 | 999564 | 999566 | 999568 | 999570 | 999572 | 999574 | 999576 | 999578 | 999580 | 999582 | 999584 | 999586 | 999588 | 999590 | 999592 | 999594 | 999596 | 999598 | 999600 | 999602 | 999604 | 999606 | 999608 | 999610 | 999612 | 999614 | 999616 | 999618 | 999620 | 999622 | 999624 | 999626 | 999628 | 999630 | 999632 | 999634 | 999636 | 999638 | 999640 | 999642 | 999644 | 999646 | 999648 | 999650 | 999652 | 999654 | 999656 | 999658 | 999660 | 999662 | 999664 | 999666 | 999668 | 999670 | 999672 | 999674 | 999676 | 999678 | 999680 | 999682 | 999684 | 999686 | 999688 | 999690 | 999692 | 999694 | 999696 | 999698 | 999700 | 999702 | 999704 | 999706 | 999708 | 999710 | 999712 | 999714 | 999716 | 999718 | 999720 | 999722 | 999724 | 999726 | 999728 | 999730 | 999732 | 999734 | 999736 | 999738 | 999740 | 999742 | 999744 | 999746 | 999748 | 999750 | 999752 | 999754 | 999756 | 999758 | 999760 | 999762 | 999764 | 999766 | 999768 | 999770 | 999772 | 999774 | 999776 | 999778 | 999780 | 999782 | 999784 | 999786 | 999788 | 999790 | 999792 | 999794 | 999796 | 999798 | 999800 | 999802 | 999804 | 999806 | 999808 | 999810 | 999812 | 999814 | 999816 | 999818 | 999820 | 999822 | 999824 | 999826 | 999828 | 999830 | 999832 | 999834 | 999836 | 999838 | 999840 | 999842 | 999844 | 999846 | 999848 | 999850 | 999852 | 999854 | 999856 | 999858 | 999860 | 999862 | 999864 | 999866 | 999868 | 999870 | 999872 | 999874 | 999876 | 999878 | 999880 | 999882 | 999884 | 999886 | 999888 | 999890 | 999892 | 999894 | 999896 | 999898 | 999900 | 999902 | 999904 | 999906 | 999908 | 999910 | 999912 | 999914 | 999916 | 999918 | 999920 | 999922 | 999924 | 999926 | 999928 | 999930 | 999932 | 999934 | 999936 | 999938 | 999940 | 999942 | 999944 | 999946 | 999948 | 999950 | 999952 | 999954 | 999956 | 999958 | 999960 | 999962 | 999964 | 999966 | 999968 | 999970 | 999972 | 999974 | 999976 | 999978 | 999980 | 999982 | 999984 | 999986 | 999988 | 999990 | 999992 | 999994 | 999996 | 999998 | 1000000 |
| 1000000 random numbers sorted by using quick sorting algorithms. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

```
Time taken for quick sort in seconds: 3.940000
```

Running through the test phase of the program with Selection Sort, the results are shown below:

Selection Sort 1000 Words:

After execution the selection sort algorithm displays that the time taken to process 1000 random numbers was 0.000 seconds; this is dependant on the computer and their processors

```
The 1000 random numbers after sorting :
0 1 2 2 6 8 8 10 10 13 13 15 16 17 17 18 19 20 20 23 23 24 24 25
8 29 30 32 32 33 36 37 39 39 42 45 46 46 47 51 51 53 54 55 55 56 57
9 59 60 61 61 62 63 63 65 66 67 68 70 70 73 74 74 75 76 78 79 81 82 82
3 84 85 86 86 87 90 92 92 93 94 95 95 96 96 98 98 99 99 100 100 101 102 103 103
106 108 108 111 112 114 115 116 116 117 118 120 121 122 126 126 127 132 132 133 138 140 143 143 143
145 147 147 147 147 148 148 148 149 149 149 150 150 151 151 151 151 152 154 155 156 156 157 157
157 159 159 160 165 165 165 166 166 167 168 170 171 172 172 173 174 174 175 175 176 177 177 180 181
182 182 183 183 185 186 186 186 188 188 188 189 190 190 193 194 194 195 195 196 196 199 202 202 203
205 205 205 205 207 207 208 208 211 212 213 214 215 215 216 216 217 218 218 218 219 220 221 222 226 227
228 231 231 232 232 233 233 233 236 236 240 242 243 243 244 244 244 245 246 247 249 251 252 253 254
257 259 260 261 263 264 265 266 267 269 270 271 272 272 275 275 277 278 281 281 282 283 283 284 287 288
293 294 294 294 296 298 299 299 299 300 308 382 303 304 306 307 308 309 309 312 314 315 316 317 318
319 320 321 321 321 322 323 323 323 323 323 325 325 327 327 328 329 330 331 332 334 335 335 337 339
347 350 350 351 351 352 352 354 354 354 356 356 356 357 357 358 359 360 360 360 365 365 366 367 368
370 370 372 374 374 375 375 376 377 378 378 382 384 386 387 387 388 390 390 391 394 394 395 397 400 402
404 404 405 407 407 410 411 411 411 411 412 412 414 415 415 415 416 416 417 417 417 418 419 420 420 420
421 422 422 423 424 424 425 425 425 427 427 428 428 429 429 430 431 431 435 435 437 437 437 438 438 440
443 443 444 444 445 445 446 449 449 450 450 450 450 451 452 454 454 455 456 457 458 459 459 460 463
467 468 469 469 470 470 472 472 474 477 478 478 479 480 480 482 483 483 485 488 488 488 489 489 490 491
491 492 492 492 494 496 497 497 498 502 503 504 505 507 507 508 508 511 511 513 515 515 517 522 523
525 525 526 526 529 531 532 532 535 536 539 544 545 546 546 548 548 550 551 552 553 553 554 555 555
558 558 559 561 562 563 563 564 564 565 565 568 570 572 573 574 575 575 577 578 579 579 581 581 581
582 584 585 585 586 589 591 592 593 593 594 594 595 595 596 598 598 599 600 601 604 605 607 609 610
612 613 613 613 614 615 616 618 619 619 619 620 620 624 624 625 625 627 629 630 632 632 633 635 635
636 637 638 639 639 640 640 640 641 641 642 642 643 643 644 645 646 646 649 650 651 652 653 655 655
660 660 660 660 660 661 663 663 664 664 665 667 667 668 671 674 674 675 675 676 678 679 682 682 682
686 686 688 689 689 692 692 693 695 695 697 697 698 699 702 702 707 707 708 709 710 712 712 712 713
717 719 721 722 722 722 723 723 723 725 729 729 731 732 732 733 733 734 735 735 736 737 739 739
748 741 743 743 743 743 743 745 746 750 750 751 751 752 753 754 755 757 758 759 759 760 761 763
765 766 767 767 768 770 772 772 772 774 775 776 776 777 777 777 778 779 780 781 782 783 784 784 787
787 789 791 792 792 793 794 794 795 795 797 799 802 802 803 804 808 808 809 809 810 812 815 816 817 818
820 820 820 821 821 821 822 822 824 824 824 826 829 832 833 833 836 836 840 841 842 843 850 851 852 855
858 859 860 860 861 862 862 862 863 864 864 865 865 871 871 873 873 875 876 877 877 878 878 879 879
881 883 884 884 884 888 888 889 889 890 891 895 897 899 902 903 907 908 912 914 915 916 917 918 918
919 920 921 921 923 923 923 926 928 928 929 931 931 932 932 932 934 934 935 937 937 937 937 938
940 941 941 942 943 947 949 949 950 952 952 953 954 956 957 958 959 961 962 964 964 965 966 966 966
967 969 969 970 970 971 974 977 978 978 979 981 984 985 985 986 986 986 986 986 989 989 992 993 993
999
1000 random numbers sorted by using selection sorting algorithms.
Time taken for selection sort in seconds: 0.000000
```

Selection Sort 10,000 Words:

After execution the selection sort algorithm displays that the time taken to process 10,000 random numbers was 0.13000 seconds.

```
6 9228 9228 9230 9231 9232 9232 9232 9233 9233 9234 9236 9239 9239 9241 9243 9243 9245 9246 9247 9247 9248 9248 9248 9254 9256 925
6 9260 9260 9261 9262 9266 9268 9269 9271 9271 9271 9273 9273 9276 9277 9278 9280 9280 9289 9289 9291 9293 9295 9295 9296 9297 9299 929
9 9299 9303 9304 9305 9306 9306 9310 9310 9310 9313 9313 9316 9317 9317 9319 9320 9321 9322 9323 9324 9324 9325 9325 9326 9326 932
8 9330 9331 9334 9335 9336 9337 9337 9339 9340 9340 9341 9341 9342 9342 9342 9344 9345 9346 9346 9346 9346 9346 9347 9348 934
8 9349 9350 9351 9352 9354 9354 9354 9355 9355 9359 9360 9360 9361 9363 9365 9365 9367 9370 9370 9371 9371 9372 9372 9372 9373 937
4 9375 9376 9378 9378 9380 9382 9383 9383 9384 9385 9386 9387 9389 9389 9389 9390 9390 9391 9391 9392 9392 9392 9395 9396 939
6 9397 9398 9398 9398 9399 9400 9401 9402 9405 9406 9406 9407 9408 9408 9409 9412 9413 9413 9414 9414 9417 9417 9417 9418 9419 941
9 9423 9424 9425 9426 9426 9428 9428 9429 9429 9431 9431 9435 9441 9441 9442 9443 9447 9447 9448 9450 9451 9451 9452 9453 9453 945
3 9453 9456 9456 9457 9457 9458 9459 9460 9468 9468 9469 9470 9470 9472 9472 9475 9476 9479 9480 9481 9482 9483 9486 9486 9489 949
0 9492 9496 9496 9496 9497 9498 9499 9499 9500 9502 9503 9505 9508 9508 9510 9510 9512 9516 9516 9517 9520 9522 9523 9524 952
5 9527 9530 9532 9532 9533 9534 9537 9538 9538 9539 9541 9545 9546 9546 9546 9547 9548 9548 9548 9549 9554 9556 9557 9558 955
9 9559 9559 9559 9559 9560 9564 9566 9566 9566 9571 9571 9573 9574 9578 9578 9579 9579 9580 9581 9581 9584 9585 9586 9587 958
9 9589 9591 9591 9592 9593 9594 9596 9597 9600 9602 9603 9603 9603 9604 9605 9607 9608 9609 9611 9612 9613 9615 9615 9616
6 9617 9618 9618 9620 9620 9623 9626 9629 9630 9630 9631 9632 9632 9633 9634 9635 9636 9637 9639 9640 9640 9641 9643 9645 9645 964
5 9647 9653 9654 9655 9655 9656 9656 9659 9660 9664 9664 9666 9666 9665 9666 9667 9668 9670 9672 9672 9672 9673 9673 9673 967
6 9677 9679 9679 9679 9681 9683 9683 9684 9688 9692 9693 9693 9694 9694 9695 9698 9700 9700 9701 9701 9702 9703 9706 9708 9708
9 9710 9712 9713 9714 9716 9718 9719 9720 9721 9721 9725 9729 9729 9730 9730 9731 9731 9735 9736 9737 9737 9739 9739 9740 974
3 9744 9744 9744 9747 9749 9750 9750 9751 9752 9752 9753 9754 9755 9755 9757 9761 9761 9762 9762 9764 9769 9769 9771 9772 977
2 9772 9773 9774 9775 9775 9775 9777 9777 9777 9779 9779 9779 9780 9780 9781 9782 9783 9783 9783 9785 9785 9786 9786 9786 978
7 9788 9788 9790 9792 9794 9795 9796 9802 9802 9803 9804 9805 9806 9806 9807 9808 9808 9809 9811 9812 9812 9812 9814 9814 981
7 9817 9818 9818 9819 9820 9820 9821 9821 9822 9823 9825 9825 9826 9826 9827 9829 9829 9829 9830 9830 9831 9834 9835 9835 9836 983
7 9838 9840 9842 9843 9844 9848 9848 9848 9849 9849 9852 9853 9856 9858 9858 9859 9862 9863 9865 9867 9869 9869 9870 9870 987
0 9871 9873 9875 9875 9877 9877 9878 9878 9880 9881 9881 9881 9882 9884 9884 9885 9886 9887 9888 9889 9889 9892 9893 9894 989
5 9896 9897 9900 9900 9901 9902 9902 9903 9903 9904 9907 9907 9908 9910 9911 9912 9914 9917 9918 9918 9919 9919 9920 9921 992
2 9923 9923 9924 9926 9926 9928 9931 9933 9934 9936 9936 9937 9939 9939 9940 9941 9941 9942 9942 9943 9944 9945 9948 9948 994
9 9950 9950 9950 9951 9953 9956 9957 9957 9957 9957 9958 9959 9960 9960 9962 9963 9964 9966 9967 9968 9970 9970 9972 9972 997
2 9972 9974 9976 9977 9978 9979 9979 9982 9982 9983 9983 9983 9983 9985 9986 9987 9988 9988 9989 9989 9992 9993 9994 9994 999
4 9996 9999 9999
10000 random numbers sorted by using selection sorting algorithms.
Time taken for selection sort in seconds: 0.130000
```

Selection Sort 100,000 Words:

After execution the selection sort algorithm displays that the time taken to process 100,000 random numbers was 12.5600 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----|
| 57 | 99367 | 99367 | 99368 | 99370 | 99372 | 99375 | 99375 | 99375 | 99376 | 99376 | 99377 | 99379 | 99380 | 99381 | 99381 | 99381 | 99381 | 99381 | 99382 | 99383 | 99383 | 99384 | 99385 | 99386 | 99387 | 99388 | 993 |
| 58 | 99389 | 99389 | 99389 | 99390 | 99391 | 99392 | 99392 | 99393 | 99393 | 99394 | 99396 | 99398 | 99400 | 99403 | 99403 | 99404 | 99405 | 99407 | 99410 | 99410 | 99411 | 99413 | 99414 | 99415 | 99416 | 99417 | 994 |
| 59 | 99419 | 99419 | 99419 | 99419 | 99420 | 99420 | 99421 | 99422 | 99422 | 99423 | 99426 | 99427 | 99428 | 99428 | 99430 | 99430 | 99431 | 99431 | 99433 | 99433 | 99434 | 99434 | 99436 | 99436 | 99439 | 99439 | 994 |
| 60 | 99441 | 99441 | 99443 | 99443 | 99443 | 99444 | 99445 | 99446 | 99446 | 99448 | 99449 | 99449 | 99450 | 99450 | 99450 | 99451 | 99451 | 99452 | 99452 | 99452 | 99454 | 99457 | 99457 | 99458 | 99458 | 994 | |
| 61 | 99463 | 99463 | 99468 | 99469 | 99470 | 99470 | 99471 | 99472 | 99473 | 99474 | 99476 | 99480 | 99480 | 99481 | 99482 | 99484 | 99484 | 99487 | 99488 | 99488 | 99489 | 99490 | 99491 | 99493 | 99495 | 994 | |
| 62 | 99495 | 99496 | 99496 | 99497 | 99498 | 99498 | 99500 | 99502 | 99504 | 99506 | 99509 | 99509 | 99510 | 99511 | 99511 | 99512 | 99513 | 99513 | 99514 | 99514 | 99514 | 99514 | 99514 | 99515 | 99516 | 995 | |
| 63 | 99516 | 99517 | 99518 | 99519 | 99519 | 99519 | 99520 | 99521 | 99521 | 99525 | 99527 | 99527 | 99530 | 99531 | 99532 | 99532 | 99533 | 99533 | 99533 | 99534 | 99535 | 99535 | 99535 | 99535 | 99536 | 995 | |
| 64 | 99541 | 99544 | 99544 | 99546 | 99546 | 99547 | 99548 | 99551 | 99551 | 99555 | 99556 | 99560 | 99560 | 99564 | 99565 | 99566 | 99566 | 99568 | 99569 | 99569 | 99570 | 99571 | 99571 | 99571 | 99572 | 995 | |
| 65 | 99574 | 99574 | 99575 | 99577 | 99578 | 99578 | 99578 | 99579 | 99580 | 99580 | 99581 | 99581 | 99582 | 99585 | 99586 | 99588 | 99588 | 99592 | 99593 | 99593 | 99594 | 99594 | 99595 | 99595 | 99595 | 995 | |
| 66 | 99597 | 99597 | 99597 | 99597 | 99598 | 99599 | 99599 | 99600 | 99603 | 99606 | 99607 | 99607 | 99608 | 99608 | 99608 | 99611 | 99612 | 99612 | 99613 | 99614 | 99614 | 99614 | 99615 | 99616 | 99616 | 99617 | 996 |
| 67 | 99618 | 99618 | 99619 | 99620 | 99621 | 99621 | 99623 | 99623 | 99626 | 99628 | 99628 | 99630 | 99631 | 99631 | 99631 | 99634 | 99634 | 99634 | 99634 | 99636 | 99637 | 99639 | 99639 | 99641 | 99641 | 996 | |
| 68 | 99642 | 99643 | 99643 | 99644 | 99645 | 99645 | 99646 | 99646 | 99647 | 99648 | 99648 | 99649 | 99649 | 99650 | 99651 | 99652 | 99652 | 99654 | 99655 | 99655 | 99656 | 99656 | 99656 | 99658 | 99658 | 996 | |
| 69 | 99659 | 99660 | 99660 | 99663 | 99665 | 99666 | 99667 | 99667 | 99667 | 99668 | 99668 | 99669 | 99671 | 99673 | 99674 | 99674 | 99676 | 99677 | 99680 | 99685 | 99687 | 99687 | 99688 | 99688 | 99689 | 996 | |
| 70 | 99694 | 99695 | 99697 | 99697 | 99698 | 99699 | 99700 | 99700 | 99702 | 99702 | 99703 | 99704 | 99704 | 99707 | 99707 | 99712 | 99713 | 99713 | 99715 | 99716 | 99716 | 99716 | 99717 | 99718 | 99719 | 997 | |
| 71 | 99723 | 99723 | 99723 | 99724 | 99724 | 99725 | 99725 | 99725 | 99726 | 99727 | 99727 | 99728 | 99728 | 99730 | 99730 | 99730 | 99730 | 99731 | 99731 | 99731 | 99732 | 99734 | 99734 | 99736 | 99736 | 997 | |
| 72 | 99737 | 99737 | 99738 | 99739 | 99740 | 99741 | 99741 | 99741 | 99741 | 99743 | 99746 | 99749 | 99752 | 99754 | 99756 | 99756 | 99759 | 99761 | 99761 | 99766 | 99767 | 99768 | 99769 | 99774 | 99774 | 997 | |
| 73 | 99775 | 99776 | 99776 | 99777 | 99778 | 99781 | 99781 | 99783 | 99783 | 99785 | 99785 | 99787 | 99788 | 99790 | 99791 | 99792 | 99793 | 99794 | 99795 | 99795 | 99796 | 99797 | 99798 | 99799 | 99800 | 998 | |
| 74 | 99804 | 99808 | 99808 | 99809 | 99812 | 99813 | 99813 | 99814 | 99816 | 99817 | 99819 | 99819 | 99820 | 99820 | 99821 | 99822 | 99824 | 99825 | 99825 | 99827 | 99828 | 99829 | 99830 | 99831 | 99831 | 998 | |
| 75 | 99832 | 99832 | 99834 | 99835 | 99835 | 99836 | 99836 | 99837 | 99839 | 99840 | 99841 | 99841 | 99842 | 99843 | 99843 | 99844 | 99844 | 99846 | 99846 | 99847 | 99848 | 99850 | 99851 | 99851 | 998 | | |
| 76 | 99852 | 99854 | 99854 | 99855 | 99855 | 99856 | 99857 | 99860 | 99860 | 99862 | 99862 | 99863 | 99863 | 99864 | 99865 | 99865 | 99866 | 99869 | 99869 | 99870 | 99875 | 99875 | 99876 | 99877 | 998 | | |
| 77 | 99879 | 99879 | 99879 | 99880 | 99880 | 99880 | 99881 | 99881 | 99881 | 99881 | 99884 | 99884 | 99885 | 99887 | 99887 | 99888 | 99888 | 99888 | 99888 | 99889 | 99890 | 99890 | 99890 | 99890 | 99890 | 998 | |
| 78 | 99898 | 99901 | 99905 | 99905 | 99907 | 99910 | 99911 | 99911 | 99912 | 99912 | 99913 | 99913 | 99914 | 99914 | 99915 | 99915 | 99916 | 99917 | 99919 | 99920 | 99922 | 99923 | 99923 | 99926 | 999 | | |
| 79 | 99926 | 99930 | 99930 | 99932 | 99932 | 99935 | 99935 | 99936 | 99937 | 99939 | 99940 | 99941 | 99941 | 99942 | 99945 | 99948 | 99949 | 99950 | 99950 | 99951 | 99952 | 99952 | 99953 | 99954 | 99957 | 999 | |
| 80 | 99960 | 99960 | 99960 | 99961 | 99962 | 99962 | 99963 | 99964 | 99964 | 99965 | 99965 | 99966 | 99967 | 99967 | 99968 | 99968 | 99969 | 99971 | 99974 | 99975 | 99975 | 99976 | 99978 | 99979 | 99979 | 999 | |
| 81 | 99980 | 99984 | 99984 | 99984 | 99986 | 99990 | 99990 | 99991 | 99992 | 99992 | 99993 | 99994 | 99994 | 99995 | 99995 | 99997 | 99998 | 99998 | 99998 | 99998 | 99998 | 99998 | 99999 | 99999 | 99999 | 999 | |
| 100000 random numbers sorted by using selection sorting algorithms. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Time taken for selection sort in seconds: 12.560000 | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Selection Sort 500,000 Words:

After execution the selection sort algorithm displays that the time taken to process 500,000 random numbers was 315.5600 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|
| 416 | 499397 | 499398 | 499402 | 499402 | 499403 | 499403 | 499404 | 499404 | 499405 | 499406 | 499407 | 499408 | 499409 | 499409 | 499410 | 499410 | 499410 | 499410 | 499410 | 499410 | 499412 | 499412 | 499412 | 499413 | 499413 | 499414 | 499416 | 499 |
| 439 | 499417 | 499417 | 499419 | 499419 | 499421 | 499422 | 499423 | 499424 | 499424 | 499425 | 499427 | 499429 | 499438 | 499431 | 499432 | 499432 | 499436 | 499436 | 499436 | 499436 | 499437 | 499437 | 499437 | 499437 | 499437 | 499439 | 499 | |
| 462 | 499440 | 499441 | 499441 | 499443 | 499443 | 499444 | 499445 | 499446 | 499446 | 499447 | 499447 | 499448 | 499448 | 499449 | 499449 | 499452 | 499452 | 499453 | 499455 | 499456 | 499457 | 499460 | 499460 | 499462 | 499462 | 499466 | 499 | |
| 498 | 499463 | 499463 | 499463 | 499465 | 499469 | 499470 | 499471 | 499476 | 499480 | 499481 | 499482 | 499482 | 499483 | 499484 | 499487 | 499487 | 499488 | 499489 | 499490 | 499493 | 499495 | 499496 | 499496 | 499496 | 499498 | 499498 | 499 | |
| 526 | 499501 | 499501 | 499502 | 499503 | 499504 | 499504 | 499506 | 499506 | 499510 | 499510 | 499511 | 499512 | 499515 | 499516 | 499516 | 499517 | 499519 | 499519 | 499522 | 499523 | 499523 | 499523 | 499523 | 499524 | 499524 | 499526 | 499 | |
| 558 | 499526 | 499527 | 499527 | 499527 | 499528 | 499528 | 499528 | 499529 | 499529 | 499534 | 499535 | 499537 | 499537 | 499538 | 499539 | 499540 | 499541 | 499547 | 499549 | 499550 | 499551 | 499551 | 499553 | 499553 | 499555 | 499555 | 499 | |
| 586 | 499559 | 499559 | 499560 | 499560 | 499563 | 499564 | 499566 | 499566 | 499566 | 499567 | 499567 | 499567 | 499569 | 499569 | 499570 | 499573 | 499577 | 499578 | 499578 | 499578 | 499579 | 499580 | 499584 | 499584 | 499584 | 499584 | 499 | |
| 609 | 499588 | 499590 | 499592 | 499594 | 499595 | 499596 | 499596 | 499597 | 499597 | 499598 | 499598 | 499599 | 499602 | 499603 | 499603 | 499604 | 499604 | 499605 | 499606 | 499606 | 499607 | 499607 | 499608 | 499608 | 499608 | 499608 | 499 | |
| 640 | 499611 | 499612 | 499614 | 499615 | 499615 | 499620 | 499620 | 499621 | 499621 | 499623 | 499625 | 499625 | 499626 | 499626 | 499626 | 499628 | 499630 | 499631 | 499632 | 499633 | 499634 | 499634 | 499635 | 499638 | 499639 | 499 | | |
| 668 | 499641 | 499642 | 499642 | 499642 | 499642 | 499643 | 499644 | 499646 | 499648 | 499649 | 499649 | 499650 | 499653 | 499655 | 499655 | 499659 | 499659 | 499660 | 499661 | 499661 | 499662 | 499663 | 499664 | 499665 | 499668 | 499668 | 499 | |
| 698 | 499670 | 499670 | 499671 | 499675 | 499675 | 499675 | 499676 | 499679 | 499680 | 499683 | 499684 | 499685 | 499688 | 499689 | 499691 | 499691 | 499692 | 499693 | 499693 | 499694 | 499695 | 499695 | 499696 | 499696 | 499698 | 499698 | 499 | |
| 717 | 499699 | 499700 | 499700 | 499701 | 499702 | 499702 | 499704 | 499705 | 499707 | 499708 | 499709 | 499709 | 499710 | 499710 | 499711 | 499711 | 499711 | 499711 | 499711 | 499713 | 499713 | 499713 | 499716 | 499716 | 499717 | 499717 | 499 | |
| 745 | 499717 | 499719 | 499719 | 499720 | 499720 | 499721 | 499721 | 499722 | 499724 | 499724 | 499725 | 499726 | 499731 | 499732 | 499732 | 499733 | 499734 | 499734 | 499735 | 499738 | 499743 | 499743 | 499744 | 499744 | 499744 | 499744 | 499 | |
| 776 | 499746 | 499750 | 499753 | 499754 | 499755 | 499756 | 499757 | 499757 | 499758 | 499759 | 499760 | 499761 | 499763 | 499764 | 499766 | 499766 | 499766 | 499767 | 499769 | 499770 | 499770 | 499774 | 499775 | 499775 | 499776 | 499776 | 499 | |
| 804 | 499776 | 499777 | 499779 | 499780 | 499781 | 499785 | 499785 | 499787 | 499789 | 499789 | 499789 | 499792 | 499794 | 499795 | 499795 | 499797 | 499798 | 499798 | 499799 | 499799 | 499801 | 499802 | 499802 | 499803 | 499803 | 499803 | 499 | |
| 831 | 499805 | 499805 | 499805 | 499806 | 499807 | 499812 | 499812 | 499814 | 499815 | 499815 | 499817 | 499821 | 499822 | 499823 | 499823 | 499824 | 499825 | 499825 | 499826 | 499826 | 499828 | 499828 | 499829 | 499830 | 499831 | 499831 | 499 | |
| 859 | 499832 | 499837 | 499838 | 499839 | 499840 | 499842 | 499842 | 499843 | 499845 | 499846 | 499847 | 499848 | 499849 | 499849 | 499849 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499850 | 499 | |
| 889 | 499860 | 499864 | 499866 | 499868 | 499868 | 499868 | 499868 | 499869 | 499869 | 499872 | 499872 | 499873 | 499874 | 499876 | 499877 | 499878 | 499881 | 499882 | 499883 | 499884 | 499886 | 499890 | 499891 | 499891 | 499891 | 499891 | 499 | |
| 919 | 499891 | 499893 | 499894 | 499894 | 499896 | 499897 | 499900 | 499900 | 499901 | 499901 | 499902 | 499902 | 499903 | 499904 | 499904 | 499906 | 499910 | 499911 | 499912 | 499913 | 499914 | 499915 | 499915 | 499915 | 499915 | 499915 | 499 | |
| 923 | 499924 | 499924 | 499925 | 499927 | 499928 | 499928 | 499928 | 499928 | 499928 | 499929 | 499929 | 499930 | 499932 | 499932 | 499932 | 499932 | 499933 | 499934 | 499935 | 499937 | 499938 | 499938 | 499938 | 499938 | 499939 | 499939 | 499 | |
| 933 | 499941 | 499941 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499943 | 499 | |
| 963 | 499956 | 499967 | 499967 | 499968 | 499969 | 499969 | 499970 | 499971 | 499972 | 499973 | 499975 | 499976 | 499977 | 499979 | 499982 | 499982 | 499983 | 499984 | 499986 | 499987 | 499987 | 499988 | 499988 | 499988 | 499988 | 499988 | 499 | |
| 993 | 499994 | 499994 | 499995 | 499995 | 499996 | 499996 | 499997 | 499997 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499998 | 499 | |
| 500000 random numbers sorted by using selection sorting algorithms. | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Time taken for selection sort in seconds: 315.560000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Running through the test phase of the program with Merge Sort, the results are shown below:

Merge Sort 1000 Words:

After execution the merge sort algorithm displays that the time taken to process 1000 random numbers was 0.000 second; this is due to the algorithm for merge sort is too quick, this is also dependant on the computer.

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 489 | 489 | 489 | 410 | 418 | 410 | 411 | 413 | 416 | 417 | 418 | 419 | 422 | 422 | 425 | 426 | 426 | 427 | 427 | 427 | 427 | 429 | 431 | 431 | 431 | 432 | 433 |
| 434 | 436 | 438 | 439 | 439 | 442 | 444 | 445 | 445 | 448 | 449 | 454 | 455 | 456 | 457 | 457 | 458 | 459 | 459 | 459 | 459 | 460 | 460 | 460 | 461 | 464 | 464 |
| 465 | 465 | 465 | 468 | 468 | 468 | 470 | 474 | 476 | 476 | 477 | 479 | 480 | 480 | 480 | 480 | 480 | 481 | 482 | 483 | 485 | 486 | 487 | 487 | 488 | 491 | 492 |
| 493 | 494 | 496 | 498 | 498 | 498 | 499 | 500 | 501 | 501 | 504 | 504 | 504 | 505 | 505 | 509 | 509 | 512 | 515 | 516 | 516 | 518 | 519 | 519 | 519 | 520 | 520 |
| 521 | 522 | 523 | 526 | 526 | 527 | 528 | 530 | 530 | 530 | 532 | 532 | 533 | 533 | 534 | 534 | 534 | 534 | 535 | 535 | 535 | 535 | 538 | 538 | 539 | 539 | 540 |
| 542 | 545 | 545 | 546 | 547 | 548 | 552 | 552 | 552 | 553 | 554 | 554 | 554 | 555 | 555 | 557 | 557 | 558 | 558 | 558 | 558 | 560 | 560 | 562 | 563 | 564 | |
| 564 | 566 | 570 | 570 | 570 | 570 | 571 | 571 | 571 | 572 | 572 | 573 | 580 | 581 | 581 | 583 | 583 | 585 | 585 | 586 | 586 | 587 | 589 | 591 | 591 | 594 | 595 |
| 597 | 597 | 599 | 599 | 600 | 601 | 602 | 603 | 604 | 604 | 606 | 606 | 606 | 606 | 607 | 607 | 608 | 609 | 610 | 611 | 611 | 611 | 611 | 611 | 611 | 614 | 614 |
| 616 | 616 | 618 | 619 | 619 | 620 | 620 | 621 | 621 | 622 | 623 | 625 | 626 | 626 | 629 | 631 | 634 | 634 | 634 | 635 | 635 | 637 | 640 | 640 | 641 | 641 | 643 |
| 644 | 646 | 646 | 646 | 647 | 648 | 648 | 649 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 658 | 659 | 661 | 662 | 663 | 667 | 671 | 671 | 672 | 672 | 674 | 674 |
| 675 | 676 | 676 | 682 | 684 | 686 | 686 | 686 | 686 | 688 | 689 | 689 | 692 | 692 | 693 | 695 | 698 | 699 | 702 | 703 | 704 | 705 | 707 | 708 | 709 | 710 | 710 |
| 711 | 711 | 712 | 712 | 713 | 713 | 713 | 715 | 716 | 718 | 718 | 719 | 720 | 721 | 722 | 724 | 725 | 726 | 729 | 729 | 734 | 735 | 736 | 737 | 737 | 738 | 739 |
| 739 | 742 | 743 | 744 | 744 | 745 | 746 | 746 | 746 | 747 | 747 | 750 | 750 | 753 | 753 | 754 | 754 | 754 | 755 | 756 | 758 | 759 | 762 | 762 | 764 | 765 | 765 |
| 767 | 769 | 771 | 773 | 774 | 774 | 776 | 777 | 777 | 777 | 777 | 779 | 779 | 780 | 780 | 781 | 783 | 784 | 784 | 785 | 785 | 786 | 788 | 791 | 791 | 794 | 795 |
| 799 | 801 | 803 | 804 | 804 | 805 | 806 | 806 | 807 | 808 | 811 | 811 | 812 | 813 | 813 | 814 | 815 | 815 | 817 | 817 | 818 | 818 | 818 | 818 | 819 | 819 | 819 |
| 820 | 821 | 822 | 822 | 822 | 823 | 823 | 824 | 824 | 824 | 824 | 825 | 826 | 826 | 826 | 829 | 831 | 831 | 831 | 832 | 834 | 835 | 836 | 837 | 841 | 841 | 841 |
| 842 | 843 | 844 | 846 | 846 | 847 | 847 | 848 | 848 | 851 | 852 | 853 | 855 | 856 | 856 | 859 | 861 | 863 | 863 | 866 | 867 | 867 | 868 | 868 | 868 | 870 | 870 |
| 876 | 872 | 873 | 875 | 876 | 877 | 878 | 879 | 879 | 881 | 882 | 884 | 885 | 885 | 885 | 886 | 886 | 888 | 890 | 891 | 891 | 891 | 892 | 893 | 894 | 894 | 895 |
| 897 | 897 | 898 | 899 | 900 | 901 | 901 | 901 | 902 | 902 | 903 | 904 | 904 | 904 | 905 | 905 | 906 | 908 | 909 | 911 | 911 | 911 | 911 | 913 | 913 | 914 | 914 |
| 915 | 915 | 915 | 916 | 917 | 919 | 920 | 921 | 923 | 923 | 924 | 924 | 924 | 925 | 928 | 929 | 931 | 931 | 931 | 932 | 932 | 933 | 935 | 935 | 936 | 936 | 936 |
| 937 | 938 | 938 | 942 | 944 | 944 | 945 | 946 | 946 | 947 | 947 | 949 | 950 | 950 | 951 | 952 | 952 | 954 | 954 | 954 | 955 | 955 | 955 | 956 | 959 | 959 | 960 |
| 962 | 963 | 963 | 966 | 968 | 968 | 968 | 969 | 971 | 974 | 974 | 978 | 979 | 981 | 982 | 985 | 991 | 992 | 992 | 994 | 994 | 994 | 994 | 994 | 995 | 996 | 996 |

1000 random numbers sorted by using merge sorting algorithms.

Time taken for merge sort in seconds: 0.000000

Merge Sort 100,000 Words:

After execution the merge sort algorithm displays that the time taken to process 100,000 random numbers was 0.0600 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----|
| 46 | 99346 | 99347 | 99348 | 99348 | 99348 | 99350 | 99351 | 99351 | 99352 | 99352 | 99353 | 99354 | 99356 | 99358 | 99358 | 99359 | 99359 | 99360 | 99360 | 99362 | 99362 | 99363 | 99366 | 99366 | 99367 | 993 |
| 57 | 99368 | 99368 | 99369 | 99369 | 99370 | 99370 | 99371 | 99373 | 99374 | 99374 | 99376 | 99377 | 99378 | 99378 | 99379 | 99380 | 99380 | 99380 | 99384 | 99384 | 99385 | 99386 | 99387 | 99388 | 99388 | 993 |
| 69 | 99390 | 99390 | 99394 | 99394 | 99395 | 99396 | 99397 | 99398 | 99398 | 99398 | 99401 | 99401 | 99401 | 99402 | 99404 | 99404 | 99405 | 99406 | 99410 | 99411 | 99412 | 99412 | 99413 | 99414 | 99414 | 994 |
| 14 | 99415 | 99415 | 99415 | 99415 | 99416 | 99417 | 99417 | 99420 | 99420 | 99422 | 99423 | 99423 | 99423 | 99424 | 99424 | 99425 | 99425 | 99425 | 99427 | 99430 | 99430 | 99431 | 99431 | 99433 | 99435 | 994 |
| 37 | 99437 | 99440 | 99440 | 99442 | 99448 | 99448 | 99448 | 99449 | 99450 | 99451 | 99452 | 99452 | 99453 | 99454 | 99455 | 99455 | 99455 | 99456 | 99456 | 99458 | 99458 | 99459 | 99459 | 99460 | 99461 | 994 |
| 61 | 99461 | 99462 | 99464 | 99464 | 99465 | 99467 | 99468 | 99468 | 99468 | 99471 | 99471 | 99471 | 99476 | 99477 | 99479 | 99479 | 99482 | 99482 | 99482 | 99483 | 99484 | 99484 | 99487 | 99488 | 99488 | 994 |
| 88 | 99493 | 99493 | 99494 | 99495 | 99495 | 99496 | 99496 | 99497 | 99498 | 99498 | 99498 | 99498 | 99498 | 99501 | 99501 | 99502 | 99503 | 99503 | 99506 | 99507 | 99508 | 99511 | 99511 | 99512 | 99513 | 995 |
| 15 | 99517 | 99517 | 99519 | 99519 | 99520 | 99522 | 99523 | 99524 | 99524 | 99525 | 99525 | 99525 | 99526 | 99526 | 99526 | 99528 | 99529 | 99531 | 99531 | 99532 | 99533 | 99533 | 99535 | 99536 | 99537 | 995 |
| 38 | 99538 | 99540 | 99541 | 99544 | 99546 | 99549 | 99550 | 99551 | 99552 | 99554 | 99557 | 99558 | 99558 | 99560 | 99561 | 99562 | 99563 | 99564 | 99564 | 99566 | 99566 | 99568 | 99568 | 99569 | 99571 | 995 |
| 71 | 99571 | 99572 | 99572 | 99573 | 99573 | 99574 | 99575 | 99576 | 99576 | 99577 | 99578 | 99578 | 99579 | 99580 | 99580 | 99581 | 99583 | 99584 | 99585 | 99586 | 99587 | 99588 | 99588 | 99588 | 99589 | 995 |
| 90 | 99590 | 99592 | 99593 | 99593 | 99595 | 99596 | 99596 | 99597 | 99599 | 99601 | 99601 | 99603 | 99606 | 99607 | 99608 | 99610 | 99611 | 99612 | 99612 | 99612 | 99612 | 99615 | 99618 | 99618 | 99619 | 996 |
| 19 | 99622 | 99624 | 99624 | 99624 | 99626 | 99626 | 99628 | 99629 | 99629 | 99631 | 99632 | 99632 | 99636 | 99637 | 99639 | 99640 | 99643 | 99644 | 99646 | 99646 | 99646 | 99648 | 99649 | 99652 | 99652 | 996 |
| 54 | 99656 | 99663 | 99664 | 99664 | 99664 | 99668 | 99670 | 99670 | 99670 | 99671 | 99671 | 99672 | 99672 | 99673 | 99674 | 99674 | 99674 | 99676 | 99677 | 99678 | 99678 | 99679 | 99679 | 99680 | 99683 | 996 |
| 83 | 99684 | 99685 | 99685 | 99686 | 99688 | 99688 | 99689 | 99689 | 99689 | 99691 | 99693 | 99698 | 99698 | 99700 | 99700 | 99702 | 99703 | 99704 | 99706 | 99706 | 99706 | 99709 | 99709 | 99710 | 99711 | 997 |
| 12 | 99713 | 99715 | 99717 | 99717 | 99717 | 99718 | 99718 | 99721 | 99721 | 99722 | 99723 | 99723 | 99725 | 99725 | 99725 | 99726 | 99727 | 99730 | 99731 | 99731 | 99731 | 99732 | 99733 | 99734 | 99735 | 997 |
| 36 | 99739 | 99741 | 99744 | 99744 | 99746 | 99747 | 99747 | 99751 | 99752 | 99752 | 99753 | 99754 | 99754 | 99754 | 99754 | 99755 | 99757 | 99757 | 99757 | 99759 | 99759 | 99760 | 99763 | 99763 | 99764 | 997 |
| 66 | 99766 | 99770 | 99770 | 99771 | 99774 | 99774 | 99775 | 99779 | 99780 | 99780 | 99780 | 99781 | 99781 | 99785 | 99785 | 99786 | 99786 | 99786 | 99787 | 99789 | 99790 | 99790 | 99790 | 99791 | 99791 | 997 |
| 93 | 99794 | 99794 | 99795 | 99795 | 99796 | 99797 | 99797 | 99797 | 99799 | 99800 | 99800 | 99801 | 99802 | 99802 | 99803 | 99803 | 99805 | 99806 | 99807 | 99807 | 99808 | 99808 | 99809 | 99810 | 99810 | 998 |
| 11 | 99812 | 99812 | 99812 | 99814 | 99814 | 99815 | 99816 | 99817 | 99817 | 99818 | 99819 | 99820 | 99821 | 99822 | 99824 | 99825 | 99827 | 99829 | 99829 | 99833 | 99833 | 99835 | 99837 | 99839 | 99839 | 998 |
| 39 | 99841 | 99842 | 99843 | 99843 | 99843 | 99844 | 99844 | 99846 | 99847 | 99848 | 99850 | 99853 | 99855 | 99857 | 99859 | 99860 | 99861 | 99861 | 99861 | 99862 | 99863 | 99865 | 99867 | 99868 | 99869 | 998 |
| 71 | 99872 | 99874 | 99874 | 99874 | 99875 | 99876 | 99876 | 99877 | 99877 | 99878 | 99878 | 99880 | 99880 | 99882 | 99883 | 99883 | 99883 | 99884 | 99884 | 99885 | 99885 | 99886 | 99886 | 99887 | 99887 | 998 |
| 89 | 99890 | 99894 | 99895 | 99896 | 99896 | 99898 | 99899 | 99901 | 99901 | 99902 | 99903 | 99903 | 99903 | 99904 | 99906 | 99907 | 99910 | 99910 | 99911 | 99914 | 99914 | 99915 | 99915 | 99917 | 99917 | 999 |
| 17 | 99918 | 99918 | 99919 | 99919 | 99920 | 99922 | 99922 | 99923 | 99924 | 99925 | 99925 | 99925 | 99926 | 99928 | 99929 | 99930 | 99930 | 99931 | 99933 | 99933 | 99935 | 99936 | 99936 | 99937 | 99940 | 999 |
| 40 | 99942 | 99943 | 99944 | 99945 | 99945 | 99945 | 99945 | 99946 | 99946 | 99946 | 99946 | 99948 | 99949 | 99951 | 99952 | 99952 | 99954 | 99954 | 99955 | 99958 | 99958 | 99958 | 99958 | 99959 | 99959 | 999 |
| 60 | 99960 | 99960 | 99960 | 99962 | 99963 | 99964 | 99965 | 99965 | 99967 | 99968 | 99969 | 99969 | 99970 | 99971 | 99974 | 99974 | 99974 | 99978 | 99978 | 99978 | 99978 | 99979 | 99980 | 99980 | 99982 | 999 |
| 82 | 99985 | 99985 | 99986 | 99986 | 99987 | 99988 | 99992 | 99992 | 99992 | 99992 | 99992 | 99993 | 99993 | 99994 | 99995 | 99996 | 99997 | | | | | | | | | |

100000 random numbers sorted by using merge sorting algorithms.

Merge Sort 1000000 Words:

After execution the merge sort algorithm displays that the time taken to process 1,000,000 random numbers was 0.64000 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|
| 255 | 999256 | 999257 | 999258 | 999259 | 999260 | 999262 | 999261 | 999261 | 999263 | 999265 | 999265 | 999266 | 999266 | 999267 | 999268 | 999268 | 999271 | 999274 | 999274 | 999275 | 999275 | 999278 | 999279 | 999279 | 999279 | 999 | |
| 279 | 999280 | 999281 | 999282 | 999282 | 999283 | 999283 | 999284 | 999285 | 999286 | 999289 | 999289 | 999291 | 999291 | 999292 | 999293 | 999294 | 999295 | 999295 | 999295 | 999295 | 999296 | 999297 | 999297 | 999298 | 999298 | 999 | |
| 301 | 999302 | 999302 | 999303 | 999303 | 999304 | 999305 | 999305 | 999308 | 999310 | 999313 | 999315 | 999316 | 999320 | 999320 | 999320 | 999321 | 999322 | 999323 | 999325 | 999326 | 999327 | 999327 | 999328 | 999328 | 999330 | 999 | |
| 331 | 999331 | 999332 | 999333 | 999334 | 999334 | 999336 | 999337 | 999338 | 999338 | 999338 | 999339 | 999340 | 999341 | 999341 | 999341 | 999342 | 999343 | 999344 | 999345 | 999348 | 999349 | 999349 | 999352 | 999352 | 999355 | 999 | |
| 355 | 999356 | 999357 | 999357 | 999358 | 999358 | 999360 | 999361 | 999362 | 999366 | 999366 | 999367 | 999368 | 999369 | 999370 | 999373 | 999374 | 999375 | 999375 | 999378 | 999378 | 999379 | 999380 | 999380 | 999381 | 999381 | 999 | |
| 383 | 999386 | 999387 | 999390 | 999394 | 999395 | 999396 | 999396 | 999401 | 999401 | 999402 | 999402 | 999403 | 999403 | 999403 | 999406 | 999407 | 999408 | 999411 | 999411 | 999413 | 999413 | 999415 | 999418 | 999423 | 999425 | 999 | |
| 427 | 999428 | 999431 | 999431 | 999433 | 999434 | 999435 | 999435 | 999440 | 999437 | 999437 | 999438 | 999439 | 999443 | 999444 | 999445 | 999445 | 999446 | 999448 | 999448 | 999448 | 999448 | 999449 | 999450 | 999450 | 999450 | 999 | |
| 450 | 999451 | 999451 | 999451 | 999452 | 999452 | 999453 | 999453 | 999454 | 999454 | 999455 | 999455 | 999456 | 999456 | 999457 | 999458 | 999458 | 999458 | 999459 | 999459 | 999460 | 999460 | 999460 | 999461 | 999461 | 999461 | 999 | |
| 463 | 999464 | 999466 | 999468 | 999468 | 999470 | 999476 | 999471 | 999472 | 999473 | 999473 | 999474 | 999475 | 999477 | 999478 | 999479 | 999480 | 999482 | 999484 | 999485 | 999486 | 999486 | 999487 | 999487 | 999488 | 999489 | 999 | |
| 492 | 999493 | 999494 | 999495 | 999496 | 999500 | 999500 | 999501 | 999502 | 999502 | 999504 | 999505 | 999505 | 999505 | 999506 | 999507 | 999507 | 999509 | 999509 | 999509 | 999510 | 999511 | 999512 | 999512 | 999513 | 999513 | 999 | |
| 513 | 999515 | 999515 | 999517 | 999518 | 999518 | 999518 | 999519 | 999520 | 999520 | 999522 | 999524 | 999525 | 999526 | 999526 | 999526 | 999527 | 999528 | 999528 | 999530 | 999532 | 999533 | 999534 | 999535 | 999535 | 999537 | 999 | |
| 537 | 999538 | 999539 | 999544 | 999544 | 999545 | 999547 | 999548 | 999548 | 999549 | 999550 | 999551 | 999551 | 999552 | 999553 | 999556 | 999557 | 999558 | 999560 | 999561 | 999564 | 999565 | 999569 | 999569 | 999571 | 999572 | 999 | |
| 572 | 999574 | 999574 | 999576 | 999578 | 999578 | 999582 | 999583 | 999585 | 999586 | 999587 | 999588 | 999588 | 999595 | 999601 | 999601 | 999601 | 999602 | 999602 | 999606 | 999606 | 999606 | 999609 | 999610 | 999613 | 999614 | 999618 | 999 |
| 618 | 999618 | 999620 | 999620 | 999620 | 999621 | 999621 | 999622 | 999624 | 999625 | 999625 | 999625 | 999626 | 999627 | 999627 | 999628 | 999628 | 999629 | 999629 | 999629 | 999630 | 999630 | 999631 | 999631 | 999632 | 999632 | 999 | |
| 632 | 999633 | 999635 | 999635 | 999635 | 999636 | 999637 | 999637 | 999638 | 999641 | 999642 | 999642 | 999642 | 999642 | 999646 | 999647 | 999647 | 999647 | 999648 | 999648 | 999649 | 999649 | 999650 | 999651 | 999651 | 999651 | 999 | |
| 652 | 999654 | 999654 | 999654 | 999656 | 999656 | 999656 | 999657 | 999659 | 999661 | 999663 | 999664 | 999664 | 999664 | 999666 | 999667 | 999667 | 999669 | 999670 | 999671 | 999672 | 999676 | 999677 | 999678 | 999678 | 999679 | 999 | |
| 679 | 999683 | 999685 | 999685 | 999686 | 999686 | 999687 | 999687 | 999688 | 999690 | 999690 | 999691 | 999691 | 999693 | 999695 | 999696 | 999697 | 999698 | 999699 | 999699 | 999700 | 999701 | 999702 | 999705 | 999706 | 999706 | 999 | |
| 707 | 999710 | 999711 | 999711 | 999714 | 999715 | 999717 | 999717 | 999718 | 999719 | 999721 | 999724 | 999724 | 999728 | 999728 | 999728 | 999728 | 999731 | 999732 | 999734 | 999736 | 999739 | 999741 | 999741 | 999742 | 999742 | 999 | |
| 744 | 999745 | 999745 | 999746 | 999747 | 999749 | 999750 | 999750 | 999750 | 999751 | 999752 | 999752 | 999753 | 999753 | 999753 | 999756 | 999756 | 999761 | 999763 | 999763 | 999763 | 999763 | 999765 | 999766 | 999768 | 999768 | 999 | |
| 775 | 999777 | 999777 | 999778 | 999780 | 999781 | 999783 | 999787 | 999789 | 999791 | 999792 | 999793 | 999794 | 999795 | 999795 | 999796 | 999796 | 999797 | 999797 | 999797 | 999797 | 999797 | 999799 | 999800 | 999800 | 999800 | 999 | |
| 803 | 999804 | 999804 | 999804 | 999804 | 999806 | 999806 | 999807 | 999808 | 999810 | 999812 | 999813 | 999814 | 999815 | 999816 | 999817 | 999817 | 999818 | 999822 | 999824 | 999825 | 999825 | 999825 | 999825 | 999825 | 999825 | 999 | |
| 826 | 999827 | 999829 | 999829 | 999830 | 999831 | 999831 | 999831 | 999831 | 999833 | 999833 | 999834 | 999835 | 999836 | 999836 | 999836 | 999837 | 999840 | 999846 | 999842 | 999842 | 999843 | 999845 | 999847 | 999848 | 999848 | 999 | |
| 851 | 999852 | 999853 | 999855 | 999862 | 999864 | 999865 | 999866 | 999866 | 999867 | 999870 | 999870 | 999872 | 999874 | 999874 | 999875 | 999875 | 999877 | 999877 | 999878 | 999878 | 999881 | 999882 | 999887 | 999889 | 999892 | 999 | |
| 892 | 999893 | 999893 | 999895 | 999898 | 999898 | 999898 | 999900 | 999900 | 999902 | 999902 | 999903 | 999904 | 999906 | 999906 | 999906 | 999907 | 999907 | 999908 | 999908 | 999909 | 999909 | 999910 | 999910 | 999911 | 999911 | 999 | |
| 912 | 999915 | 999915 | 999915 | 999916 | 999917 | 999917 | 999917 | 999918 | 999919 | 999920 | 999920 | 999923 | 999924 | 999924 | 999927 | 999928 | 999929 | 999932 | 999932 | 999935 | 999936 | 999936 | 999937 | 999937 | 999938 | 999 | |
| 939 | 999939 | 999944 | 999944 | 999944 | 999945 | 999945 | 999947 | 999948 | 999948 | 999948 | 999948 | 999949 | 999954 | 999955 | 999957 | 999957 | 999961 | 999962 | 999962 | 999963 | 999963 | 999963 | 999963 | 999966 | 999968 | 999 | |
| 969 | 999970 | 999970 | 999971 | 999971 | 999972 | 999973 | 999975 | 999975 | 999976 | 999976 | 999977 | 999978 | 999980 | 999981 | 999983 | 999983 | 999985 | 999985 | 999986 | 999987 | 999989 | 999990 | 999991 | 999993 | 999994 | 999 | |
| 984 | 999997 | 999998 | 999998 | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000000 random numbers sorted by using merge sorting algorithms. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Time taken for merge sort in seconds: 0.640000 | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Merge Sort 2000000 Words:

After execution the merge sort algorithm displays that the time taken to process 2,000,000 random numbers was 1.29000 seconds.

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------|--|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|--|
| 9429 | 1999423 | 1999423 | 1999423 | 1999424 | 1999426 | 1999426 | 1999426 | 1999429 | 1999429 | 1999430 | 1999430 | 1999431 | 1999433 | 1999435 | 1999435 | 1999437 | 1999438 | 1999439 | 1999442 | 1999442 | 1999444 | 1999446 | 1999446 | 1999448 | 1999449 | |
| 9449 | 1999449 | 1999450 | 1999450 | 1999455 | 1999456 | 1999458 | 1999458 | 1999459 | 1999460 | 1999462 | 1999465 | 1999466 | 1999466 | 1999466 | 1999466 | 1999468 | 1999468 | 1999468 | 1999469 | 1999469 | 1999471 | 1999472 | 1999472 | 1999472 | 1999472 | |
| 9473 | 1999474 | 1999475 | 1999475 | 1999476 | 1999476 | 1999478 | 1999479 | 1999482 | 1999482 | 1999483 | 1999484 | 1999486 | 1999486 | 1999487 | 1999487 | 1999489 | 1999489 | 1999490 | 1999491 | 1999492 | 1999492 | 1999493 | 1999494 | 1999495 | 1999495 | |
| 9497 | 1999497 | 1999499 | 1999501 | 1999502 | 1999502 | 1999503 | 1999503 | 1999503 | 1999504 | 1999511 | 1999512 | 1999513 | 1999513 | 1999515 | 1999515 | 1999517 | 1999517 | 1999517 | 1999521 | 1999521 | 1999522 | 1999522 | 1999523 | 1999525 | 1999525 | |
| 9527 | 1999527 | 1999528 | 1999529 | 1999530 | 1999536 | 1999538 | 1999535 | 1999536 | 1999540 | 1999548 | 1999542 | 1999543 | 1999544 | 1999545 | 1999546 | 1999548 | 1999548 | 1999550 | 1999551 | 1999552 | 1999552 | 1999553 | 1999553 | 1999553 | 1999553 | |
| 9554 | 1999555 | 1999555 | 1999556 | 1999556 | 1999557 | 1999558 | 1999559 | 1999559 | 1999560 | 1999560 | 1999563 | 1999564 | 1999566 | 1999569 | 1999570 | 1999571 | 1999572 | 1999573 | 1999573 | 1999573 | 1999575 | 1999576 | 1999577 | 1999579 | 1999579 | |
| 9577 | 1999578 | 1999578 | 1999580 | 1999581 | 1999582 | 1999586 | 1999587 | 1999587 | 1999588 | 1999588 | 1999590 | 1999591 | 1999591 | 1999592 | 1999593 | 1999593 | 1999593 | 1999593 | 1999594 | 1999594 | 1999597 | 1999597 | 1999597 | 1999602 | 1999602 | |
| 9603 | 1999605 | 1999608 | 1999612 | 1999612 | 1999613 | 1999613 | 1999613 | 1999615 | 1999616 | 1999619 | 1999619 | 1999620 | 1999623 | 1999623 | 1999623 | 1999626 | 1999628 | 1999628 | 1999628 | 1999628 | 1999631 | 1999632 | 1999632 | 1999633 | 1999633 | |
| 9635 | 1999635 | 1999636 | 1999638 | 1999640 | 1999641 | 1999641 | 1999641 | 1999642 | 1999644 | 1999645 | 1999645 | 1999646 | 1999647 | 1999647 | 1999648 | 1999649 | 1999650 | 1999651 | 1999651 | 1999652 | 1999652 | 1999654 | 1999656 | 1999656 | 1999656 | |
| 9657 | 1999657 | 1999661 | 1999662 | 1999662 | 1999663 | 1999663 | 1999663 | 1999666 | 1999669 | 1999671 | 1999672 | 1999675 | 1999679 | 1999681 | 1999682 | 1999684 | 1999684 | 1999684 | 1999684 | 1999684 | 1999687 | 1999687 | 1999688 | 1999688 | 1999688 | |
| 9689 | 1999691 | 1999692 | 1999692 | 1999694 | 1999695 | 1999695 | 1999696 | 1999696 | 1999696 | 1999697 | 1999697 | 1999697 | 1999698 | 1999698 | 1999698 | 1999699 | 1999699 | 1999699 | 1999700 | 1999701 | 1999705 | 1999705 | 1999706 | 1999706 | 1999706 | |
| 9706 | 1999706 | 1999707 | 1999711 | 1999714 | 1999715 | 1999716 | 1999717 | 1999717 | 1999717 | 1999721 | 1999721 | 1999721 | 1999722 | 1999722 | 1999723 | 1999723 | 1999724 | 1999725 | 1999727 | 1999727 | 1999731 | 1999731 | 1999732 | 1999734 | 1999735 | |
| 9737 | 1999738 | 1999738 | 1999738 | 1999738 | 1999738 | 1999740 | 1999743 | 1999745 | 1999745 | 1999745 | 1999746 | 1999746 | 1999748 | 1999748 | 1999748 | 1999749 | 1999750 | 1999751 | 1999752 | 1999752 | 1999754 | 1999755 | 1999756 | 1999758 | 1999758 | |
| 9758 | 1999759 | 1999761 | 1999763 | 1999764 | 1999765 | 1999765 | 1999766 | 1999766 | 1999767 | 1999767 | 1999768 | 1999769 | 1999770 | 1999771 | 1999771 | 1999773 | 1999774 | 1999774 | 1999774 | 1999776 | 1999777 | 1999778 | 1999780 | 1999780 | 1999780 | |
| 9781 | 1999781 | 1999782 | 1999782 | 1999783 | 1999784 | 1999785 | 1999786 | 1999786 | 1999787 | 1999788 | 1999788 | 1999789 | 1999789 | 1999791 | 1999794 | 1999794 | 1999794 | 1999794 | 1999795 | 1999795 | 1999796 | 1999796 | 1999797 | 1999797 | 1999797 | |
| 9798 | 1999799 | 1999799 | 1999800 | 1999801 | 1999801 | 1999802 | 1999803 | 1999803 | 1999803 | 1999806 | 1999808 | 1999808 | 1999809 | 1999809 | 1999810 | 1999811 | 1999813 | 1999814 | 1999815 | 1999816 | 1999817 | 1999817 | 1999818 | 1999818 | 1999818 | |
| 9819 | 1999819 | 1999820 | 1999820 | 1999821 | 1999821 | 1999823 | 1999823 | 1999823 | 1999823 | 1999826 | 1999826 | 1999826 | 1999826 | 1999827 | 1999827 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | 1999828 | |
| 9850 | 1999852 | 1999853 | 1999853 | 1999855 | 1999855 | 1999856 | 1999856 | 1999856 | 1999856 | 1999862 | 1999863 | 1999864 | 1999865 | 1999865 | 1999866 | 1999866 | 1999866 | 1999867 | 1999867 | 1999867 | 1999871 | 1999871 | 1999874 | 1999878 | 1999880 | |
| 9880 | 1999886 | 1999886 | 1999887 | 1999887 | 1999888 | 1999888 | 1999888 | 1999889 | 1999889 | 1999890 | 1999891 | 1999891 | 1999892 | 1999894 | 1999894 | 1999896 | 1999896 | 1999896 | 1999896 | 1999898 | 1999898 | 1999900 | 1999900 | 1999901 | 1999901 | |
| 9902 | 1999907 | 1999910 | 1999911 | 1999912 | 1999913 | 1999914 | 1999916 | 1999917 | 1999918 | 1999919 | 1999922 | 1999922 | 1999922 | 1999923 | 1999924 | 1999927 | 1999927 | 1999928 | 1999929 | 1999929 | 1999931 | 1999935 | 1999935 | 1999940 | 1999941 | |
| 9941 | 1999942 | 1999944 | 1999945 | 1999947 | 1999947 | 1999947 | 1999947 | 1999950 | 1999950 | 1999951 | 1999951 | 1999951 | 1999951 | 1999953 | 1999953 | 1999953 | 1999955 | 1999955 | 1999955 | 1999958 | 1999968 | 1999970 | 1999972 | 1999972 | 1999974 | |
| 9956 | 1999975 | 1999976 | 1999977 | 1999980 | 1999980 | 1999980 | 1999983 | 1999983 | 1999984 | 1999984 | 1999985 | 1999986 | 1999988 | 1999988 | 1999989 | 1999991 | 1999991 | 1999993 | 1999993 | 1999993 | 1999997 | 1999997 | 1999997 | 1999997 | 1999997 | |
| 2000000 | random numbers sorted by using merge sorting algorithms. | | | | | | | | | | | | | | | | | | | | | | | | | |

Time taken for merge sort in seconds: 1.290880

The Testing phases on the Gnuplot were plotted as such:

GNU PLOT TEST

Run the program for the 10000 and 50000 random numbers

In graph, x axis is Length of the numbers and y axis is Time in seconds.

First I run the program for 10000 unsorted numbers and sorted by all sorting types i.e quick, merge, selection and bubble sorting

Time taken by quick sorting – 0.0000 seconds

Time taken by merge sorting – 0.0100 seconds

Time taken by selection sorting – 0.1200 seconds

Time taken by bubble sorting – 0.0300 seconds

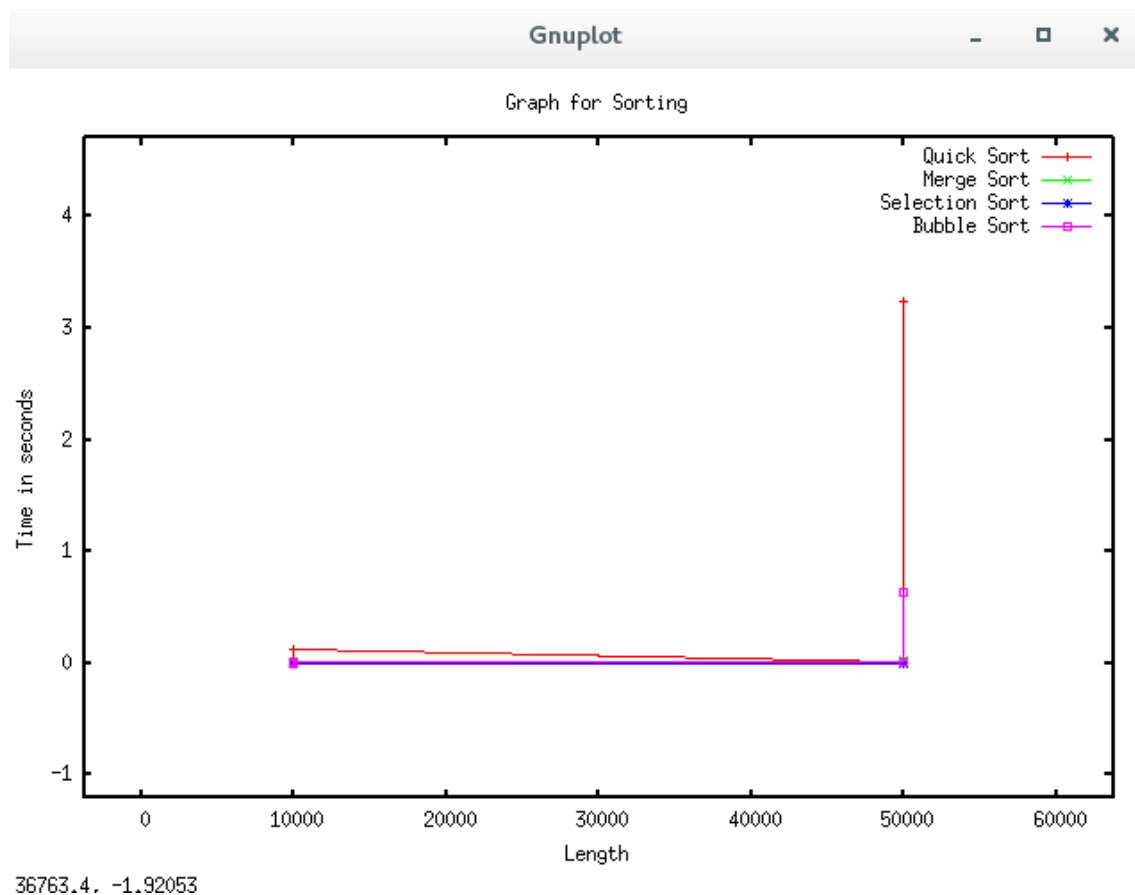
And then for 50000 unsorted numbers.

Time taken by quick sorting – 0.0200 seconds

Time taken by merge sorting – 0.0300 seconds

Time taken by selection sorting – 3.1600 seconds

Time taken by bubble sorting – 0.6200 seconds



Run the program for the 20000, 50000, 100000, 500000, 1000000, 10000000 random numbers

In graph, x axis is Length of the numbers and y axis is Time in seconds.

Time taken by quick sorting to sort 20000 numbers – 0.0100 seconds

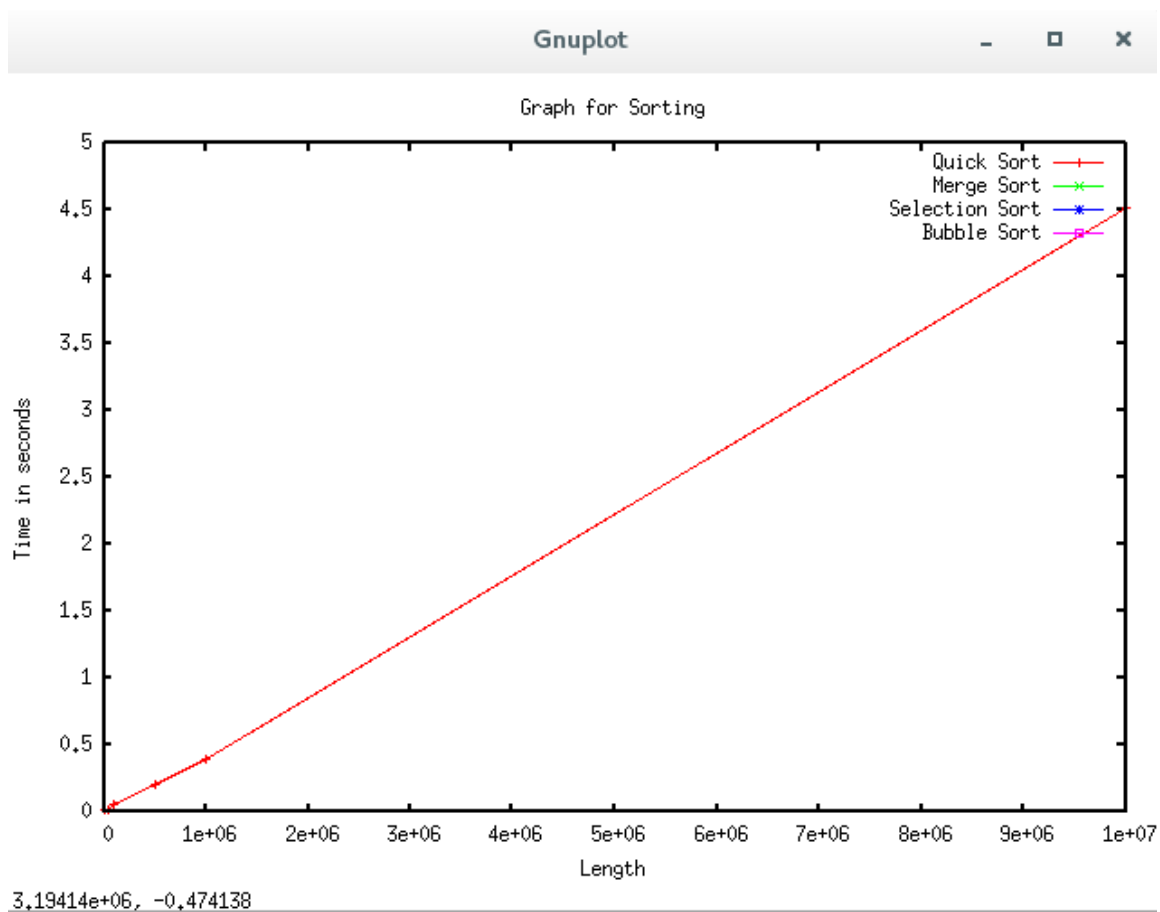
Time taken by quick sorting to sort 50000 numbers – 0.0100 seconds

Time taken by quick sorting to sort 100000 numbers – 0.0500 seconds

Time taken by quick sorting to sort 500000 numbers – 0.2200 seconds

Time taken by quick sorting to sort 1000000 numbers – 0.3900 seconds

Time taken by quick sorting to sort 10000000 numbers – 3.9900 seconds



graph.out

```
1 10000 0.010000 0.000000 0.000000 0.000000
2 50000 0.020000 0.000000 0.000000 0.000000
3 100000 0.050000 0.000000 0.000000 0.000000
4 500000 0.200000 0.000000 0.000000 0.000000
5 1000000 0.390000 0.000000 0.000000 0.000000
6 10000000 4.520000 0.000000 0.000000 0.000000
7
```

Run the program for the 20000, 50000, 1000000 and 2000000 random numbers

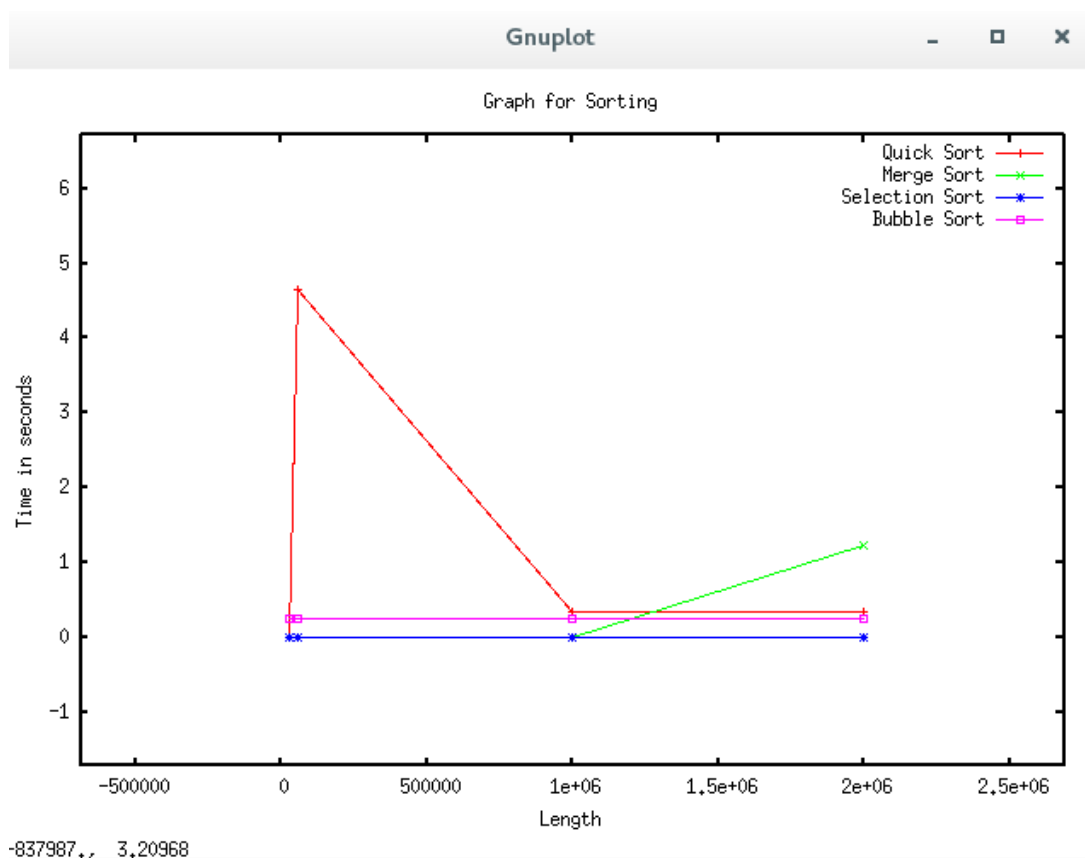
In graph, x axis is Length of the numbers and y axis is Time in seconds.

Time taken by bubble sorting to sort 30000 numbers – 0.24000 seconds

Time taken by selection sorting to sort 50000 numbers – 4.64000 seconds

Time taken by quick sorting to sort 1000000 numbers – 0.33000 seconds

Time taken by merge sorting to sort 2000000 numbers – 1.23000 seconds



This is output file of gnuplot i.e **graph.out**. As displayed in the file below, the figures show that when I ran the program to sort out the 30000 by bubble sorting method, it also included 0.0000 second for the quick, merge and selection for 30000 numbers. Same for all other numbers.

```
1 30000 0.000000 0.000000 0.000000 0.240000
2 60000 4.640000 0.000000 0.000000 0.240000
3 1000000 0.330000 0.000000 0.000000 0.240000
4 2000000 0.330000 1.230000 0.000000 0.240000
5
```

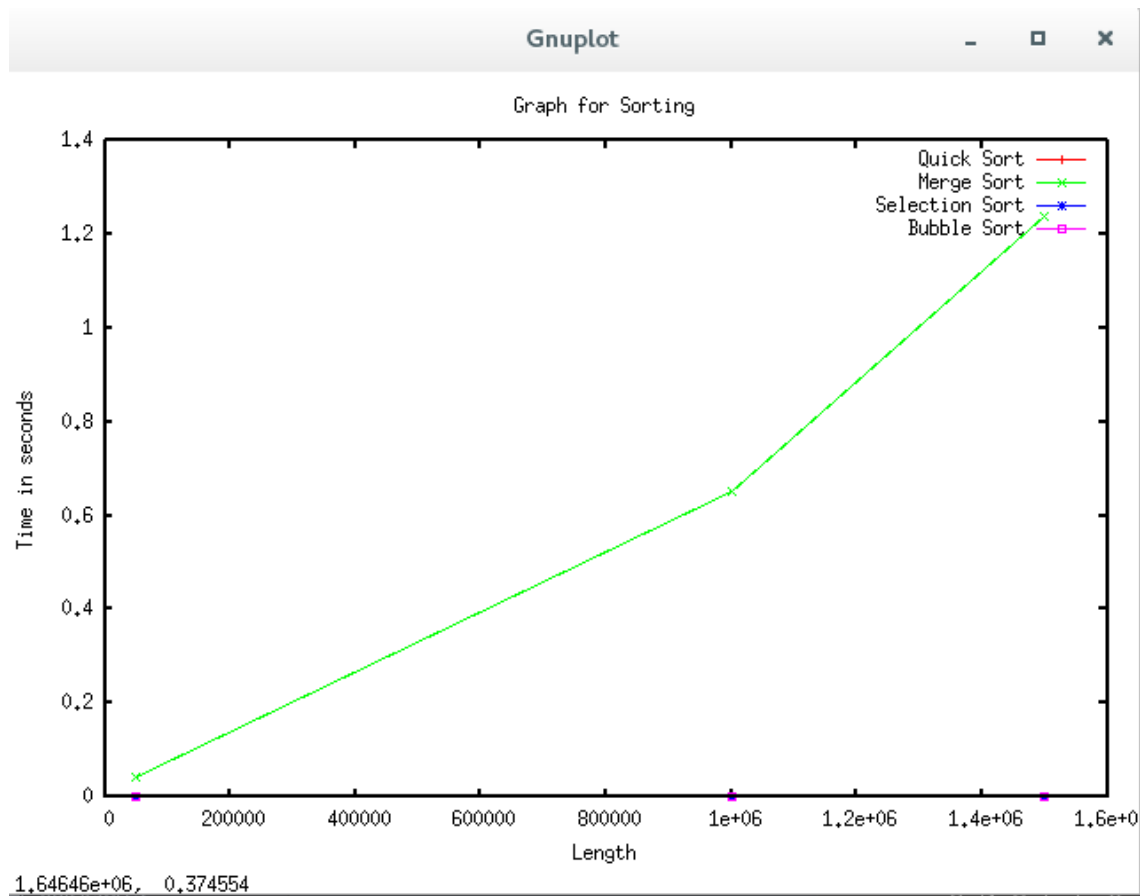

Run the program for the 50000, 1000000 and 1500000 random numbers

In graph, x axis is Length of the numbers and y axis is Time in seconds.

Time taken by merge sorting to sort 50000 numbers – 0.04000 seconds

Time taken by merge sorting to sort 1000000 numbers – 0.65000 seconds

Time taken by merge sorting to sort 1500000 numbers – 1.24000 seconds



graph.out

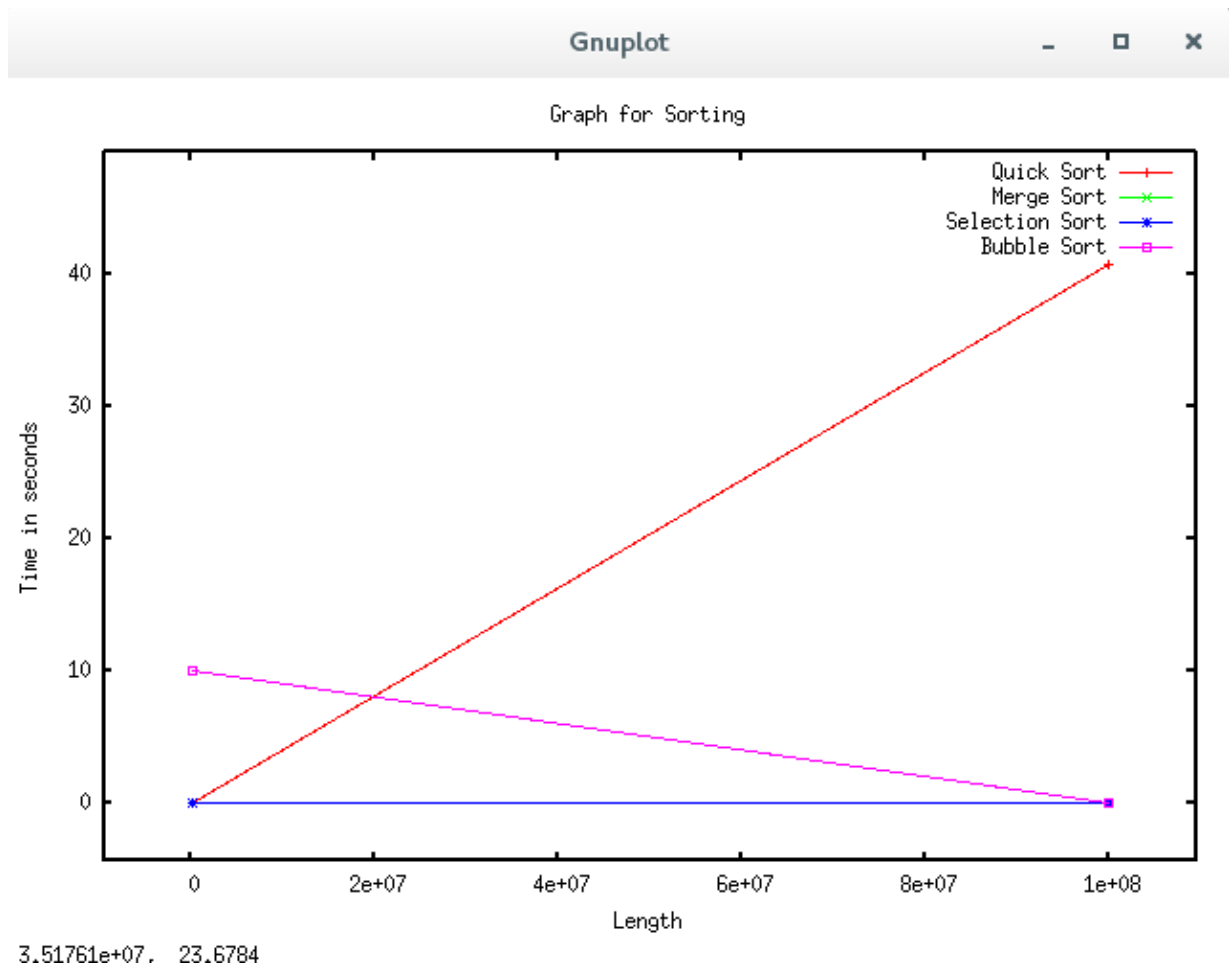
```
1 50000 0.000000 0.040000 0.000000 0.000000
2 1000000 0.000000 0.650000 0.000000 0.000000
3 1500000 0.000000 1.240000 0.000000 0.000000
4
```

Run the program for the 100000000 random numbers

In graph, x axis is Length of the numbers and y axis is Time in seconds.

Time taken by quick sorting to sort 100000000 numbers – 40.77000 seconds

Time taken by bubble sorting to sort 1000000 numbers – 10.12000 seconds



graph.out

```
1 100000000 40.770000 0.000000 0.000000 0.000000
2 200000 0.000000 0.000000 0.000000 10.120000
3 |
```

Testing valgrind debugging tool while running the code :

valgrind debugging tool: valgrind is a multipurpose code profiling and memory debugging tool for Linux. It checks and find memory leaks

- I added this tool to my program as I incurred errors in the process of testing my code, as there were memory leaks. therefore I added 'free()' command in my program to free up all used memory at the end so such error was avoided.

```
sc15hv@comp-pc3023:~/Desktop/yoyoy/documents-export-2016-05-04/new
File Edit View Search Terminal Help
[sc15hv@comp-pc3023 new]$ valgrind ./sortingAlgorithms
==16031== Memcheck, a memory error detector
==16031== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==16031== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==16031== Command: ./sortingAlgorithms
==16031==
*****
*      Benchmarking Algorithms      *
*      Harshit Verma                *
*      ID: 200978548                *
*****
Enter the random numbrs: 
```

As shown above; I have run the all different types of sorting functions and there is no memory leaking errors in my program

```
sc15hv@comp-pc3023:~/Desktop/yoyoy/documents-export-2016-05-04/new
File Edit View Search Terminal Help
Time taken for bubble sort in seconds: 0.000000
Choose the sorting type:
1. Quick sort
2. Merge sort
3. Selection sort
4. Bubble sort
5. Exit
6. For another Number of Elements
5
Option: 5
Exit!!!!!!!!!!
==16407==
==16407== HEAP SUMMARY:
==16407==    in use at exit: 0 bytes in 0 blocks
==16407==   total heap usage: 5 allocs, 5 frees, 16,568 bytes allocated
==16407==
==16407== All heap blocks were freed -- no leaks are possible
==16407==
==16407== For counts of detected and suppressed errors, rerun with: -v
==16407== ERROR SUMMARY: 62 errors from 25 contexts (suppressed: 1 from 1)
[sc15hv@comp-pc3023 new]$ 
```

Testing some other functions :

| S.NO | TESTING THE FUNCTIONS | EXPECTED | ACTUAL |
|------|---|---|--|
| 1. | rand() function | It should generates the random numbers. | This function generated random numbers succesfully |
| 2. | swapForBubbleSort(), swapforQuickSort(), swapforMergeSort(), swaforSelecionSort() | It should swap the two different numbers in the elements | These functions swap the numbers according to numeric order. |
| 3. | printBubbleSorting(), printquickSorting(), printselectionSorting(), printMergeSorting(). | It should print out the sorted list. | These functions printing out the sorted list |
| 4. | bubbleSorting() | It is the main function which should implement the bubble sorting | This function sort out the unsorted list by using bubble sorting method. |
| 5. | merge() | It is the main function which should implement the merge sorting | This function sort out the sorted list by using merge sorting method |
| 6. | quickSorting() | It is the main function which should implement the quick sorting | This function sort out the sorted list by using merge sorting method |
| 7. | SelectionSort() | It is the main function which implement the selection sort | This function sort out the sorted list by using merge selection sort |
| 8. | testBubbleSorting() testQuickSorting() testMergeSorting() testSelectionSorting() | These functions are calling the print function and main implement function to complete the sortings | These functions calls the print and main implement function properly which is called inside main file. |
| 9. | time | It should display the time taken for algorithm to sort out the unsorted random numbers | It shows the time taken by the different sorting to sort out. <u>And I have added the tests screen shots of every sorting above for different random numbers</u> |
| 10. | graph.gnu | This file should display the result on gnuplot as graph where x is length and y is time | It's plotting the point on graph. <u>And I have also added the tests and screen shots of gnuplot graphs.</u> |

CONCLUSION

In this conclusion I will outline any disadvantages to my code and any potential improvements I would recommend to myself and what benefits these will have on the program.

The first recommendation is to add more OPCOUNTS as this would to count the operational functions of the code and it also make the program execute faster.

I would also add more comments at the start of each section rather than just a short line.

This shows that the quick sort method remains the fastest above all and this method has been improved due to the efficiency of my code using the methods I have.
