Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Merge Sort |  |  |  |  |  |
| n = 10 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2700 | 3700 | 5100 | 6000 | 7100 |
|  | 1400 | 4800 | 4700 | 4400 | 5200 |
|  | 1100 | 2500 | 6900 | 4600 | 6100 |
|  | 900 | 2400 | 3700 | 9100 | 4300 |
|  | 1000 | 1900 | 2700 | 4000 | 5200 |
|  | 1000 | 2300 | 2500 | 4000 | 6800 |
|  | 900 | 2600 | 2700 | 3800 | 5800 |
|  | 1000 | 3700 | 3200 | 4500 | 6800 |
|  | 900 | 2500 | 4300 | 7500 | 6800 |
|  | 900 | 2600 | 3000 | 4700 | 6300 |
|  | 900 | 2900 | 4000 | 5400 | 4000 |
|  | 900 | 4600 | 5900 | 4700 | 6700 |
|  | 900 | 1700 | 2800 | 3500 | 4500 |
|  | 900 | 2600 | 6800 | 5000 | 11400 |
|  | 900 | 2600 | 3100 | 4500 | 7100 |
|  | 1000 | 4100 | 3700 | 5300 | 6800 |
|  | 900 | 4600 | 4400 | 3800 | 6800 |
|  | 900 | 2800 | 4400 | 18600 | 16900 |
|  | 2000 | 2000 | 3700 | 5700 | 6700 |
|  | 1100 | 2600 | 4000 | 22300 | 6700 |
| average | 1110 | 2975 | 4080 | 6570 | 6900 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Merge Sort |  |  |  |  |  |
| n = 50 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2700 | 8300 | 13900 | 19300 | 23300 |
|  | 4900 | 5800 | 14400 | 17400 | 19700 |
|  | 1300 | 12300 | 18800 | 21300 | 27300 |
|  | 1000 | 5300 | 12400 | 23500 | 21300 |
|  | 1000 | 5200 | 14100 | 18500 | 19700 |
|  | 900 | 8300 | 22300 | 29200 | 32300 |
|  | 1000 | 6600 | 15100 | 28800 | 18300 |
|  | 2000 | 5600 | 24000 | 26100 | 17300 |
|  | 1100 | 9900 | 9600 | 20100 | 17100 |
|  | 1000 | 5300 | 10600 | 10100 | 18900 |
|  | 900 | 16200 | 20200 | 5800 | 15700 |
|  | 1000 | 11300 | 10500 | 7000 | 6800 |
|  | 2500 | 19700 | 10800 | 2800 | 3600 |
|  | 1400 | 6400 | 5200 | 4600 | 2400 |
|  | 1000 | 4700 | 3100 | 4900 | 4700 |
|  | 1000 | 4600 | 1400 | 5100 | 3500 |
|  | 1000 | 5900 | 4000 | 4700 | 2300 |
|  | 2100 | 4800 | 4700 | 4500 | 3400 |
|  | 2100 | 5500 | 1700 | 3600 | 3400 |
|  | 1900 | 6900 | 3400 | 2100 | 2200 |
| average | 1590 | 7930 | 11010 | 12970 | 13160 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Merge Sort |  |  |  |  |  |
| n = 100 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2700 | 13700 | 28400 | 39200 | 44100 |
|  | 1400 | 14300 | 25000 | 46200 | 56100 |
|  | 2000 | 12400 | 32400 | 48200 | 51800 |
|  | 1100 | 12200 | 34700 | 44400 | 37800 |
|  | 1000 | 27400 | 32100 | 32200 | 18800 |
|  | 1700 | 21900 | 21900 | 34800 | 10100 |
|  | 1000 | 15300 | 20400 | 9900 | 11100 |
|  | 1100 | 25600 | 9500 | 6900 | 11600 |
|  | 1000 | 10300 | 4100 | 12200 | 7800 |
|  | 900 | 10900 | 4000 | 7800 | 9800 |
|  | 900 | 20700 | 4000 | 9500 | 6100 |
|  | 1000 | 13800 | 3000 | 5600 | 11300 |
|  | 1000 | 12100 | 4300 | 5500 | 7900 |
|  | 1000 | 5800 | 5700 | 7100 | 14700 |
|  | 1000 | 3700 | 4700 | 7600 | 7600 |
|  | 1000 | 1400 | 2800 | 7600 | 9900 |
|  | 900 | 2400 | 5900 | 9200 | 38500 |
|  | 900 | 4800 | 4500 | 5700 | 9500 |
|  | 900 | 1500 | 6000 | 7100 | 7900 |
|  | 1200 | 2700 | 3000 | 22600 | 11900 |
| average | 1185 | 11645 | 12820 | 18465 | 19215 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Merge Sort |  |  |  |  |  |
| n = 500 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2800 | 63000 | 163100 | 228100 | 264800 |
|  | 1800 | 77400 | 33400 | 54600 | 64300 |
|  | 1700 | 63900 | 23500 | 37400 | 110600 |
|  | 1100 | 15900 | 18800 | 52000 | 56500 |
|  | 900 | 11400 | 30400 | 43400 | 86000 |
|  | 900 | 19700 | 20100 | 36200 | 63100 |
|  | 900 | 13500 | 27600 | 38600 | 89100 |
|  | 1000 | 10300 | 19500 | 60800 | 50400 |
|  | 900 | 11900 | 26600 | 37300 | 55700 |
|  | 900 | 22900 | 24800 | 35000 | 76900 |
|  | 1000 | 13200 | 33700 | 32700 | 60400 |
|  | 900 | 14300 | 19500 | 51800 | 60200 |
|  | 1000 | 12800 | 21100 | 52000 | 51600 |
|  | 900 | 12800 | 18800 | 34000 | 59800 |
|  | 900 | 11900 | 28900 | 33000 | 83900 |
|  | 1900 | 12500 | 22500 | 35600 | 86700 |
|  | 1100 | 12000 | 33100 | 33800 | 84200 |
|  | 1100 | 14200 | 52600 | 36700 | 89600 |
|  | 1100 | 12000 | 33700 | 38300 | 86900 |
|  | 1300 | 11700 | 23100 | 41400 | 72500 |
| average | 1205 | 21865 | 33740 | 50635 | 82660 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Merge Sort |  |  |  |  |  |
| n = 1000 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2900 | 133900 | 277400 | 413600 | 661800 |
|  | 1700 | 66700 | 49400 | 98500 | 215000 |
|  | 2000 | 25100 | 59500 | 95100 | 116400 |
|  | 1300 | 19500 | 46000 | 86300 | 105800 |
|  | 1000 | 51400 | 45100 | 65900 | 119100 |
|  | 1000 | 23000 | 68700 | 90700 | 58400 |
|  | 900 | 22900 | 45100 | 71200 | 42600 |
|  | 900 | 22500 | 46300 | 70600 | 45700 |
|  | 900 | 23000 | 63400 | 70700 | 50700 |
|  | 900 | 22800 | 41100 | 86200 | 39300 |
|  | 900 | 30300 | 43400 | 64400 | 38800 |
|  | 900 | 21900 | 44400 | 69900 | 41000 |
|  | 900 | 22700 | 45200 | 69000 | 49500 |
|  | 900 | 22400 | 42600 | 77000 | 42100 |
|  | 1000 | 21500 | 44600 | 72700 | 213800 |
|  | 1000 | 26400 | 40800 | 68900 | 38000 |
|  | 1000 | 21500 | 52900 | 71200 | 36300 |
|  | 900 | 32100 | 54600 | 66700 | 35600 |
|  | 1300 | 20100 | 45300 | 71200 | 45100 |
|  | 1900 | 21600 | 79400 | 79200 | 41700 |
| average | 1210 | 32565 | 61760 | 92950 | 101835 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mmRule |  |  |  |  |  |
| n = 10 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 263200 | 227000 | 270500 | 277900 | 228300 |
|  | 6600 | 5800 | 13300 | 8400 | 6500 |
|  | 7500 | 5100 | 8400 | 9900 | 5800 |
|  | 5800 | 4700 | 6700 | 10000 | 5800 |
|  | 5300 | 4600 | 6300 | 8200 | 5700 |
|  | 5300 | 4800 | 6200 | 8700 | 5800 |
|  | 5300 | 4600 | 6300 | 16700 | 5700 |
|  | 5300 | 6800 | 8800 | 9500 | 5700 |
|  | 5200 | 6600 | 8400 | 8600 | 7200 |
|  | 5300 | 16100 | 6700 | 8100 | 7200 |
|  | 5200 | 7100 | 6400 | 11300 | 7100 |
|  | 5300 | 5400 | 10800 | 8800 | 13600 |
|  | 5700 | 7500 | 8300 | 10900 | 6800 |
|  | 5800 | 6600 | 6500 | 8500 | 8200 |
|  | 5400 | 5200 | 9700 | 7200 | 7800 |
|  | 5300 | 6800 | 8200 | 9000 | 5900 |
|  | 5200 | 6700 | 6500 | 11200 | 7900 |
|  | 7400 | 5500 | 6200 | 8600 | 8900 |
|  | 5900 | 6500 | 10800 | 18500 | 8100 |
|  | 5300 | 5400 | 9300 | 8500 | 6200 |
| average | 18565 | 17440 | 21215 | 23425 | 18210 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mmRule |  |  |  |  |  |
| n = 50 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 251800 | 256400 | 261600 | 233200 | 255200 |
|  | 19500 | 19000 | 12500 | 20600 | 25100 |
|  | 17100 | 16800 | 9900 | 17300 | 27600 |
|  | 18400 | 16700 | 16300 | 16800 | 22500 |
|  | 16000 | 16600 | 9200 | 21000 | 27100 |
|  | 22900 | 20100 | 9100 | 21700 | 19400 |
|  | 16600 | 16400 | 10800 | 18900 | 22900 |
|  | 25900 | 26100 | 8700 | 17900 | 35000 |
|  | 16800 | 17400 | 11200 | 17400 | 33900 |
|  | 16300 | 26200 | 11100 | 31000 | 35400 |
|  | 50100 | 59000 | 11200 | 54400 | 38000 |
|  | 19700 | 14600 | 10600 | 31400 | 13700 |
|  | 24600 | 21400 | 10700 | 13500 | 18200 |
|  | 13100 | 13700 | 10500 | 11900 | 21000 |
|  | 18800 | 11100 | 10900 | 34500 | 16600 |
|  | 8900 | 9200 | 11500 | 13300 | 12800 |
|  | 9600 | 22600 | 8900 | 7900 | 11300 |
|  | 12200 | 11900 | 9900 | 7600 | 21600 |
|  | 11300 | 10800 | 10400 | 9600 | 14200 |
|  | 7300 | 10900 | 10300 | 7500 | 10000 |
| average | 29845 | 30845 | 23265 | 30370 | 34075 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mmRule |  |  |  |  |  |
| n = 100 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 266200 | 290500 | 301200 | 257900 | 517700 |
|  | 36800 | 44600 | 44700 | 50700 | 55600 |
|  | 34600 | 43000 | 40000 | 41600 | 42600 |
|  | 30000 | 32100 | 33200 | 32100 | 47100 |
|  | 35600 | 35300 | 32800 | 49000 | 54900 |
|  | 48800 | 80900 | 60000 | 45200 | 45700 |
|  | 32400 | 59600 | 36300 | 33400 | 27900 |
|  | 45600 | 44900 | 42100 | 25500 | 26000 |
|  | 32300 | 26800 | 30600 | 21200 | 27400 |
|  | 25500 | 28000 | 25100 | 15600 | 83100 |
|  | 14400 | 18400 | 33600 | 36600 | 18700 |
|  | 20400 | 29000 | 16500 | 18300 | 26900 |
|  | 16500 | 15500 | 19200 | 17000 | 29300 |
|  | 18200 | 20200 | 23400 | 13600 | 15100 |
|  | 26900 | 16200 | 14600 | 15700 | 8500 |
|  | 15300 | 25700 | 15400 | 15100 | 12300 |
|  | 12200 | 15700 | 15200 | 15400 | 8700 |
|  | 18900 | 6100 | 22500 | 22000 | 12800 |
|  | 17000 | 4000 | 31900 | 3700 | 14300 |
|  | 14700 | 3800 | 14100 | 3500 | 6000 |
| average | 38115 | 42015 | 42620 | 36655 | 54030 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mmRule |  |  |  |  |  |
| n = 500 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 444200 | 439900 | 612800 | 441700 | 434500 |
|  | 183900 | 127900 | 158100 | 145900 | 177300 |
|  | 105600 | 134900 | 121600 | 144000 | 125400 |
|  | 97300 | 55800 | 86500 | 131800 | 89000 |
|  | 69500 | 55500 | 63600 | 92700 | 59300 |
|  | 57100 | 54200 | 61200 | 56300 | 61500 |
|  | 61900 | 50700 | 60100 | 74700 | 57600 |
|  | 54400 | 45500 | 48900 | 47600 | 53200 |
|  | 53000 | 41900 | 56500 | 20800 | 34600 |
|  | 49400 | 40200 | 49100 | 30400 | 61800 |
|  | 124300 | 32500 | 59100 | 31000 | 19500 |
|  | 17100 | 22500 | 20800 | 28700 | 24900 |
|  | 16900 | 17700 | 18800 | 26900 | 17500 |
|  | 17500 | 20300 | 18500 | 21800 | 20500 |
|  | 23700 | 16300 | 19200 | 30700 | 23100 |
|  | 17900 | 18800 | 21000 | 24300 | 34900 |
|  | 23800 | 30300 | 33200 | 48800 | 16200 |
|  | 20100 | 16700 | 19000 | 24700 | 24000 |
|  | 21700 | 17100 | 21400 | 19400 | 21400 |
|  | 19300 | 16800 | 18200 | 16700 | 26200 |
| average | 73930 | 62775 | 78380 | 72945 | 69120 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mmRule |  |  |  |  |  |
| n = 1000 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 592600 | 596300 | 560000 | 687000 | 606800 |
|  | 255800 | 193300 | 242000 | 226700 | 225800 |
|  | 118500 | 101400 | 106700 | 132400 | 112100 |
|  | 110400 | 103700 | 104200 | 114300 | 100900 |
|  | 106100 | 127500 | 95200 | 93400 | 99200 |
|  | 102000 | 71600 | 54700 | 65200 | 79000 |
|  | 38600 | 59200 | 38500 | 61500 | 42500 |
|  | 37300 | 36000 | 49200 | 58600 | 37600 |
|  | 34000 | 34600 | 40700 | 69900 | 35300 |
|  | 49700 | 49700 | 56300 | 54800 | 42600 |
|  | 43500 | 44200 | 37100 | 54900 | 40200 |
|  | 37900 | 31700 | 44500 | 45400 | 46200 |
|  | 43200 | 39200 | 37700 | 49400 | 37600 |
|  | 36200 | 37300 | 34000 | 55200 | 38600 |
|  | 161100 | 37200 | 54200 | 243900 | 155100 |
|  | 35300 | 169800 | 169600 | 48900 | 36100 |
|  | 35200 | 34700 | 36500 | 53700 | 35400 |
|  | 37800 | 35200 | 35100 | 42900 | 32800 |
|  | 47100 | 41300 | 34000 | 48100 | 37100 |
|  | 37100 | 35900 | 33900 | 52200 | 34400 |
| average | 97970 | 93990 | 93205 | 112920 | 93765 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort recursive |  |  |  |  |  |
| n = 10 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2100 | 1800 | 1700 | 1600 | 1700 |
|  | 900 | 1000 | 1100 | 900 | 2000 |
|  | 800 | 1100 | 1100 | 800 | 800 |
|  | 600 | 2900 | 800 | 600 | 1400 |
|  | 1100 | 800 | 700 | 700 | 1100 |
|  | 700 | 700 | 600 | 600 | 1200 |
|  | 700 | 1100 | 3300 | 700 | 1100 |
|  | 1000 | 700 | 700 | 600 | 1100 |
|  | 1000 | 700 | 800 | 1100 | 800 |
|  | 700 | 1100 | 600 | 600 | 1100 |
|  | 700 | 1000 | 1200 | 600 | 1100 |
|  | 700 | 1100 | 1100 | 700 | 1100 |
|  | 600 | 800 | 700 | 900 | 1000 |
|  | 600 | 800 | 700 | 800 | 1100 |
|  | 600 | 900 | 1100 | 700 | 1200 |
|  | 700 | 700 | 800 | 600 | 1300 |
|  | 700 | 600 | 700 | 700 | 1100 |
|  | 700 | 700 | 1100 | 900 | 700 |
|  | 900 | 1100 | 1100 | 1100 | 1100 |
|  | 700 | 1100 | 1100 | 1100 | 1100 |
| average | 825 | 1035 | 1050 | 815 | 1155 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort recursive |  |  |  |  |  |
| n = 50 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2200 | 2400 | 2400 | 2600 | 2300 |
|  | 1800 | 2000 | 2100 | 2000 | 2100 |
|  | 1600 | 1600 | 2100 | 2200 | 3100 |
|  | 1700 | 1600 | 1700 | 1700 | 1800 |
|  | 1800 | 1800 | 1800 | 1700 | 1800 |
|  | 1800 | 1700 | 2200 | 1700 | 2000 |
|  | 1900 | 1700 | 1700 | 2100 | 1800 |
|  | 1700 | 1800 | 2100 | 1800 | 1800 |
|  | 1900 | 1600 | 1700 | 1700 | 2100 |
|  | 1700 | 2100 | 2000 | 1700 | 2200 |
|  | 1700 | 1800 | 1800 | 1700 | 1900 |
|  | 1800 | 4700 | 1700 | 1800 | 2100 |
|  | 1800 | 1900 | 1700 | 1700 | 2000 |
|  | 1700 | 1700 | 1700 | 1700 | 1900 |
|  | 1800 | 1900 | 1700 | 1700 | 2200 |
|  | 1900 | 1800 | 1700 | 1800 | 2000 |
|  | 1800 | 1700 | 1700 | 1800 | 2100 |
|  | 1800 | 2200 | 1700 | 1700 | 2200 |
|  | 2000 | 2000 | 1900 | 5400 | 2200 |
|  | 1700 | 2100 | 2100 | 2300 | 1800 |
| average | 1805 | 2005 | 1875 | 2040 | 2070 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort recursive |  |  |  |  |  |
| n = 100 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 3100 | 3600 | 3900 | 4000 | 4000 |
|  | 3400 | 3200 | 2900 | 3800 | 3000 |
|  | 3000 | 3100 | 3000 | 3400 | 6000 |
|  | 2900 | 3500 | 2900 | 3700 | 3500 |
|  | 3000 | 3300 | 3300 | 3200 | 3200 |
|  | 2900 | 4300 | 3400 | 3500 | 3000 |
|  | 3200 | 3800 | 5300 | 3400 | 3200 |
|  | 3300 | 3300 | 4000 | 3300 | 3100 |
|  | 3100 | 3400 | 3300 | 3300 | 4000 |
|  | 3100 | 3300 | 3200 | 3200 | 4000 |
|  | 4500 | 6200 | 8600 | 6700 | 6100 |
|  | 3300 | 3500 | 3300 | 3200 | 4300 |
|  | 3300 | 3500 | 3400 | 3400 | 3200 |
|  | 3400 | 3300 | 3400 | 3400 | 3100 |
|  | 3100 | 3400 | 3300 | 3700 | 3300 |
|  | 3000 | 3300 | 3400 | 3400 | 3000 |
|  | 2900 | 3200 | 3300 | 3300 | 3500 |
|  | 3100 | 3300 | 3500 | 3300 | 3100 |
|  | 3400 | 3000 | 3500 | 3400 | 3500 |
|  | 3400 | 3300 | 3400 | 3400 | 3300 |
| average | 3220 | 3540 | 3715 | 3600 | 3670 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort recursive |  |  |  |  |  |
| n = 500 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 13400 | 14500 | 11000 | 14600 | 14800 |
|  | 11600 | 14800 | 11300 | 13200 | 12600 |
|  | 14500 | 16100 | 13200 | 15700 | 14800 |
|  | 13800 | 13400 | 13900 | 12800 | 12900 |
|  | 14700 | 16200 | 18800 | 14500 | 16700 |
|  | 13500 | 13200 | 19100 | 16300 | 12900 |
|  | 14400 | 15300 | 16200 | 19600 | 15700 |
|  | 13800 | 12700 | 13400 | 13500 | 12700 |
|  | 14700 | 15800 | 17400 | 16300 | 15700 |
|  | 14200 | 13200 | 13200 | 13900 | 13400 |
|  | 14900 | 15900 | 15500 | 16000 | 15600 |
|  | 13900 | 13200 | 13000 | 13000 | 13400 |
|  | 14600 | 15800 | 15300 | 14100 | 14800 |
|  | 13800 | 13300 | 13300 | 13200 | 12900 |
|  | 14700 | 14200 | 15700 | 14000 | 16100 |
|  | 13500 | 13200 | 13500 | 13100 | 12900 |
|  | 14300 | 14400 | 14500 | 15100 | 15600 |
|  | 13400 | 13300 | 13600 | 15700 | 12800 |
|  | 14400 | 15200 | 15800 | 15800 | 15600 |
|  | 13200 | 13300 | 13300 | 13100 | 12900 |
| average | 13965 | 14350 | 14550 | 14675 | 14240 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort recursive |  |  |  |  |  |
| n = 1000 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 19900 | 22800 | 27600 | 25500 | 17200 |
|  | 28400 | 29000 | 28800 | 27900 | 20300 |
|  | 29100 | 24200 | 35200 | 27300 | 26800 |
|  | 27400 | 28600 | 29200 | 26800 | 26200 |
|  | 28600 | 28900 | 27100 | 27400 | 28500 |
|  | 29000 | 28500 | 27700 | 28800 | 29100 |
|  | 28800 | 27800 | 30200 | 27600 | 27600 |
|  | 29500 | 28100 | 28300 | 28500 | 28500 |
|  | 28300 | 31100 | 28400 | 28100 | 28000 |
|  | 27400 | 30900 | 27400 | 36200 | 28600 |
|  | 26900 | 37300 | 27500 | 35800 | 28600 |
|  | 28600 | 36200 | 26700 | 37000 | 28500 |
|  | 27200 | 37700 | 26000 | 35100 | 27800 |
|  | 28700 | 34200 | 27300 | 33300 | 27600 |
|  | 27100 | 40700 | 28700 | 32000 | 26800 |
|  | 26800 | 39100 | 28600 | 40600 | 28900 |
|  | 28000 | 35200 | 29800 | 28900 | 33100 |
|  | 28900 | 34200 | 27500 | 39400 | 28400 |
|  | 27900 | 50200 | 31200 | 38200 | 28700 |
|  | 27500 | 28300 | 28400 | 34600 | 27300 |
| average | 27700 | 32650 | 28580 | 31950 | 27325 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort Partition |  |  |  |  |  |
| n = 10 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 1600 | 1600 | 1700 | 1600 | 1500 |
|  | 900 | 900 | 1100 | 1100 | 1700 |
|  | 1000 | 1000 | 1000 | 1200 | 1100 |
|  | 700 | 700 | 1000 | 1000 | 700 |
|  | 1100 | 700 | 3000 | 1100 | 600 |
|  | 600 | 1100 | 800 | 1100 | 1100 |
|  | 600 | 4500 | 1100 | 2100 | 1400 |
|  | 600 | 1000 | 700 | 1300 | 1100 |
|  | 600 | 1100 | 700 | 1000 | 900 |
|  | 1100 | 900 | 600 | 1200 | 1100 |
|  | 600 | 1200 | 600 | 1200 | 1000 |
|  | 700 | 700 | 600 | 1200 | 1000 |
|  | 500 | 700 | 700 | 1000 | 1100 |
|  | 600 | 1100 | 1100 | 900 | 1100 |
|  | 600 | 900 | 1000 | 1000 | 700 |
|  | 600 | 800 | 1000 | 1900 | 700 |
|  | 500 | 1100 | 900 | 1100 | 700 |
|  | 2000 | 1100 | 700 | 1000 | 1100 |
|  | 800 | 1000 | 700 | 1300 | 1100 |
|  | 1200 | 1000 | 1100 | 1100 | 700 |
| average | 845 | 1155 | 1005 | 1220 | 1020 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort Partition |  |  |  |  |  |
| n = 50 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 2400 | 2700 | 2500 | 2700 | 2800 |
|  | 1800 | 4500 | 1700 | 2900 | 1800 |
|  | 1600 | 2100 | 1600 | 1800 | 1500 |
|  | 1600 | 1900 | 1600 | 1700 | 1900 |
|  | 1500 | 1600 | 1600 | 1500 | 1800 |
|  | 1500 | 2000 | 1600 | 1600 | 4300 |
|  | 1600 | 2000 | 1900 | 1500 | 2600 |
|  | 1600 | 1800 | 2500 | 2100 | 1700 |
|  | 1600 | 2100 | 2000 | 1900 | 1500 |
|  | 1700 | 2000 | 1800 | 1600 | 2700 |
|  | 1500 | 1800 | 1600 | 1900 | 2000 |
|  | 1600 | 1900 | 1800 | 2000 | 2000 |
|  | 1500 | 1900 | 1600 | 2100 | 1900 |
|  | 1500 | 2000 | 1500 | 1800 | 2000 |
|  | 1800 | 1900 | 1600 | 2000 | 2000 |
|  | 1700 | 1900 | 1800 | 1600 | 1700 |
|  | 1700 | 2000 | 2000 | 2000 | 1900 |
|  | 1900 | 1800 | 1900 | 1600 | 2000 |
|  | 1900 | 2100 | 1600 | 1500 | 1800 |
|  | 2000 | 1900 | 1600 | 2000 | 2000 |
| average | 1700 | 2095 | 1790 | 1890 | 2095 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort Partition |  |  |  |  |  |
| n = 100 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 3000 | 2400 | 3800 | 3600 | 3700 |
|  | 2000 | 3700 | 3000 | 2800 | 2900 |
|  | 3100 | 4200 | 3000 | 2900 | 2800 |
|  | 2800 | 3000 | 2800 | 2900 | 2900 |
|  | 3000 | 3500 | 2700 | 2700 | 2700 |
|  | 3000 | 6900 | 5200 | 2800 | 4500 |
|  | 2800 | 3100 | 3200 | 3000 | 3100 |
|  | 3000 | 5500 | 3200 | 2900 | 3100 |
|  | 3200 | 3300 | 3100 | 5300 | 3100 |
|  | 6100 | 3100 | 3000 | 3300 | 2800 |
|  | 7000 | 5900 | 5200 | 8600 | 5800 |
|  | 3000 | 3100 | 2900 | 3000 | 3100 |
|  | 2600 | 3000 | 3000 | 3200 | 3100 |
|  | 3100 | 3100 | 3000 | 3000 | 2800 |
|  | 3000 | 3200 | 2900 | 3000 | 3000 |
|  | 3000 | 4000 | 3100 | 3000 | 2900 |
|  | 3100 | 3100 | 3100 | 3100 | 2800 |
|  | 3000 | 3000 | 3400 | 4200 | 2900 |
|  | 3000 | 2700 | 3100 | 3100 | 2700 |
|  | 3100 | 3000 | 3100 | 3400 | 2900 |
| average | 3295 | 3640 | 3290 | 3490 | 3180 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort Partition |  |  |  |  |  |
| n = 500 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 13400 | 10600 | 12600 | 13100 | 13000 |
|  | 11400 | 8700 | 11200 | 11600 | 11600 |
|  | 13100 | 13000 | 14900 | 13900 | 13600 |
|  | 12000 | 11800 | 11900 | 11800 | 13200 |
|  | 13100 | 13100 | 13800 | 14200 | 13100 |
|  | 12200 | 12300 | 12900 | 12000 | 11600 |
|  | 12700 | 14500 | 12600 | 13400 | 14100 |
|  | 12200 | 12000 | 12100 | 12500 | 12400 |
|  | 13100 | 12600 | 14100 | 13200 | 13400 |
|  | 14400 | 12900 | 11900 | 11700 | 11900 |
|  | 13900 | 13400 | 12600 | 13500 | 18200 |
|  | 11600 | 12000 | 12200 | 11700 | 11900 |
|  | 12900 | 13200 | 12800 | 13400 | 14000 |
|  | 12300 | 12200 | 12200 | 12600 | 11400 |
|  | 12700 | 12700 | 12900 | 13200 | 14100 |
|  | 12200 | 12200 | 12100 | 14700 | 12400 |
|  | 13100 | 13000 | 13000 | 13100 | 14000 |
|  | 12200 | 12100 | 12100 | 11900 | 11900 |
|  | 13200 | 13900 | 12700 | 14400 | 16900 |
|  | 12200 | 11900 | 12100 | 11500 | 15500 |
| average | 12695 | 12405 | 12635 | 12870 | 13410 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| qSort Partition |  |  |  |  |  |
| n = 1000 |  |  |  |  |  |
| k | 1 | n/4 | n/2 | 3n/4 | n |
|  | 23400 | 22200 | 22500 | 34200 | 23700 |
|  | 19100 | 20700 | 23800 | 26500 | 24600 |
|  | 23600 | 24500 | 22500 | 30200 | 25300 |
|  | 24400 | 24600 | 28200 | 30600 | 24900 |
|  | 23800 | 24700 | 23900 | 33900 | 25200 |
|  | 24400 | 25000 | 30800 | 31400 | 25000 |
|  | 24900 | 25100 | 25600 | 29500 | 24900 |
|  | 23000 | 24900 | 25200 | 31000 | 24500 |
|  | 23900 | 25700 | 27800 | 49400 | 28900 |
|  | 24700 | 25600 | 25100 | 33100 | 27400 |
|  | 23700 | 25900 | 25100 | 37100 | 25900 |
|  | 24400 | 26200 | 26800 | 32000 | 26000 |
|  | 24400 | 25100 | 25100 | 32100 | 25800 |
|  | 23600 | 24200 | 30100 | 32300 | 25300 |
|  | 24300 | 25400 | 25100 | 33800 | 23800 |
|  | 24400 | 25000 | 25000 | 62500 | 24000 |
|  | 24300 | 26000 | 25100 | 30800 | 24000 |
|  | 24400 | 24100 | 24800 | 29900 | 25300 |
|  | 25600 | 25600 | 24600 | 36500 | 23600 |
|  | 25400 | 25300 | 25900 | 38500 | 23900 |
| average | 23985 | 24790 | 25650 | 34765 | 25100 |

Test Strategies

The first algorithm uses Merge Sort to find the kth smallest element. The algorithm first separates the array into two smaller subarrays. It will the recursively call itself to repeat this process on both halves of the subarray and continuously create halves of the array. The process continues until the array has been sorted with the smallest element on the left and the largest element on the right.

The Quick Sort algorithm has a recursive and partition method for the second algorithm. For the partition method of Quick Sort, the last element of the array was used as a pivot. Using the first element of the array will provide the same complexity. At the point of pivot on the array, starting from the beginning of the array at index zero, the algorithm will then move the values that are less than the pivot to the left and swap with those that are greater. Essentially, values that are less than the pivot will be on the left and values that are greater than the pivot will be on the right. The recursive method of Quick Sort continuously executes this method until it is order from least to greatest.

Finally for the third algorithm, the median of medians rule was used. Similar to the pervious algorithms, we would dive the array into smaller subarrays and find the median of all the subarrays. The partition method will then be called to determine the position of the element. Multiple methods were created to support the median of medians rule.

To test the algorithms, each algorithm was executed with five different sized arrays (10, 50, 100, 500 and 1000) and with each size, a different kth element was being searched for (1, n/2, n/2, 3n/4, and n) . A method called “fillArray” was created in order to randomly create an array of size n. Each group was executed twenty times. This means that there is a total of 500 data points collected.

Results

The data collected from the tables and graphs shows us that the Quick Sort algorithms, on average, had a faster runtime compared to the other algorithms. As expected, with the small sample size the test was conducted, these two algorithms performed faster than the others. Quick sort partition method has a time complexity of O(n) for its best and average case but a time complexity of O(n2) for its worst case. The case is similar for Quick sort recursive. If we increased the arrays size to an unfathomable size, then perhaps we can find the Merge Sort method to be the fastest. Merge sort has a tie complexity of O(n logn) for its best, worst and average cases. In this test, our sample size was fairly small, so we do not see the potential Algorithm 1 has. Similarly, for the median of medians rule, the time complexity of its worst case is O(n).

The value of the kth element increases from 1 to the size of the array n. At k = 1, all the algorithms have the lowest runtime it can possibly be. The Quick sort algorithms have the fastest times with and average of around 800 nanoseconds and the mm Rule had the slowest time of an average of 1800 nanoseconds. The Merge Sort method fell in between with an average of 1100 nanoseconds. However, as the size of the array increased, the Quick Sort methods quickly fell behind to the Merge Sort method. Ideally this is not the case and when we compare the data to the other k values, we find the Quick Sort is the fastest with all k values. This could be an error in the code since it was also being modified during the attempts of running the tests (found errors while running). Ideally, for k = 1, we do not expect Merge Sort to be the fastest however this is a special case. As the value of k increases, we see a trend of Quick Sort methods being the fastest, Merge Sort in the middle and mmRule the slowest. Another instance of a differing trend is at k = n. The mmRule is the slowest for the smaller sized arrays, however as the size of the array increases, the Merge Sort method quickly overtakes it and obtains a slower runtime. This is not expected but there is also no guarantee of protection from interferences. This could be from a program running the background during the runtime of the test. However, if we continued to test the larger sizes, we can see from the graph that the lines of Merge Sort and mm Rule will possibly meet and potentially cross again. Further testing is needed with minimal interferences. Theoretically, the size of the array is not large enough to compare the potential of Merge Sort. In the case that I do obtain the resources to use an extremely large array size, we can expect the Quick Sort methods to increase in time exponentially, overtaking the other algorithms. However we cannot see it here.

Theoretical Complexity and Comparisons

Merge Sort is a recursive algorithm with a time complexity of O(n logn) for its best, worst and average case. As mentioned before, it takes the array and divide it into two subarrays. This process continues recursively and to merge all the elements of all arrays together takes linear time (n). This gives us a total time complexity of O(n logn). Due to the algorithm needing to copy the elements of an array into a temporary array, the space complexity is O(n). This algorithm, for smaller sized arrays, will tend to perform worse than Quick sort because Merge Sort still needs perform its entire process even after the array has been completely sorted.

The Quick sort had a best and average time complexity of O(n logn) and a worst time complexity of O(n2). The best case of this algorithm is when the pivot is chosen as a mean. Then a sort of binary tree will form with a height of log n. Each level of the tree, with the best case, will be traversed n times. There so the best case is O(n logn). The worst case occurs when either ends of the array is used instead of the mean. The height of the tree formed will be n and each top node will perform n operations continuously n – 1 until only 1 is reached. This creates the worst time complexity of O(n2). Considering the best-case scenario, it may be possible to make it even better. If we know the half of the array the kth index resides in, we can simply perform the algorithm on that side of tree only. Therefore, the better time complexity will be O(n).

The partition algorithm with the median of medians rule has a time complexity of O(n). similar to the Quick sort algorithm, we take the median of the subarrays that will be created, hence the name median of medians. The time it takes to find the median is only O(1) because it is a simple formula. The remainder of the code that creates subarrays is only O(n). The worst case is only the largest number to the median of medians or the smallest number to the median of medians.

Strengths and Constraints

The strength in the experiment followed my expectations well. The time complexity and runtime was expected for the most part. There are few outliers such as when k = 1 as mentioned as above. An easy solution of improvement for that is to perform testing first before utilizing the code for data. I could have made sure that the entirety of the codes worked before starting the tests. Even with this slight error, the main image of the algorithms has been portrayed properly. Another major improvement for this code is the size of the array. It could have been increased more to better visualize the runtime of the algorithms. However, that also falls into my constraints. I would need a computer capable of dealing with that much data. My PC also had many other programs running in the background. Such as videos, games and even video games which can take a good portion of the RAM. This could have fixed the outlier data as well. Otherwise, most of the data collected was within reasonable ranges considering the capabilities of my available resources.