Sorting MiniProject:

(48 pts) Due May 21

Contents

Part A: Getting the Sorting Algorithms Working

(3 pts) MakeArray:

Write a function that takes as input parameters an array of integers and a size (of the array) and creates a new (on the heap) array of random numbers and returns that array.

(3 pts) PrintArray

Write a function that takes as input parameters an array of integers and a size (of the array and prints out the array

Get the following functions to work (Tested as indicated below):

(2 pts) SelectionSort,

(2 pts) InsertionSort,

(4 pts) QuickSort, and

(4 pts)MergeSort

(note: I gave you the code for these in the hands-on part!)

Test to make sure the above is working by, for each of the above sorting algorithm:

a. Generating a new random array of 100,
b. Sort the array using one of the 4 sorting algorithms
c. Print out the sorted array
d. Delete the array from the heap

Note: The random number generator uses a time seed to generate random numbers. You should only seed the random number generator once. If you repeat the seeding process, the random number generator will start over where it left off. So you want to seed the random number generator once in Main, and then it will continue where it left off every time it generates new random numbers and you will get a different random sequence each time.

Part B: Testing the Timing of the Sorting Algorith

For this part, you're going to be testing the speed of the different sorting algorithms. You will want to increase the size of your arrays to 10000 so we can test the speed of sorting on decently sized random arrays of numbers. In order to determine the running time of sorting 10000 random integers in microseconds, you will use the following:

You will need to include:

```
#include <chrono>
using namespace std;
using namespace std::chrono;
```

At the top of your .cpp file,and then in order to determine the duration in microseconds, you'll be using:

```
auto start = high_resolution_clock::now();
```

and

```
auto stop = high_resolution_clock::now();
```

Then for the timing:

```
auto duration = duration_cast<microseconds>(stop - start);
int ms= duration.count();
```

ms now holds the duration in ms.

To give you an example, if I wanted to determine the timing of insertionSort sorting 10000 integers in microseconds, I'd do the following:

```
#include <chrono>
using namespace std;
using namespace std::chrono;



size = 10000;
arr = makearr(size);
auto start = high_resolution_clock::now();
insertionSort(arr,size);
auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop - start);
cout << "Time for insertionSort is " << duration.count() <<endl;
```

Timing: (12 pts, 3 for each sorting algorithm)

For this part you should do the following:

A. For each of the four sorting algorithms:
   a. Loop 5 times, and each time:
      i. generate a random array of size 10000
      ii. Sort the random array, while:
      iii. Timing the sorting of the random array
      iv. Print out the time in microseconds
      v. Keep track of the average
   b. Then print out the average

Do this for each of the 4 sorting algorithms, to see their average running times. Place the average running time in the Hands-on Exercise.

Part C: Sorting Modifications:

(6 pts) MinMaxSort

SelectionSort is a basic sorting algorithm in which you go through an array to find the smallest, and then swap it with the first value in the array. The process is repeated until the entire array is soted.

You can speed up SelectionSort by simultaneously searching for the smallest value and the largest value, Then when you are done one pass, swap the smallest into place at the beginning of the array and the largest into place at the end of the array.

- Write MinMaxSelectionSort (a new function, modified as specified above).
- Generate a random array of size 100, and run MinMaxSelectionSort on this array, and print out the array to test it.

(6 pts): TimSort

QuickSort is another algorithm that has been modified to improve its efficiency. There's an algorithm named TimSort, and this algorithm works as follows: If the size of the partitions is greater than a certain number, quicksort works as normal. If, however, the partition gets under a certain size, InsertionSort is used to sort that partition.

For this, you may use the existing functions. That said, you might want to make a copy of the InsertionSort function and call it, maybe, InsertionSort2. This second version might be ever so slightly different than the InsertionSort I gave you – one change being the number of input parameters (hint – you'll need another one). To be clear, changes to the InsertionSort2 are very minimal.

Write TimSort

Generate a random array of size 1000, and run TimSort on it, with the condition that when the partition gets to be size 100 or less you'll use the InsertionSort2 function. Print out the resulting array.

(6 pts): Find Average Duration:

B) Now, as you did with the first 4 sorting algorithms, for each of these 2 sorting algorithms, generate 5 random arrays of size 10000. Time the duration necessary for sorting these arrays, and keep track of the average time needed to sort all 5 random arrays for both of the 2 new sorting algorithms.

C) Write the average time down in the Hands-on activity.

That's it!!!! Congratulations!!! You survived this semester!!

/**********/

MY OUTPUT:

Printing SelectionSort
65, 125, 174, 216, 334, 365, 367, 454, 558, 583, 604, 770, 1066, 1167, 1198, 1240, 1273, 1378, 1414, 1426,
1499, 1723, 1736, 1819, 1957, 1964, 2194, 2554, 2636, 2637, 2702, 2802, 2881, 3131, 3135, 3140, 3193, 3290, 3581, 3708,
3788, 4003, 4183, 4447, 4515, 4592, 4692, 4780, 5026, 5213, 5408, 5521, 5599, 5684, 5701, 5788, 5915, 5996, 6051, 6055,
6146, 6253, 6326, 6460, 6558, 6720, 6793, 6831, 6879, 6939, 7172, 7177, 7226, 7327, 7375, 7612, 7874, 7894, 7983, 8086,
8177, 8267, 8322, 8367, 8656, 8903, 8905, 8985, 9023, 9089, 9114, 9156, 9310, 9330, 9353, 9548, 9610, 9625, 9640, 9952,

Printing InsertionSort
9, 22, 86, 111, 169, 235, 293, 301, 342, 465, 542, 582, 700, 740, 990, 1017, 1090, 1092, 1203, 1285,
1483, 1598, 1690, 1915, 2067, 2201, 2326, 2508, 2513, 2563, 2570, 2685, 2966, 2992, 3118, 3308, 4026, 4050, 4125, 4143,
4204, 4235, 4608, 4724, 5090, 5117, 5190, 5225, 5320, 5402, 5460, 5605, 5770, 5807, 5864, 5866, 6103, 6184, 6202, 6450,
6455, 6523, 6552, 6653, 6745, 6776, 7010, 7074, 7201, 7253, 7258, 7275, 7405, 7689, 7868, 7932, 8033, 8095, 8112, 8132,
8222, 8301, 8357, 8651, 8705, 8790, 8865, 8963, 9421, 9475, 9524, 9617, 9651, 9662, 9710, 9739, 9820, 9835, 9846, 9852,

Printing QuickSort
90, 161, 340, 610, 686, 811, 842, 878, 922, 1016, 1035, 1340, 1345, 1412, 1429, 1444, 1618, 1626, 1680, 1710,

1988, 2141, 2235, 2245, 2281, 2304, 2346, 2351, 2374, 2394, 2515, 2588, 2732, 3058, 3213, 3292, 3308, 3655, 3666, 3794,
3796, 3886, 3911, 4189, 4264, 4271, 4372, 4410, 4436, 4492, 4671, 4678, 4768, 4975, 5119, 5173, 5855, 5882, 6220, 6242,
6711, 6857, 6879, 6900, 7042, 7072, 7289, 7310, 7324, 7332, 7458, 7531, 7557, 7837, 7921, 8170, 8315, 8468, 8495, 8525,
8650, 8656, 8683, 8877, 8881, 8911, 8942, 8989, 9173, 9249, 9258, 9264, 9329, 9508, 9549, 9791, 9797, 9928, 9930, 9957,

Printing MergeSort
60, 127, 170, 179, 218, 234, 531, 594, 649, 691, 810, 934, 948, 987, 1022, 1269, 1383, 1509, 1526, 1604,
1914, 1942, 2130, 2139, 2203, 2226, 2294, 2475, 2549, 2597, 2753, 2790, 2856, 2993, 3081, 3090, 3129, 3362, 3488, 3573,
3708, 3950, 3975, 4137, 4205, 4216, 4322, 4432, 4612, 4630, 4936, 5140, 5363, 5371, 5808, 5964, 6110, 6272, 6429, 6616,
6880, 6885, 7026, 7150, 7185, 7206, 7400, 7420, 7445, 7507, 7663, 7666, 7681, 7682, 7713, 7866, 7908, 8059, 8198, 8260,
8349, 8483, 8532, 8566, 8578, 8679, 8713, 8723, 8854, 9028, 9092, 9143, 9349, 9467, 9482, 9556, 9720, 9721, 9788, 9889,

**********PART B: Changing array size to 10000 and getting running times**********
Time for selectionSort is 121676
Time for selectionSort is 121706
Time for selectionSort is 117685
Time for selectionSort is 114694
Time for selectionSort is 116686

Ave time of selectionSort: 118489

Time for insertionSort is 62832
Time for insertionSort is 59840
Time for insertionSort is 59062
Time for insertionSort is 62529
Time for insertionSort is 61826

Ave time of insertionSort: 61217.8

Time for quickSort is 999
Time for quickSort is 964
Time for quickSort is 1036
Time for quickSort is 998
Time for quickSort is 997

Ave time of quickSort: 998.8

Time for mergeSort is 1993
Time for mergeSort is 999
Time for mergeSort is 1996
Time for mergeSort is 1993
Time for mergeSort is 1987

Ave time of mergeSort: 1793.6

**********PART C**********

Printing MinMaxSelectionSort (size 100)

86, 92, 106, 238, 521, 638, 706, 710, 724, 756, 774, 899, 919, 929, 1043, 1114, 1120, 1217, 1253, 1283,
1539, 1606, 1621, 1649, 1876, 1890, 1930, 1965, 1981, 2046, 2053, 2070, 2168, 2236, 2332, 2333, 2342, 2453, 2478, 2519,
2582, 2625, 2646, 2658, 2662, 2671, 2678, 2876, 2931, 2971, 3015, 3044, 3177, 3283, 3364, 3539, 3544, 3702, 3746, 4166,
4444, 4476, 4491, 4923, 4952, 4953, 5937, 6021, 6068, 6181, 6215, 6824, 7129, 7580, 7691, 7743, 7776, 7850, 7932, 8100,
8146, 8238, 8240, 8369, 8425, 8542, 8628, 8712, 8764, 8837, 8846, 9101, 9127, 9430, 9494, 9635, 9815, 9864, 9887, 9922,

Printing Timsort (size 1000)

4, 8, 25, 27, 30, 31, 36, 63, 66, 82, 83, 93, 101, 102, 103, 138, 150, 165, 185, 189,
215, 238, 253, 262, 264, 278, 281, 285, 287, 288, 294, 303, 313, 324, 324, 330, 336, 336, 340, 347,
356, 357, 359, 366, 369, 371, 372, 376, 390, 410, 428, 429, 434, 439, 456, 458, 464, 471, 489, 502,
502, 502, 509, 510, 512, 514, 527, 534, 549, 558, 563, 581, 589, 598, 603, 605, 607, 610, 616, 617,
625, 629, 647, 647, 653, 653, 657, 672, 673, 692, 702, 707, 731, 734, 755, 756, 762, 762, 789, 800,
800, 808, 809, 812, 814, 818, 825, 840, 845, 852, 855, 858, 860, 862, 863, 881, 884, 894, 897, 902,
903, 907, 911, 914, 915, 917, 931, 933, 933, 933, 939, 958, 965, 972, 980, 995, 1001, 1015, 1016, 1023,
1029, 1032, 1038, 1045, 1046, 1056, 1063, 1064, 1067, 1083, 1094, 1095, 1101, 1105, 1110, 1113, 1114, 1119, 1126, 1144,
1165, 1168, 1174, 1181, 1189, 1191, 1194, 1194, 1213, 1259, 1280, 1290, 1293, 1299, 1302, 1308, 1324, 1339, 1348, 1353,
1377, 1409, 1424, 1429, 1441, 1445, 1452, 1469, 1480, 1487, 1490, 1499, 1500, 1511, 1517, 1528, 1543, 1613, 1631, 1631,
1655, 1656, 1658, 1669, 1677, 1684, 1693, 1714, 1738, 1754, 1764, 1769, 1777, 1791, 1797, 1799, 1804, 1808, 1819, 1822,
1824, 1825, 1836, 1840, 1849, 1854, 1860, 1860, 1861, 1864, 1869, 1894, 1897, 1899, 1899, 1904, 1905, 1910, 1923, 1931,
1949, 1951, 1951, 1952, 1969, 1971, 1983, 1992, 2003, 2009, 2033, 2036, 2047, 2049, 2050, 2073, 2085, 2091, 2098, 2111,
2122, 2122, 2137, 2140, 2147, 2174, 2179, 2193, 2198, 2200, 2225, 2240, 2244, 2245, 2248, 2252, 2257, 2273, 2281, 2282,
2322, 2324, 2330, 2343, 2350, 2351, 2356, 2367, 2375, 2392, 2406, 2413, 2414, 2429, 2430, 2435, 2437, 2444, 2447, 2452,
2455, 2460, 2461, 2467, 2473, 2481, 2486, 2487, 2490, 2491, 2496, 2501, 2522, 2531, 2534, 2536, 2540, 2549, 2552, 2588,
2595, 2615, 2617, 2625, 2626, 2630, 2630, 2632, 2634, 2642, 2646, 2665, 2679, 2682, 2693, 2698, 2711, 2714, 2732, 2736,
2739, 2765, 2783, 2786, 2802, 2805, 2806, 2806, 2814, 2825, 2826, 2837, 2837, 2848, 2871, 2880, 2886, 2892, 2907, 2930,
2953, 2965, 2987, 2996, 2996, 2998, 3005, 3018, 3023, 3044, 3072, 3075, 3077, 3098, 3104, 3110, 3112, 3124, 3127, 3135,
3139, 3152, 3167, 3177, 3179, 3190, 3203, 3204, 3214, 3218, 3225, 3227, 3228, 3247, 3249, 3263, 3282, 3286, 3295, 3345,
3347, 3350, 3355, 3384, 3394, 3398, 3401, 3407, 3431, 3445, 3468, 3484, 3486, 3509, 3513, 3521, 3555, 3559, 3561, 3574,
3575, 3594, 3595, 3597, 3598, 3620, 3634, 3635, 3647, 3652, 3672, 3677, 3705, 3724, 3737, 3750, 3756, 3781, 3790, 3792,
3804, 3812, 3830, 3831, 3837, 3838, 3857, 3870, 3887, 3899, 3899, 3941, 3957, 3981, 3994, 3995, 3998, 4017, 4028, 4047,
4079, 4103, 4107, 4113, 4115, 4118, 4120, 4151, 4153, 4164, 4171, 4180, 4181, 4226, 4234, 4235, 4250, 4252, 4256, 4257,
4261, 4261, 4292, 4296, 4301, 4323, 4330, 4331, 4340, 4344, 4357, 4359, 4378, 4381, 4399, 4420, 4436, 4441, 4471, 4474,
4487, 4502, 4512, 4517, 4522, 4542, 4547, 4555, 4557, 4574, 4586, 4596, 4597, 4631, 4640, 4661, 4670, 4685, 4715, 4715,
4720, 4722, 4741, 4747, 4747, 4752, 4754, 4755, 4770, 4792, 4794, 4802, 4827, 4851, 4860, 4874, 4892, 4903, 4922, 4935,
4945, 4952, 4966, 4966, 4983, 4985, 4998, 5001, 5011, 5012, 5022, 5045, 5076, 5088, 5103, 5123, 5129, 5131, 5160, 5173,
5174, 5189, 5211, 5220, 5224, 5234, 5234, 5245, 5251, 5264, 5279, 5301, 5315, 5317, 5321, 5331, 5341, 5344, 5349, 5362,
5368, 5374, 5380, 5383, 5406, 5437, 5462, 5466, 5487, 5492, 5495, 5507, 5513, 5514, 5519, 5521, 5526, 5545, 5553, 5558,
5559, 5566, 5600, 5602, 5605, 5609, 5634, 5646, 5663, 5671, 5685, 5699, 5702, 5715, 5747, 5752, 5757, 5757, 5807, 5812,
5819, 5827, 5831, 5862, 5863, 5867, 5878, 5879, 5880, 5913, 5940, 5943, 5947, 5955, 5961, 5981, 5984, 5989, 6017, 6034,
6043, 6044, 6050, 6064, 6068, 6069, 6076, 6079, 6082, 6083, 6104, 6106, 6116, 6121, 6136, 6149, 6155, 6175, 6186, 6204,
6206, 6223, 6256, 6265, 6288, 6289, 6303, 6333, 6338, 6345, 6346, 6348, 6353, 6355, 6359, 6366, 6372, 6380, 6383, 6393,
6404, 6405, 6431, 6433, 6444, 6444, 6478, 6485, 6486, 6503, 6515, 6521, 6526, 6574, 6577, 6579, 6580, 6591, 6598, 6608,
6613, 6617, 6652, 6664, 6715, 6719, 6725, 6726, 6728, 6734, 6747, 6760, 6775, 6776, 6782, 6782, 6820, 6823, 6841, 6842,
6914, 6923, 6925, 6939, 6940, 6942, 6955, 6973, 6997, 7007, 7015, 7018, 7020, 7038, 7055, 7064, 7069, 7071, 7091, 7097,
7102, 7104, 7121, 7144, 7144, 7150, 7160, 7172, 7207, 7207, 7219, 7244, 7245, 7271, 7301, 7306, 7310, 7338, 7344, 7363,
7365, 7385, 7401, 7415, 7421, 7424, 7457, 7468, 7481, 7489, 7490, 7514, 7517, 7558, 7598, 7602, 7603, 7607, 7608, 7619,
7638, 7638, 7642, 7654, 7655, 7661, 7675, 7700, 7703, 7709, 7709, 7718, 7729, 7742, 7748, 7749, 7753, 7790, 7809, 7820,
7820, 7822, 7831, 7850, 7864, 7883, 7887, 7907, 7908, 7933, 7943, 7948, 7950, 7960, 7966, 7973, 7974, 7979, 7998, 7999,
8015, 8032, 8033, 8050, 8075, 8077, 8081, 8099, 8105, 8111, 8128, 8133, 8207, 8212, 8217, 8230, 8245, 8253, 8281, 8291,
8317, 8318, 8351, 8369, 8382, 8385, 8401, 8422, 8426, 8427, 8443, 8462, 8463, 8478, 8491, 8491, 8515, 8517, 8531, 8534,
8544, 8565, 8570, 8575, 8577, 8578, 8584, 8610, 8627, 8629, 8631, 8678, 8678, 8683, 8683, 8684, 8685, 8699, 8717, 8718,
8718, 8725, 8728, 8730, 8731, 8739, 8747, 8796, 8801, 8811, 8812, 8818, 8831, 8847, 8863, 8881, 8891, 8896, 8906, 8908,
8939, 8946, 8952, 8953, 8954, 8973, 9003, 9006, 9007, 9021, 9029, 9057, 9077, 9077, 9092, 9092, 9108, 9130, 9130, 9139,
9164, 9186, 9195, 9216, 9261, 9264, 9292, 9295, 9323, 9328, 9335, 9364, 9372, 9374, 9383, 9395, 9397, 9401, 9405, 9421,
9432, 9468, 9473, 9507, 9512, 9517, 9518, 9519, 9529, 9531, 9533, 9539, 9554, 9556, 9561, 9563, 9573, 9582, 9586, 9598,
9601, 9604, 9620, 9623, 9631, 9635, 9650, 9658, 9663, 9668, 9713, 9723, 9730, 9749, 9759, 9765, 9768, 9769, 9802, 9803,
9808, 9810, 9812, 9838, 9845, 9850, 9854, 9859, 9860, 9875, 9880, 9883, 9893, 9904, 9920, 9936, 9945, 9960, 9989, 9995,

Printing ave running times of MinMax and TimSort (size 10000)
Time for minMaxSelectionSort is 60869
Time for minMaxSelectionSort is 63837
Time for minMaxSelectionSort is 66790
Time for minMaxSelectionSort is 67860
Time for minMaxSelectionSort is 64326

Ave time of minMaxSelectionSort: 64736.4

Time for TimSort is 999
Time for TimSort is 996
Time for TimSort is 997
Time for TimSort is 997
Time for TimSort is 995

Ave time of timSort: 996.8