**Appendix 1 Merge Sort Example**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | | | | | 8 | | | 15 | | | | | 10 | | | 4 | | | | | 16 | | | | | 5 | | | | | | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 12 | | | 8 | | | | 15 | | | | 10 | | | |  | | 4 | | | | | 16 | | | | | | 5 | | | | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 12 | | 8 | | | |  | | | 15 | | | 10 | | |  | | 4 | | 16 | | | | | | |  | | | | | 5 | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 12 |  | | | 8 | |  | | | 15 |  | | | | 10 |  | | 4 |  | | | | | | 16 | | |  | | | | | 5 | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 8 | | 12 | | | |  | | | 10 | | | 15 | | |  | | 4 | | | 16 | | | | |  | | | | | 5 | | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 8 | | | 10 | | | | 12 | | | | 15 | | | |  | | 4 | | | | | 5 | | | | | | 16 | | | | |
|  | | |  | | | |  | | | |  | | | |  | |  | | | | | |  | | | | | |  | | | |
| 4 | | | | | 5 | | | 8 | | | | | 10 | | | 12 | | | | | 15 | | | | | 16 | | | | | | |

|  |  |
| --- | --- |
|  | Unsorted Array |
|  | Sorted Array |

**Appendix 2 Merge Sort Worst Data Set**

         For the worst data set, I have used the general concept of merge sort in the designing in the design process. Merge sort core concept is comparing the smallest element in two sorted arrays until one array is empty. Hence, merge sort performs the worst when require to compare two arrays with alternating monotone sequence. As shown in the diagram on the slide. From a sorted array, alternating elements in a sub array would be even and odd intergers, afterwards, alternating the sub array would be alternating even elements, and alternating odd elements. The process repeats itself until the sub array is less than one, then the alternating elements would switches position again to create a reverse order. The final result is the worst data set for the general merge sort.

Diagram

Description automatically generated

**Appendix A**

The bellow github link contains all the code used for this paper. Each algorithm is sorted by the sorting type and language used. There are comments in each file explaining details of each part of the code. Please feel to take reference from the code.(Uncomplete, await for university arrangement for upload code)(for code please contact [patrickyyung@gmail.com](mailto:patrickyyung@gmail.com) for request)

<https://github.com/patrick-yung/Sorting-Algorithum.git>

Graphical user interface, text, application, email

Description automatically generated

**Appendix B**

The bellow table is the data set used to run all the code in Appendix A and its desire output.

|  |  |
| --- | --- |
| **Test Case** | **Sorted** |
| **Best Time Possible**  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 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 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 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 |
| **Worst Time Possible**  100, 99, 98, 97, 96, 95, 94, 93, 92, 91,90, 89, 88, 87, 86, 85, 84, 83, 82, 81,80, 79, 78, 77, 76, 75, 74, 73, 72, 71,70, 69, 68, 67, 66, 65, 64, 63, 62, 61,60, 59, 58, 57, 56, 55, 54, 53, 52, 51,50, 49, 48, 47, 46, 45, 44, 43, 42, 41,40, 39, 38, 37, 36, 35, 34, 33, 32, 31,30, 29, 28, 27, 26, 25, 24, 23, 22, 21,20, 19, 18, 17, 16, 15, 14, 13, 12, 11,10, 9, 8, 7, 6, 5, 4, 3, 2, 1 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 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 |
| **Random 1**  463, 627, 62, 616, 997, 958, -294, -721, 178, -552, -65, -998, 209, 174, -309, 311, 242, -730, 980, -609, -540, 991, -835, -970, -252, -514, 788, 310, 423, -267, 68, -784, 55, 115, 829, 585, 764, -985, 516, -482, -915, -935, -31, 886, -547, 720, -677, -494, 351, -685, 952, 842, 629, 654, 79, 474, -64, 488, 996, -661, -446, -341, -869, 530, 831, 120, 899, 552, 840, 353, 69, -326, 389, 8, 694, -602, -42, 950, 626, -322, 494, -371, 727, 885, -132, -333, -191, 13, 277, -630, -971, 978, -77, 138, -555, -270, -107, 709, -477, 319 | -998 -985 -971 -970 -935 -915 -869 -835 -784 -730 -721 -685 -677 -661 -630 -609 -602 -555 -552 -547 -540 -514 -494 -482 -477 -446 -371 -341 -333 -326 -322 -309 -294 -270 -267 -252 -191 -132 -107 -77 -65 -64 -42 -31 8 13 55 62 68 69 79 115 120 138 174 178 209 242 277 310 311 319 351 353 389 423 463 474 488 494 516 530 552 585 616 626 627 629 654 694 709 720 727 764 788 829 831 840 842 885 886 899 950 952 958 978 980 991 996 997 |
| **Random 2**  -648, -25, -168, -716, 885, -394, -399, 834, 184, -733, -167, 355, 597, 282, 624, 147, 234, 472, 156, 5, 828, 615, 694, 491, -994, 762, 116, -495, 135, -693, 186, -270, -564, -247, 511, 1, 392, 528, 102, -628, -786, -431, -645, -60, -968, 882, 461, 845, -472, -682, 893, -449, -300, -752, -174, 14, 581, 416, 134, 575, 217, 247, -959, -575, 3, -458, 467, -413, -954, 485, 339, 245, 289, 124, 982, -607, -678, -102, 578, -578, -711, -207, -352, -199, -303, -838, 630, -953, -187, 902, -570, 852, -613, -386, -818, -79, -749, 420, -732, 360 | -994 -968 -959 -954 -953 -838 -818 -786 -752 -749 -733 -732 -716 -711 -693 -682 -678 -648 -645 -628 -613 -607 -578 -575 -570 -564 -495 -472 -458 -449 -431 -413 -399 -394 -386 -352 -303 -300 -270 -247 -207 -199 -187 -174 -168 -167 -102 -79 -60 -25 1 3 5 14 102 116 124 134 135 147 156 184 186 217 234 245 247 282 289 339 355 360 392 416 420 461 467 472 485 491 511 528 575 578 581 597 615 624 630 694 762 828 834 845 852 882 885 893 902 982 |
| **Random 3**  573, -187, -400, -356, -100, 440, 355, -116, -9, 229, -5, 969, -179, 894, -664, 205, -753, -132, 271, 441, 976, -931, -67, -247, -382, 889, 496, -487, -998, -154, -573, 864, 962, -572, -676, -534, 478, -691, -555, -386, -868, -766, -660, 197, 742, 933, -943, 887, -201, -898, 971, -607, -260, 40, -148, 353, -449, 93, 52, -548, 174, 250, 96, -174, -231, 233, 286, 70, -372, -960, 358, 22, -981, -554, -562, 558, 746, 490, -621, -822, -86, 696, -642, -93, -873, 411, 619, -926, 780, -203, 420, -553, -523, -447, -68, 786, 645, 674, -383, 448 | -998 -981 -960 -943 -931 -926 -898 -873 -868 -822 -766 -753 -691 -676 -664 -660 -642 -621 -607 -573 -572 -562 -555 -554 -553 -548 -534 -523 -487 -449 -447 -400 -386 -383 -382 -372 -356 -260 -247 -231 -203 -201 -187 -179 -174 -154 -148 -132 -116 -100 -93 -86 -68 -67 -9 -5 22 40 52 70 93 96 174 197 205 229 233 250 271 286 353 355 358 411 420 440 441 448 478 490 496 558 573 619 645 674 696 742 746 780 786 864 887 889 894 933 962 969 971 976 |

**Appendix C**

Due to the amount of data collected please refer the google sheet, link below for the full detail time for each trial run. The average for each sorting algorithm version is displayed with its variance.

<https://docs.google.com/spreadsheets/d/1cI4CexnblpUPqF-EViVGJo3WUcEG4Qs5s8_39Kc-EHc/edit?usp=sharing>