**Data Structure Project 1**

Deadline: Apr 8

This project requires students to compare five sorting algorithms, which are “Bubble Sort”, “Insertion Sort”, “Merge Sort”, “Quick Sort”, and “Heap Sort” in the aspect of time complexity, best&worst case scenario.

Requirement:

1. Implement the five sorting algorithms based on the skeleton code provided.
2. Compare the running time of five sorting algorithms, and fill the following table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| **Bubble** | **0** | **0** | **0** | **0** | **0** | **0** | **0.003** | **0.007** | **0.032** | **0.096** | **0.378** | **1.97** | **7.678** | **22.488** | **96.277** |
| **Insertion** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0.002** | **0.008** | **0.024** | **0.121** | **0.372** | **1.134** | **4.769** | **21.7** |
| **Merge** | **0** | **0** | **0** | **0** | **0** | **0** | **0.001** | **0.002** | **0.009** | **0.008** | **0.012** | **0.022** | **0.028** | **0.064** | **0.165** |
| **Quick** | **0** | **0** | **0** | **0** | **0** | **0** | **0.001** | **0.001** | **0.002** | **0.004** | **0.007** | **0.011** | **0.012** | **0.026** | **0.113** |
| **Heap** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0.001** | **0.003** | **0.001** | **0.003** | **0.011** | **0.013** | **0.027** | **0.077** |

where each cell in the table denotes the running time (recorded by C++ timer) given the input size (number of elements in the list to be sorted) 2t. For example, at column “17”, each soring algorithm should sort the list containing 217 random integers. Note: in order to be fairness to all the sorting algorithms, the input random integer list should be the same.

1. Use “t” as X-axis and running time (value in each cell in above table) as Y-axis, plot all the points and sketch the curve (You may do this by Excel) for each sorting algorithms. Draw all five curves in one X-Y coordinate plane. Compare the five curves and explain the reason.

Because “Merge Sort”, “Quick Sort”, and “Heap Sort” is more efficient than “Insertion Sort” and “Merge Sort”. It would save a lot time to sort.

1. Describe the best/worst case and the corresponding time complexity of each sorting algorithm. You may fill the tables below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best case  description | Best case example | Best case time  complexity |
| Bubble | Average Time |  | N^2 |
| Insertion | Had been sored | 1,2,3,4,5 | N |
| Merge | Average Time |  | NLOGN |
| Quick | Can divided the array almost into two equal-size-array. | 1,2,3,4,5,6,7 | NLOGN |
| Heap | Had been sored | 1,2,3,4,5 | NLOGN |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Worst case  description | Worst case  example | Worst case time  complexity |
| Bubble | Average Time |  | N^2 |
| Insertion | Almost inverse | 5,4,3,2,1 | N^2 |
| Merge | Average Time |  | NLOGN |
| Quick | Every choice the smallest one as the middle(key) | 4,1,3 | N^2 |
| Heap | Almost inverse and Disorder | 5,4,3,2,1 | NLOGN |

看成中午时间了......over