Spring 2023 – CS 303 Algorithms and Data Structures
Lab 5

Notes:

- Implement the algorithm and analyze the results using the give input files
- **Deliverables**: Report.pdf file and your code file (please do not send a zip file. If you have more than one class in your code, then submit each file separately through Canvas.)
- Homework report must follow the guidelines provided in the sample report uploaded in Canvas

Objectives:

- Implement basic quick sort algorithm
- Implement quick sort using median of 3 partitioning
- Compare the performance of insertion sort, merge sort, heap sort, and quick sort.

Problems

1. Implement a method to sort a given array using the basic quicksort algorithm. Use the algorithm from the textbook (see page 2)
2. Write a driver program to test the quicksort algorithm for the file uploaded in the canvas.
3. Compare the performance of the quicksort algorithm with 3 cases of input files: sorted, reversed sorted, and random. These files are provided in Canvas in the Quicksort Input Files folder.
4. Use the median of 3 partitioning algorithm (given in the next page) to implement quick sort. This algorithm chooses the pivot element as the median of the 3 elements namely: A[p], A[r], and A[(p+r)/2].
5. Compare the performance of the quicksort using median of 3 partitioning with the basic quicksort algorithm using the input files located on Canvas in Quicksort Input Files folder.
6. Compare the execution time of quicksort with the execution time of insertion sort, merge sort, heap sort. Make sure you use the same array to compare the performance. Use a table or plot to summarize the results and document your observations and analysis in the report. You can use the input files ranging from 100 to 500,000 or the input files ranging from 16 to 8192.

To sort an entire array A, the initial call is QUICKSORT$(A, 1, A.length)$.

```
QUICKSORT(A, p, r)
1  if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4       if A[j] ≤ x
5            i = i + 1
6            exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Quicksort Algorithm using Median of 3 partitioning

```
Quicksort(A, p, r)
1. if p < r
2.    N = r - p + 1
3.    m = Median3(A, p, p + N/2 , r)
4.    exchange A[m] with A[r]
5.    q = Partition(A, p, r)
6.    Quicksort(A, p, q-1)
7     Quicksort(A, q+1, r)
```

Median3(A, i, j, k) : Returns among i, j and k, the position that holds the median value.

Partition(A, p, r) : is just as before.

Note: The above pseudo code assumes that the array indexing is starting from 1. If you are using a programming language that uses array indexing starting from 0, you have to modify the pseudo code accordingly.

Submission:

- You are required to submit a written report (.pdf) and your project file (.zip) to the Canvas
- Homework report must follow the guidelines provided in the sample report uploaded in Canvas. Please include the screenshot of your code and outputs of your code at the end of your report.
- Do not forget to submit Independent Completion Form
- When you create the code to read the file use relative path instead of absolute path

DATA

16,32,64,128,256,512,1024,2048,4096,8192, random, reverseSorted, sorted

Grading Rubric

Coding	Implementing Algorithms	20 points
	Producing Correct Outputs	20 points
Report	Explaining the algorithms used	10 points
	Displaying the output with a graph or table	20 points
	Comparing the outputs and discussing the time complexity of algorithms	20 points
	Correct submissions of the files (ICF, Code.zip, report.pdf)	10 points