**Latex End sem Sample Question 2**

**Sorting Algorithms**

For this activity, you are required to replicate the given pdf “Sortingalgos.pdf".

You have to fill up the main.tex, which can generate a pdf as close as possible to “Sortingalgos.pdf".

Below is the detailed description of the document you need to create.

All the packages required for this activity are already imported in the given "main.tex" file. You shouldn't and mustn't import any other packages.

Following are the major components in each part of the document, which would be considered for evaluation:

1) Preamble

* Title
* Student Id
* Table of contents

2) Section 1 : Introduction

* basic introduction about algorithms, you can use the same text as in the “Sortingalgos.pdf” file.

3) Section 2: Types of Sorting Algorithms

* use list to represent the different types of sorting algorithms

4) Section 3: Complexity of Sorting Algorithms

* basic terminology about the metrics to evaluate the sorting algorithms, you can use the same text as in the “Sortingalgos.pdf” file.

5) Section 4: Time Complexity Equations

* time complexity equations of various sorting algorithms.

6) Section 5: Comparison

* use tables to compare the performance of various algorithms in various cases.

7) Section 6: Conclusion

* basic conclusion information about sorting algorithm, you can use the same text as in “Sortingalgos.pdf” file.

Usage :

pdflatex main

Grading details : Your submission will be graded on the basis of the certain tags in the "main.tex" file, though iit must generate a pdf without any errors, warnings are acceptable.

Hence, in most cases exact match between your rendered pdf and “Sortingalgos.pdf” is not considered, yet it is advisable to create your pdf similar to the given“Sortingalgos.pdf".

\documentclass{article}

% Do not include any other packages

\title{SSL}

\author{22XXXX(ID) }

\date{May 2023}

\begin{document}

% preamble

\maketitle

% below line auto generates the table of contents

\tableofcontents

\clearpage

% section --1

% Introduction section %

% paragraph

\section{Intro}

A sorting algorithm is used to arrange elements of an array/list in a specific

order. A sorting algorithm is considered stable if the two or more items with

the same value maintain the same relative positions even after sorting. \\

There are various sorting algorithms that can be used to complete this operation.

And, we can use any algorithm based on the requirement.

% section --2

% Types of Sorting algorithms

\section{Types of Sorting algorithms}

Sorting algorithms are a fundamental concept in computer science that arrange a

collection of elements or data in a particular order. There are numerous sorting

algorithms, each with its own characteristics, advantages, and disadvantages.

Here are descriptions of some commonly used sorting algorithms:

\begin{enumerate}

\item bubble sort

\item insertion sort

\end{enumerate}

% section --3

% Complexity of Sorting algorithms

\section{Complexity}

The efficiency of any sorting algorithm is determined by the time complexity

and space complexity of the algorithm.

\begin{itemize}

\item Time Complexity: Time complexity refers to the time taken by an algo

rithm to complete its execution with respect to the size of the input.

\item Space Complexity: Space complexity refers to the total amount of memory

used by the algorithm for a complete execution. It includes both the

auxiliary memory and the input

\end{itemize}

%section --4

%Time complexity equations

\section{Time complex}

Time Complexity Equations for various Algorithms :

\begin{equation}

T(n) = 2T(n/2)+O(n) \end{equation}

\begin{equation} hello \end{equation}

\begin{equation} T(n) = T(2n/3)+O(n)

\end{equation}

equation 1 is the time complexity equation corresponding to Merge Sort and

equation 2 is the time complexity equation corresponding to Quick Sort

%section --5

%Comparison

Let’s see a complexity analysis of different sorting algorithms

\begin{tabular}{|c|c|c|c|c|}

\hline \\

& & & & \\ \hline

& & & & \\ \hline

& & & & \\ \hline

& & & & \\ \hline

& & & & \\ \hline

\end{tabular}

%section --6

% Conclusion

\section{Conclusion}

In conclusion, the choice of the best sorting algorithm depends on various factors

such as the size of the data set, the degree of pre-sorting, stability requirements,

and the trade-off between time complexity and space complexity. Here’s a brief

summary of the comparison and recommendations for different cases:

\begin{itemize}

\item Bubble Sort is simple to understand and implement, but it has a quadratic

time complexity. It is suitable for small datasets or nearly sorted lists.

\item Merge Sort has a time complexity of O(n log n) and is stable. It per

forms well on large datasets and is suitable for external sorting where

data doesn’t fit into memory.

\item Quick Sort is efficient in practice. It is suitable for large datasets

\end{itemize}

\end{document}