DC ASSIGNMENT – 3

Name: Rahul Varma

Roll No: S20200010212

(a) Odd-Even Transposition Algorithm for distributed sorting on a line network

(b) Sasaki's time-optimal algorithm for distributed sorting on a line network

(c) An alternative time-optimal algorithm for distributed sorting on a line network

1. Odd-Even Transposition Algorithm for distributed sorting on a line network:

N Rounds Algorithm.

Implementation Details:

The code consists of the implementation of the Odd-Even Transposition Sort algorithm using message passing between processes. It defines two structs, one for a process and one for an event, which includes information about the source and target processes, the type of event (send or receive), a flag to indicate whether it's an even or odd round, and a value associated with the event.

The main function generates a sequence of processes and then starts the sorting algorithm. It initializes an event queue and then enters a loop that executes for n/2 rounds, where n is the number of processes. Each round is divided into two phases, an even phase and an odd phase. In the even phase, each process sends a message to its right neighbor if its flag is even, and in the odd phase, each process sends a message to its left neighbor if its flag is odd. After sending a message, the process waits to receive a message from its left or right neighbor depending on the phase. When a process receives a message, it compares the value in the message with its own value and swaps them if necessary. Finally, the event queue is cleaned up and the sorted sequence is printed.

Time and Space Complexity:

The complexity of this algorithm is O(n^2), where n is the number of processes. This is because each process sends and receives messages with its neighbors for n/2 rounds, and each message exchange requires O(1) time. Therefore, the total time complexity is O(n^2). The space complexity is O(n) because the sequence of processes and the event queue both require O(n) space.

Choice of Data Structure:

The program uses several data structures, including:

* struct process: a structure that contains an ID and a value. It is used to represent the processes that will be sorted.
* struct event: a structure that contains information about an event, such as its source, target, type, flag, and value. It is used to implement the distributed algorithm for sorting.
* Linked list: events are stored in a linked list data structure.
* Pointers: pointers are used extensively to manipulate the linked list.
* Dynamic memory allocation: dynamic memory allocation is used to create and manipulate the linked list.
* Arrays: arrays are used to store and manipulate the process sequence that will be sorted.
* Stack: a stack is used to keep track of the processes that are being sorted.

1. Sasaki's time-optimal algorithm for distributed sorting on a line network

N -1 Round’s Algorithm

Implementation Details:

No Global position

Make copies of elements at intermediate nodes

Rule to select Final Solution

If Area = -1 -> select Right One

else -> select Left One.

Computing n at runtime

Time and Space Complexity:

Space Complexity

Choice of Data Structure:

1. An alternative time-optimal algorithm for distributed sorting on a line network

N – 1 Round’s Algorithm