

SPPU-BE-COMP-CONTENT - KSKA Git

Q1) Do the Analysis of Parallel Bubble sort and find out its Time Complexity.

Ans. Parallel Bubble Sort

- parallel bubble sort is often implemented using the odd-even transportation sort method which allows parallel comparisons.

→ Working:-

(1) Odd Phase

Compare and swap
(1,2), (3,4), (5,6),

(2) Even phase

(0,1), (2,3), (4,5),

Each phase can run comparisons in parallel because no. two comparisons overlap.

→ Analysis:-

• n elements

• p process

Number of phases

To Guarantee sorting we need:-

' n ' phases.

→ CASE 1:- Unlimited Processors ($p \geq n/2$)

- Each phase performs about $n/2$ comparisons in parallel.

• Time per phase = $O(1)$

• Total Phase = n

Time Complexity = $O(n)$

→ CASE 2:- Limited Processors ($p < n/2$)

Each phase performs:

$= n/2$ comparisons.

SPPU-BE-COMP-CONTENT – KSKA Git

With p processors, comparisons per processor
 $= (n/2) / p$

Time per phase:

$$O(n/p)$$

Total time:

$$n \times (n/p) \equiv O(n^2/p)$$

→ Hence, Time Complexity is:-

① For parallel Bubble sort ($p \geq n/2$) $O(n)$

② For Parallel Bubble sort ($p < n/2$) $O(n^2/p)$

Q2. Difference Between Parallel Bubble Sort and Parallel Merge Sort.

ANS.

1. PARALLEL BUBBLE SORT:-	2. PARALLEL MERGE SORT:-
1. Repeatedly compares and swaps adjacent elements using multiple processors.	1. Divides array into sub-arrays, sort them in parallel and merge.
2. Comparison and swapping based sorting.	2. Divide and Conquer based sorting.
3. Uses techniques like Odd-Even Transportation sort.	3. Different processor sorts different sub-arrays and then merge them in stages.
4. Time Complexity: $O(n^2)$; Parallel improvement is limited.	4. Time Complexity: $O(n \log n)$ Parallelism improves further.
5. Poor scalability with increasing processors.	5. Good Scalability with increasing processors.

SPPU-BE-COMP-CONTENT - KSKA Git

Q3>

Comment on scaling parallel Merge sort, 32019 660

ANS.

Parallel Merge sort scales well with increasing processor because:-

(1) Divide and Conquer Nature/Strategy:-

- The input array is divided into smaller sub-arrays that can be sorted independently by different processors.

(2) Balanced Workloads:-

- Each processor handles rough & equal sized data segments, reducing load imbalance

(3) Parallel Merging

- Merging stages can also be parallelized, although they require communication between processors.

(4) Speed Up

- For 'p' processors, time complexity can approach

$$O\left(\frac{n \log n}{p} + \log p\right)$$

(5) Limitations:-

- ① Communication Overhead during merging.
- ② Memory Requirement for temporary Arrays

Q4>

How Bubble Sort Algorithm can be parallelized?

ANS.

The Bubble-sort can be parallelized using the Odd-Even Transposition Method:

→ STEPS:-

1. Divide the array among processors.
2. Perform two phases independently, and Repeatedly.

SPPU-BE-COMP-CONTENT - KSKA Git

• Odd phase:-

- Compare and swap elements:

$(1,2), (3,4), (5,6), \dots$

- These comparisons are independent and run in parallel.

• Even Phase:-

- Compare and swap the elements:-

$(0,1), (2,3), (4,5), \dots$

(3) Continue, alternating odd-even phase until the array is sorted

For Example:-

Array $[8, 4, 6, 2]$

Odd phase :-

$(1,2) \rightarrow$ Compare 4 and 6.

Even phase :-

$(0,1) \rightarrow$ Compare 4 and 8 ; $(2,3) \rightarrow$ Compare 6 and 2