#### **Question 5**

Input Array Size : N

Number of Processors : P

Constraint : N >> P

#### **Normal Bubble Sort Algorithm**

Bubble-sort (A)

for i = 1 to N **do**

**for** j = N **to** i +1 **do**

**If** A[j] < A[j-1] **then**

**E**xchange A[j] ↔ A[j-1]

##### **Explanation :**

##### The above algorithm simply means that

##### Start from the first element of the array.

##### Compare 2 consecutive elements.

##### If the present element is greater than the element to right : SWAP them.

##### When no swapping is required : elements are sorted.

#### **Parallel Bubble Sort Algorithm**

Bubble Sort has various parallel variants such as :

1. Odd-Even Transposition
2. Cocktail sort

Bubble Sort (A)

**begin**

**for** i = 1 **to** N **do**

**begin**

**if** i **is** odd **then**

**for** j = 0 **to** n/2-1 **do**

**If** A[2i+1] > A[2i+2] **then**

Interchange A[2i+1] ↔ A[2i+2]

**else**

**if** i **is** even **then**

**for** j =1 **to** n/2 -1 **do**

**If** A[2i] > A[2i+1] **then**

Interchange A[2i] ↔ A[2i+1]

**END** **for**

**END**

##### **Explanation:**

##### The idea is processors are grouped int odd/even and even/odd pairs.

##### Odd/even Phase : The odd processes P compare and exchange their elements with the even processors P+1.

##### Even/Odd Phase : The even processes compare and exchange their elements with the odd processors P+1.

##### **Analysis of this Parallel Algorithm**

Both the phases of the algorithm requires O(N) comparisons.

Taking the worst case where all the elements are sorted **in** Descending order **and** we have to sort them **in** ascending order :

8 7 6 5 4 3 2 1

Pass 1a : 7 8 5 6 3 4 1 2

Pass 1b : 7 5 8 3 6 1 4 2

Pass 2a : 5 7 3 8 1 6 2 4

Pass 2b : 5 3 7 1 8 2 6 4

Pass 3a : 3 5 1 7 2 8 4 6

Pass 3b : 3 1 5 2 7 4 8 6

Pass 4a : 1 3 2 5 4 7 6 8

Pass 4b : 1 2 3 4 5 6 7 8 <- SORTED

This means that **is** we have 8 elements **then** we have to **do** 4 complete passes i.e. 4\*2 = 8 passes to sort the elements. So we have to **do** n passes **in** an array **in** the worst case where n **is** the number **of** elements **in** the array.

#### **How is this different from Normal Bubble Sort :**

Bubble sort **is** inherently sequential because every step **of** computation **is** dependent on the result **of** the previous step. By "offsetting" the computation slightly however, we get rid **of** **this** dependency, allowing the sorting **of** adjacent pairs to be carried out **in** parallel - That's the difference between odd-even and bubble sort.﻿