Quick Sort:-

- Quicksort is one of the most common sorting algorithms for sequential computers because of its simplicity, low overhead, and optimal average complexity.
- Quicksort selects one of the entries in the sequence to be the pivot and divides the sequence into two - one with all elements less than the pivot and other greater.
- The process is recursively applied to each of the sublists.

Cont…

- Average optimal sequential complexity: O(n log n)
- Parallel efficiency limitations
  - Partitions are unbalanced
  - A single processor performs the initial partitioning

Example of quicksort

- Let S = (6,5 ,9,2,4,3,5 , 1, 7,5,8).

  T he first call to procedure Q U I C K S O R T produces 5 as the median element of S, and hence

  S1 = {2,4,3,1,5,5} and

  S2 = {6,9,7,8,5}.

  Note that S1 = 6 and S2= 5. A recursive call to Q U I C K S O R T with S, as input produces the two subsequences {2,1,3} and {4,5,5}. The second call with S, as input produces {6,5,7}an d {9,8}. Further recursive calls complete the sorting of these sequences.

Quicksort algo….

procedure QUICKSORT (S)

if $|S| = 2$ **and** $s_2 < s_1$
then $s_1 \leftrightarrow s_2$
else **if** $|S| > 2$ **then**
(1) {Determine m, the median element of S}
SEQUENTIAL SELECT $(S, \lceil |S|/2 \rceil)$

(2) {Split S into two subsequences S_1 and S_2}
(2.1) $S_1 \leftarrow \{s_i : s_i \leqslant m\}$ and $|S_1| = \lceil |S|/2 \rceil$
(2.2) $S_2 \leftarrow \{s_i : s_i \geqslant m\}$ and $|S_2| = \lfloor |S|/2 \rfloor$
(3) QUICKSORT(S_1)
(4) QUICKSORT(S_2)
end if
end if. $\square$

COMPLEXITY OF QUICKSORT

For some constant *c*, ***we can express the running time of procedure***
QUICKSORT as

$$t(n) = cn + 2t(n/2)$$

$$= O(n \log n),$$

1.4 SORTING ON THE CRCW MODEL

- By this algorithm write conflicts problem can be resolved.

- we shall assume that write conflicts are created whenever several processors attempt to write potentially different integers into the same address. The conflict is resolved by storing the sum of these integers in that address.

Cont......

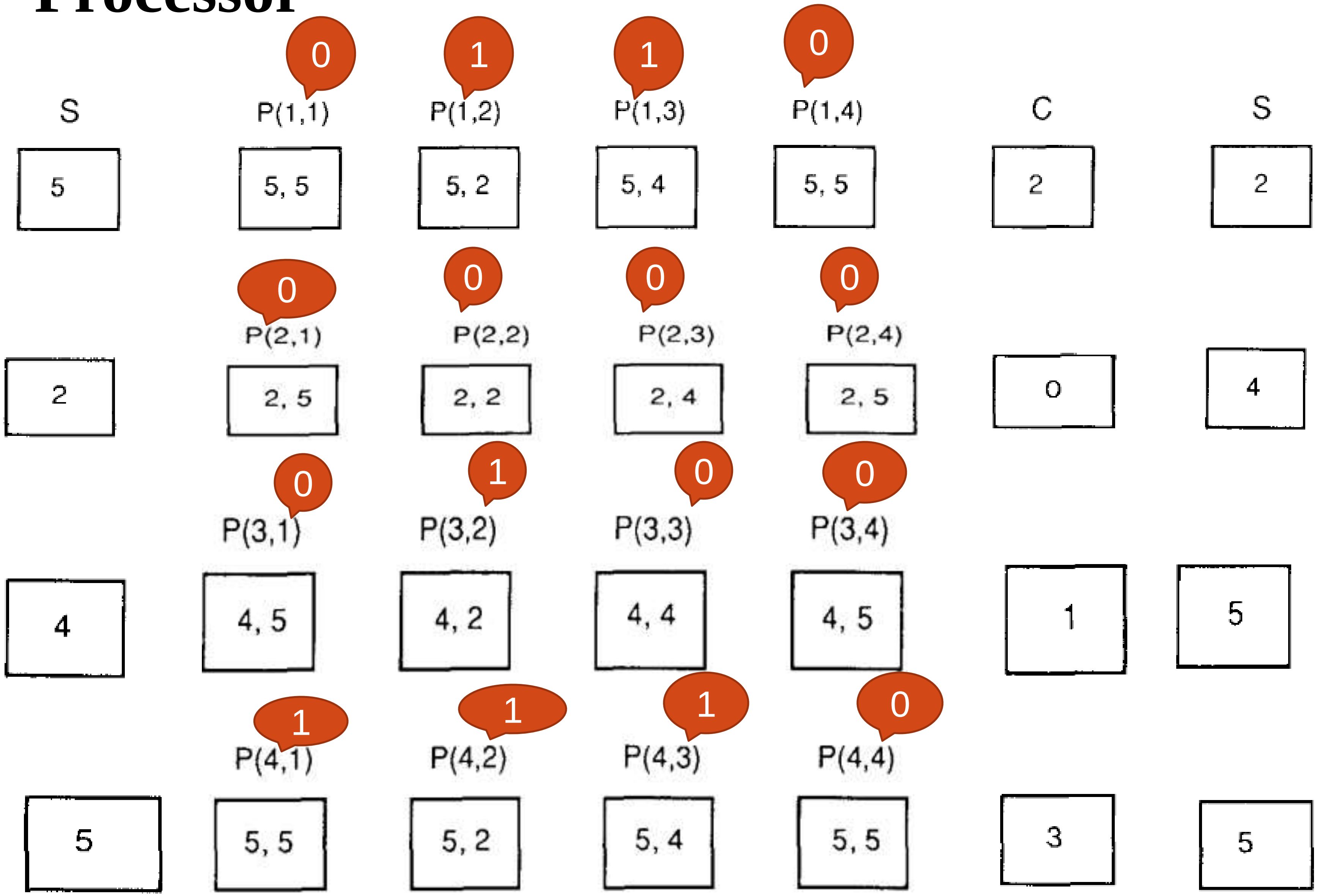
- Assume that n^2 processors are available on such a CRCW computer to sort the sequence

  $S = \{ s\,1 , s2, \ldots , s^n)$.
- If two elements si and sj are equal, then si is taken to be the larger of the two if i > j; otherwise sj is the larger.

Cont....

```
procedure CRCW SORT (S)
Step 1: for i = 1 to n do in parallel
                for j = 1 to n do in parallel
                   if (si > sj) or (si = sj and i > j )
                       then P(i, j) writes 1 in ci
                          else P(i, j ) writes 0 in ci
                       end if
                     end for ---
                  end for.
Step 2: for i = 1 to n do in parallel
P(i, 1 ) stores si in position 1 + ci of S
end for
```

Example: Let S = (5,2,4, 5)   n=4 so n^2 =16 Processor



Cont...

- Update si array
- i: 1+ci position
- 5: 1+2=3
- 2: 1+0=1
- 3:1+1=2
- 4:1+3=4

Cont……

Analysis:- Each of steps 1 and *2* consists of an operation requiring constant time. Therefore

Running Time $t(n) = O(1)$.

- Since $p(n) = n^2$
- The cost of procedure CRCW SORT is:-

$$C(n)= O(n^2) \text{ (which is not optimal)}$$

1.5 SORTING ON THE CREW MODEL

- Our purpose is to design an algorithm that is:

1. free of write conflicts.
2. uses a reasonable number of processors.
3. a running time that is small and adaptive.
4. a cost that is optimal.

- Assume that a CREW SM SIMD computer with N processors PI, P2. . . , *PN* is to be used to sort the sequence S = {s1 s2 . . . , sn), where N < n.

Algorithm:-

procedure CREW SORT *(S)*

Step 1: for i = 1 to N do in parallel

Processor Pi

(1.1) reads a distinct subsequence Si of S of size n/N

(1.2) QUICKSORT (Si)

(1.3) S_i^1 <- Si

(1.4) P_i^1 <- Pi

end for.

O((n/N)log(n/N))

Cont…

Step 2 (2.1) u =1

(2.2) v = N

(2.3) while v > 1 do

(2.3.1) for m = 1 to |_v/2_| do in parallel

(i) P^{u+1}_{m} <- $P^{u}_{2m-1} \cup p^{u}_{2m}$

(ii) The processors in the set P^{u+1}_{m} perform

CREW MERGE (s^{u}_{2m-1}, s^{u}_{2m}, s^{u+1}_{m})

end for

(2.3.2) if v is odd then

(1) $p^{U+1}_{v/2} = p^{u}_{v}$

(ii) $s^{U+1}_{v/2} = s^{U}_{V}$

end if

(2.3.3) u = u + 1

(2.3.4) V = v/2

end while.

O((n/N) + log n) time

Example

- Let S = (2, 8, 5, 10, 15, 1, 12, 6, 14, 3, 11, 7, 9, 4, 13, 16) and N = 4. Here $N<n$

Step1:- Subsequence S_i created : n/N=>16/4= 4

And Quick sort apply for sorting elements

$S_1^1 = \{2,5,8,10\}$ $S_2^1 = \{1,6,12,15\}$

$S_3^1 = \{3,7,11,14\}$ $S_4^1 = \{9,13,14,16\}$

Step2:- u=1 & v=N=4

CREW MERGE ALGO USED

for (m=1 to v/2) 4/2=2

$P_1^2 = p_1^1 \cup p_2^1$ =(p1,p2)=(1,2,5,6,8,10,12,15)

$P_2^2 = p_3^1 \cup p_4^1$ =(p3,p4)=(3,4,7,9,11,13,14,16)

Cont....

The processors {P1, P2,P3, P4} cooperate to merge S_1^2 and s_2^2 into S_1^3 = (1, 2,. . . , 16) by using **CERW MERGE** .

Analysis:- the total running time of procedure CREW SOR'T is

t(n) = O((n/N)log(n/N)) + O((n/N)log N + log n log N)

= O((n/N)log n + log2n).

- Since p(n) = N, the cost is given by:-

  c(n) = O(n log n + N log n^2).

1.6 SORTING ON THE EREW MODEL:-

- Still, procedure CREW SORT tolerates multiple-read operations. Our purpose in this section is to deal with this third difficulty.

- We assume throughout this section that N processors P1, P2 . . . , PN are available on an EREW SM SIMD computer to sort the sequence S = (s1, s2, . . . , sn)where N < n.

Cont….

- since N < n, $N=n^{1-x}$ where $0<x<1$.
- Now $mi =[i(n/2^{1/x})]$, for $1<=i<=2^{1/x-1}$.
- The mi can be used to divide S into $2^{1/x}$ subsequence of size $n/2^{1/x}$.
- These subsequences, denoted by S1,S2,..., Sj, Sj+1,........S2j, where $j =2^{(1/x)-1}$
- Every subdivision process can now be applied recursively to each of the subsequences Si until the entire sequence S is sorted in nondecreasing order.
- $K= 2^{(1/x)}$

Algorithm:-

```
procedure EREW SORT (S)
Step1  if |S| < k
             then QUICKSORT (S)
         else (1) for i = 1 to k - 1 do
                          PARALLEL SELECT (S, |i |s|/k|)      [obtain mi]
                    end for
              (2)  Si  = (s E S: s<=mi  )
              (3)  for i = 2 to k - 1 do
                          Si  ={s E S : mi-1<=s <=mi }
                     end for
```

Cont..

(4) S_k<= { s E S : s >=m_{k-1})

Step 2 for i = 1 to k/2 do in parallel

EREW SORT (Si)

end for

Step 3 for i = (k/2) + 1 to k do in parallel

EREW SORT (Si)

end for

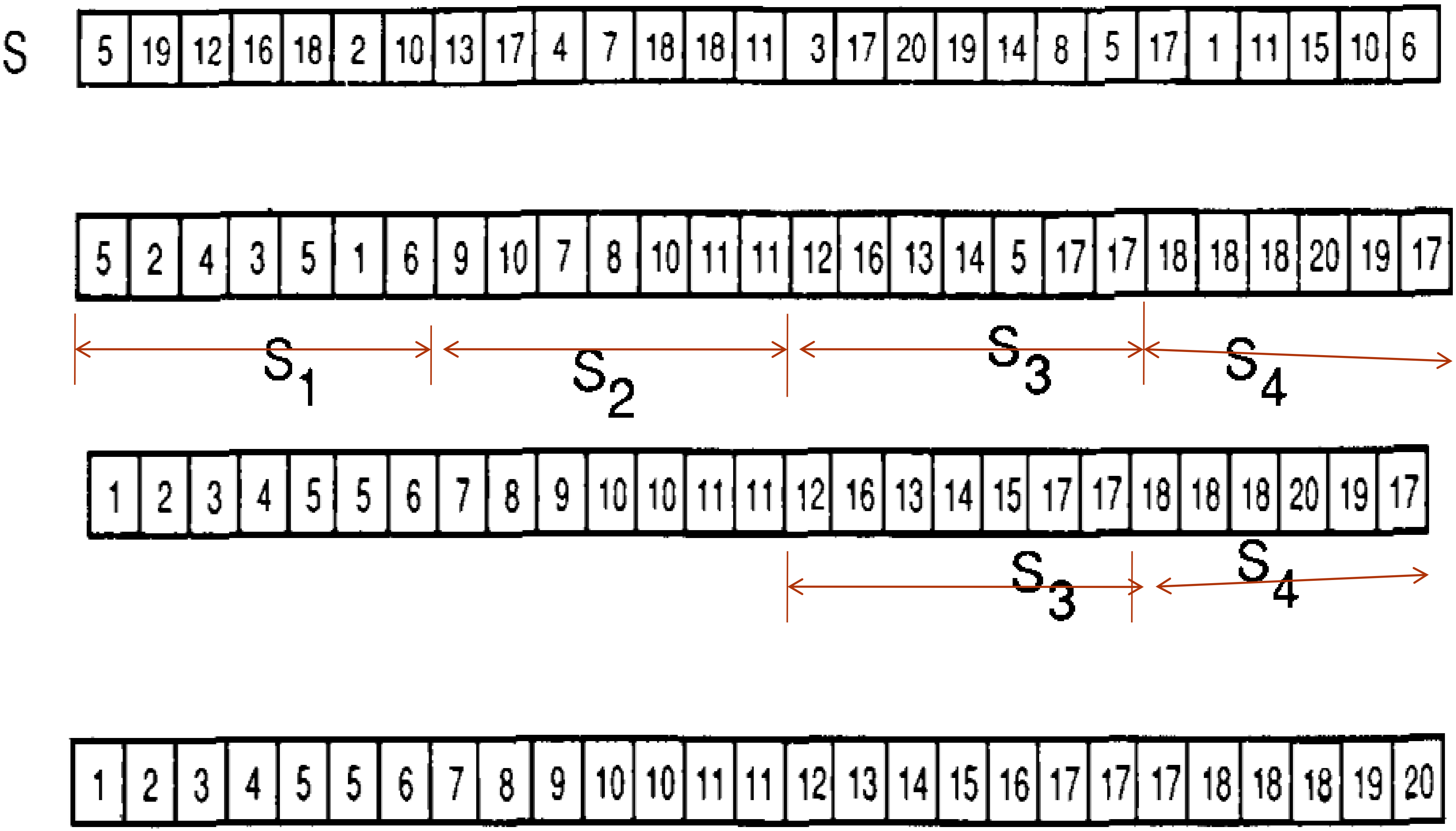
end if.

Cont...

Let S = {5,9, 12, 16, 18,2, 10, 13, 17,4,7, 18, 18, 11, 3, 17,20,19, 14, 8, 5, 17, 1, 11, 15, 10, 6) (i.e., n = 27)

- Here $N<n$ & $N=n^{1-x}$ => $N=27^{0.5} = 5$ **where $0<x<1$** ($x=0.5$).
- $K=2^{1/x}$ => $k= 2^{1/0.5} = 2^2 = 4$
- During step 1 m1= 6 m2 = 11, and m3 = 17 are computed.
- The four sub sequences S1 ,S2, S3 and S4 are created.
- In step 5 the procedure is applied recursively and simultaneously to S1 and S2.
- Compute m1 = 2, m2= 4, and m3= 5, and the four subsequence {1,2}, {3,4}, {5,5), and (6) are created each of which is already in sorted order.

Cont....


S
5 19 12 16 18 2 10 13 17 4 7 18 18 11 3 17 20 19 14 8 5 17 1 11 15 10 6
5 2 4 3 5 1 6 9 10 7 8 10 11 11 12 16 13 14 5 17 17 18 18 18 20 19 17
S1
S2
S3
S4
1 2 3 4 5 5 6 7 8 9 10 10 11 11 12 16 13 14 15 17 17 18 18 18 20 19 17
S3
S4
1 2 3 4 5 5 6 7 8 9 10 10 11 11 12 13 14 15 16 17 17 17 18 18 18 19 20

Cont....

- Running Time $t(n) = cn^x + 2t(n/k)$

  $= O(n^x \log n)$.

- Since $p(n) = \boldsymbol{n^{1-x}}$**,** the procedure's cost is given by

  $c(n) = p(n) \times t(n) = O(n \log n)$,

which is optimal.