PRAM Algorithms

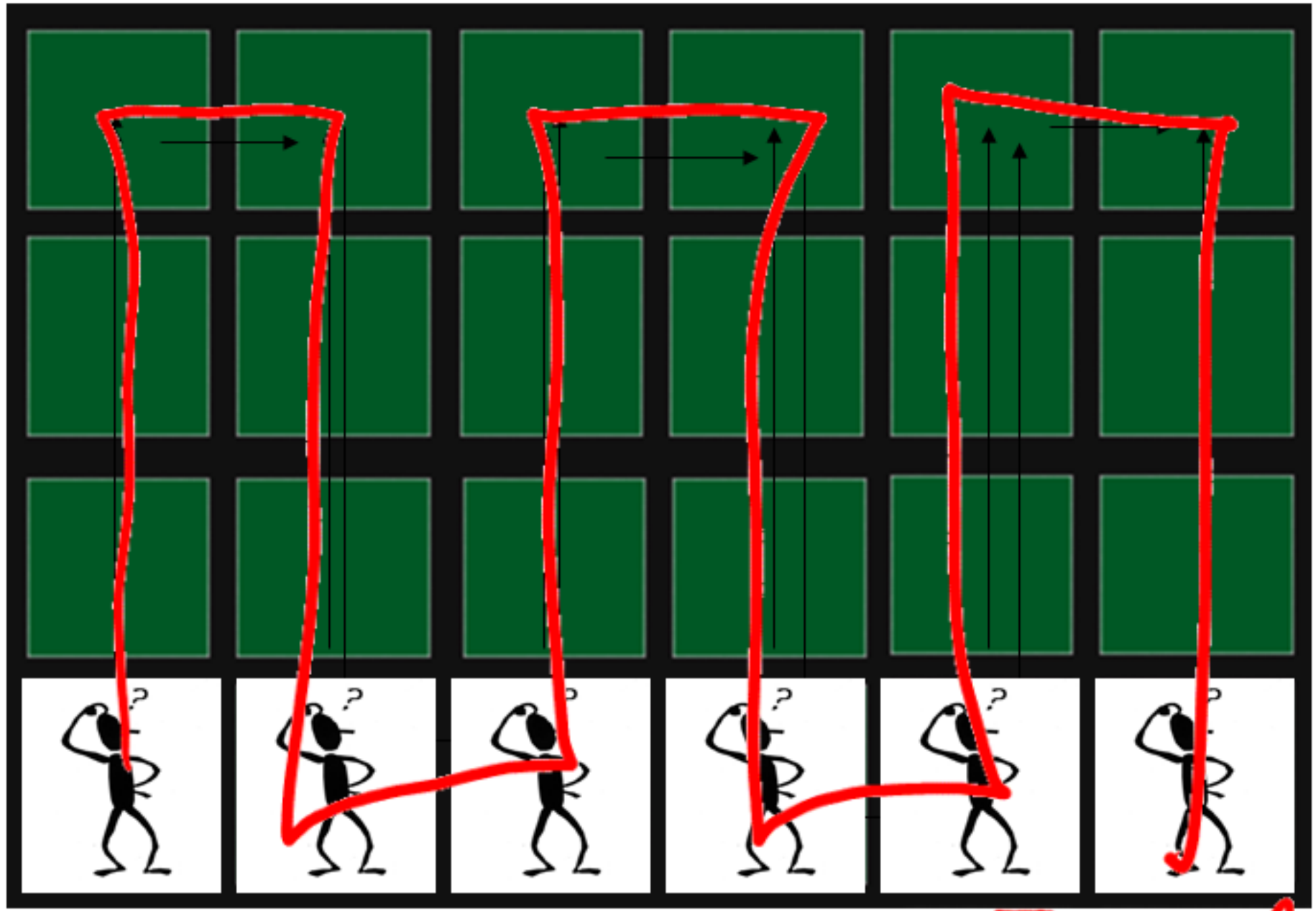
Lecture Objectives:

1) Define the concept of a PRAM model of computation.
2) Explain how the preorder tree traversal problem can be parallelized
3) Explain how to perform a merge sort in parallel
4) Explain how using a matrix of relationships, it is possible to sort an array of n elements in O(1) operations

What is a Parallel Algorithm?

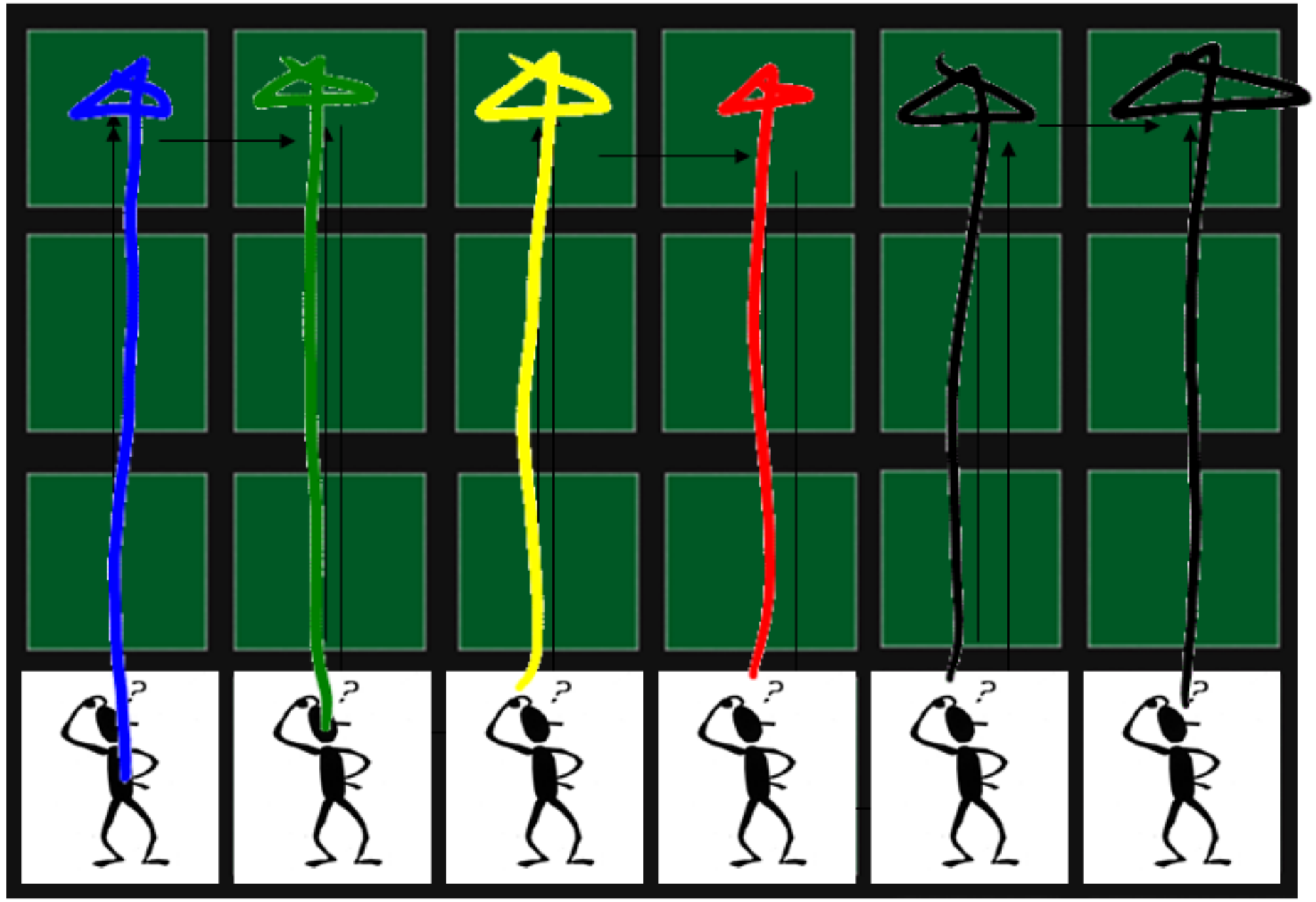
- Imagine you needed to find a lost child in the woods.
- Even in a small area, searching by yourself would be very time consuming
- Now if you gathered some friends and family to help you, you could cover the woods in much faster manner…

Sherwood Forest



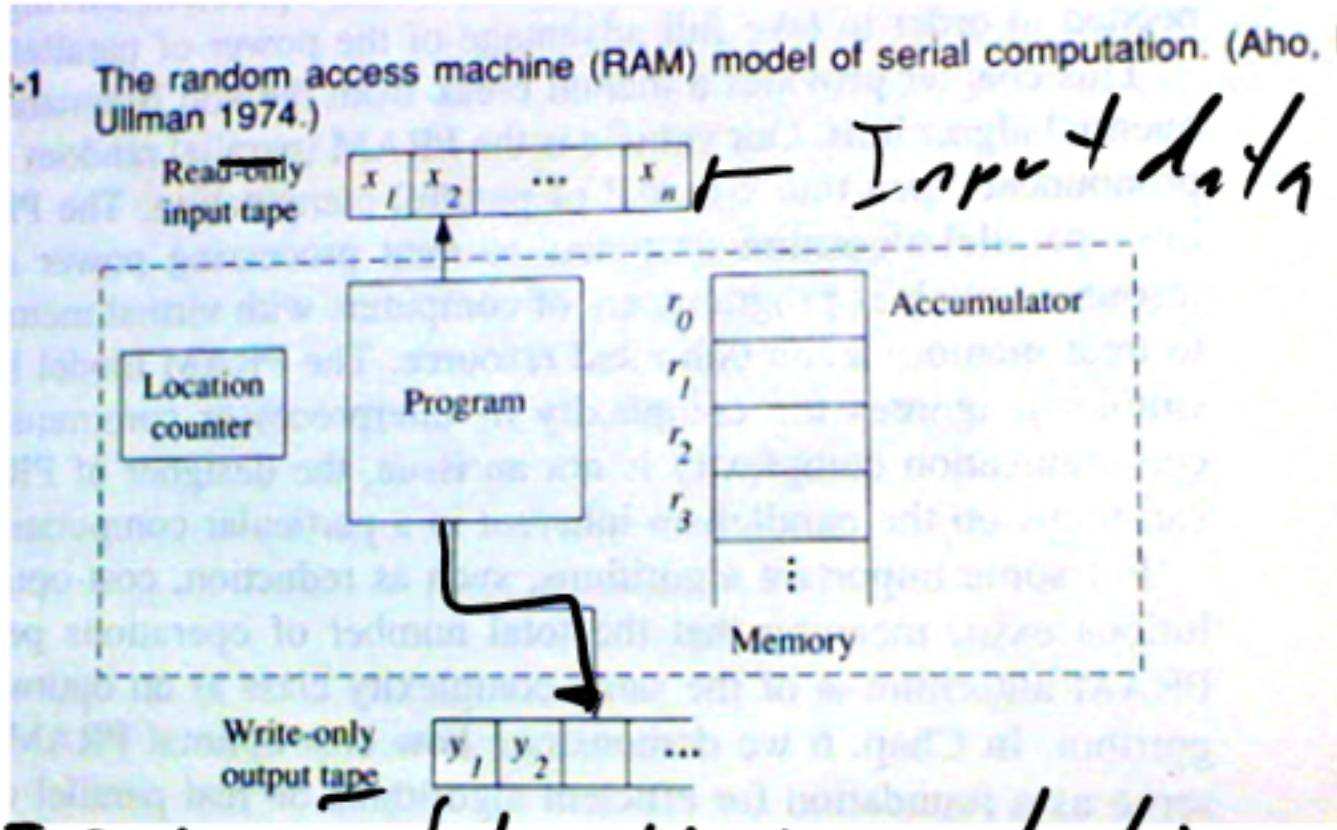
found!

Sherwood Forest



Quicker!

The RAM Model of serial computation



-1 The random access machine (RAM) model of serial computation. (Aho, Ullman 1974.)

← Input data

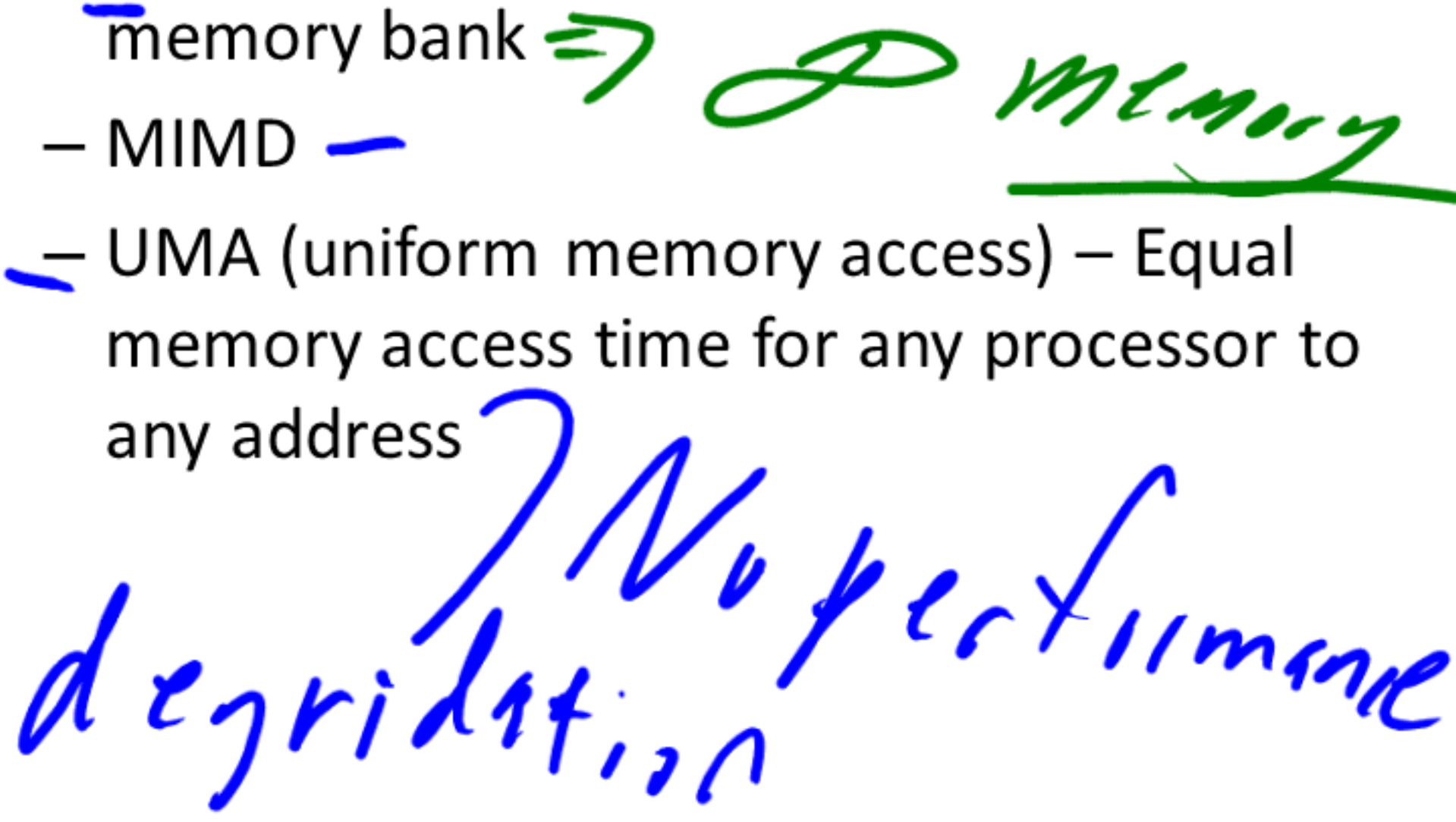
If the input data is of length n, what is the worst case time complexity? $f(n)$

MSOE

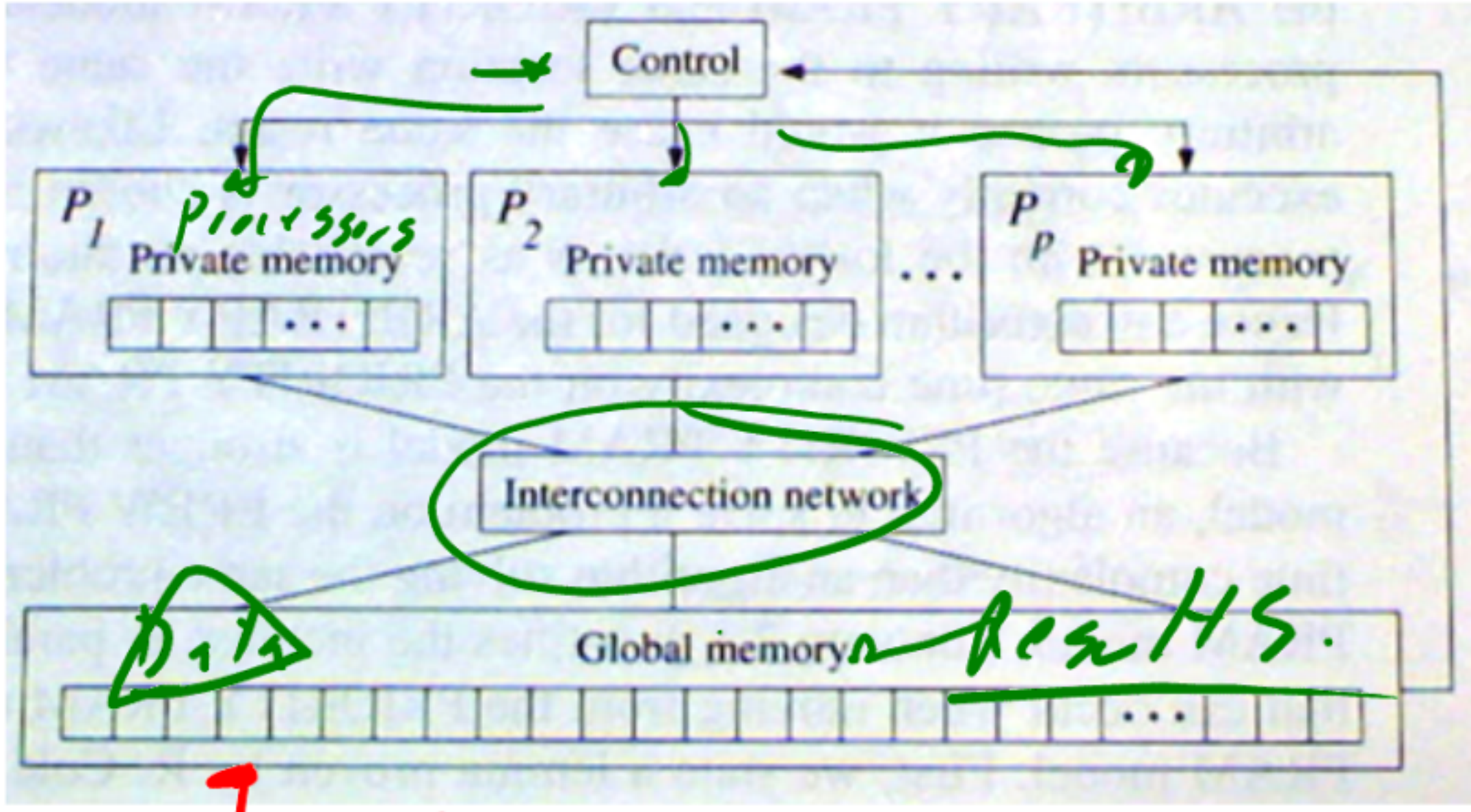
the ~~average~~ expected time complexity is the average on the executions of size n.

The PRAM Model

- **P**arallel **R**andom **A**ccess **M**achine
  - Theoretical model for parallel machines
  - p processors with uniform access to a large memory bank
  - MIMD
  - UMA (uniform memory access) – Equal memory access time for any processor to any address

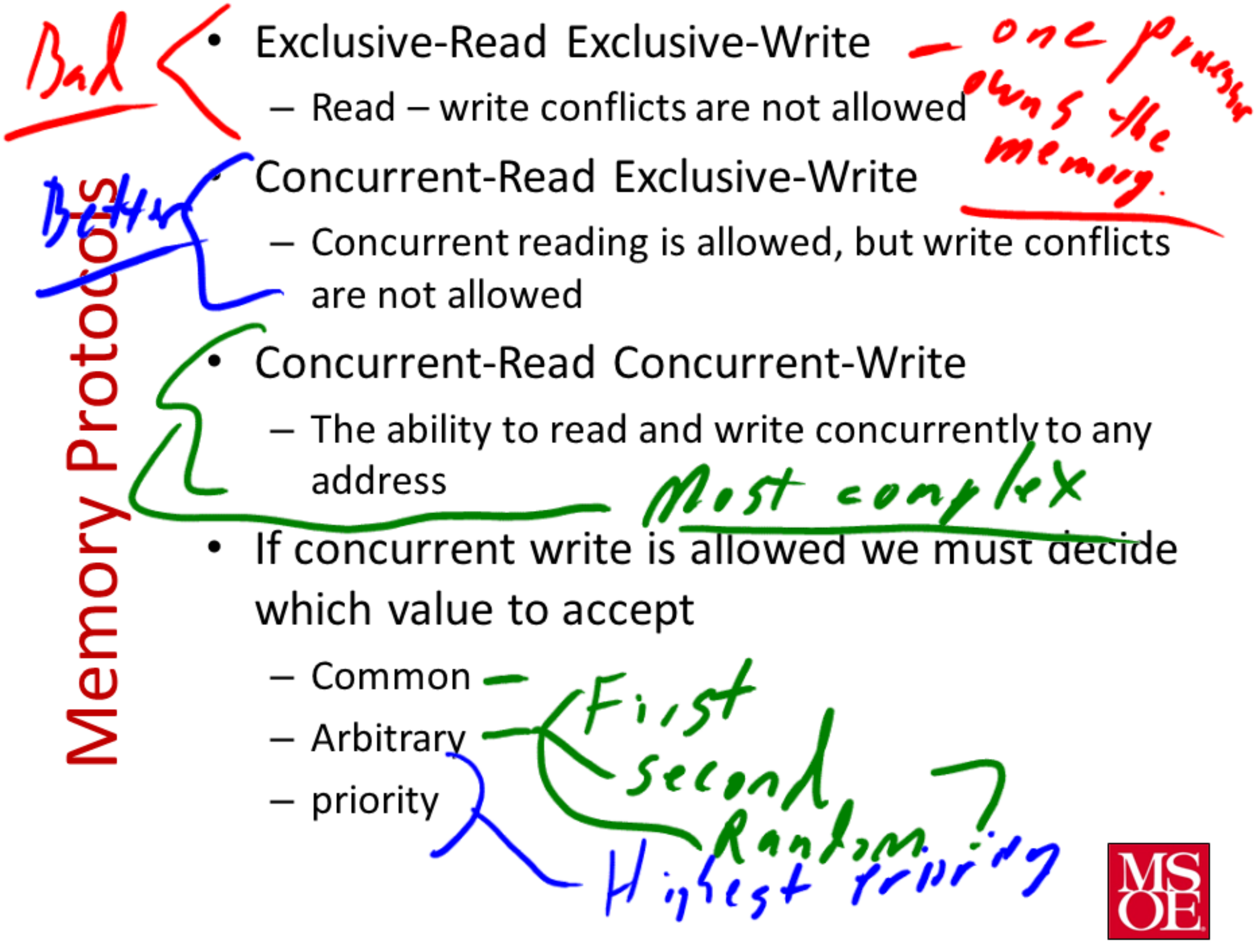


The PRAM Model



Memory Protocols

- Exclusive-Read Exclusive-Write
  - Read – write conflicts are not allowed
- Concurrent-Read Exclusive-Write
  - Concurrent reading is allowed, but write conflicts are not allowed
- Concurrent-Read Concurrent-Write
  - The ability to read and write concurrently to any address
- If concurrent write is allowed we must decide which value to accept
  - Common
  - Arbitrary
  - priority



Example: Finding the Largest key in an array

- We must do n - 1 comparisons.
- Parallel version will do this in log(n) time.

Example: Finding the Largest key in an array

- Assume that n is a power of 2 and we have n / 2 processors executing the algorithm in parallel.
- Each Processor reads two array elements into local variables called *first* and *second*
- It then writes the larger value into the first of the array slots that it has to read.
- Takes lg n steps for the largest key to be placed in S[1]

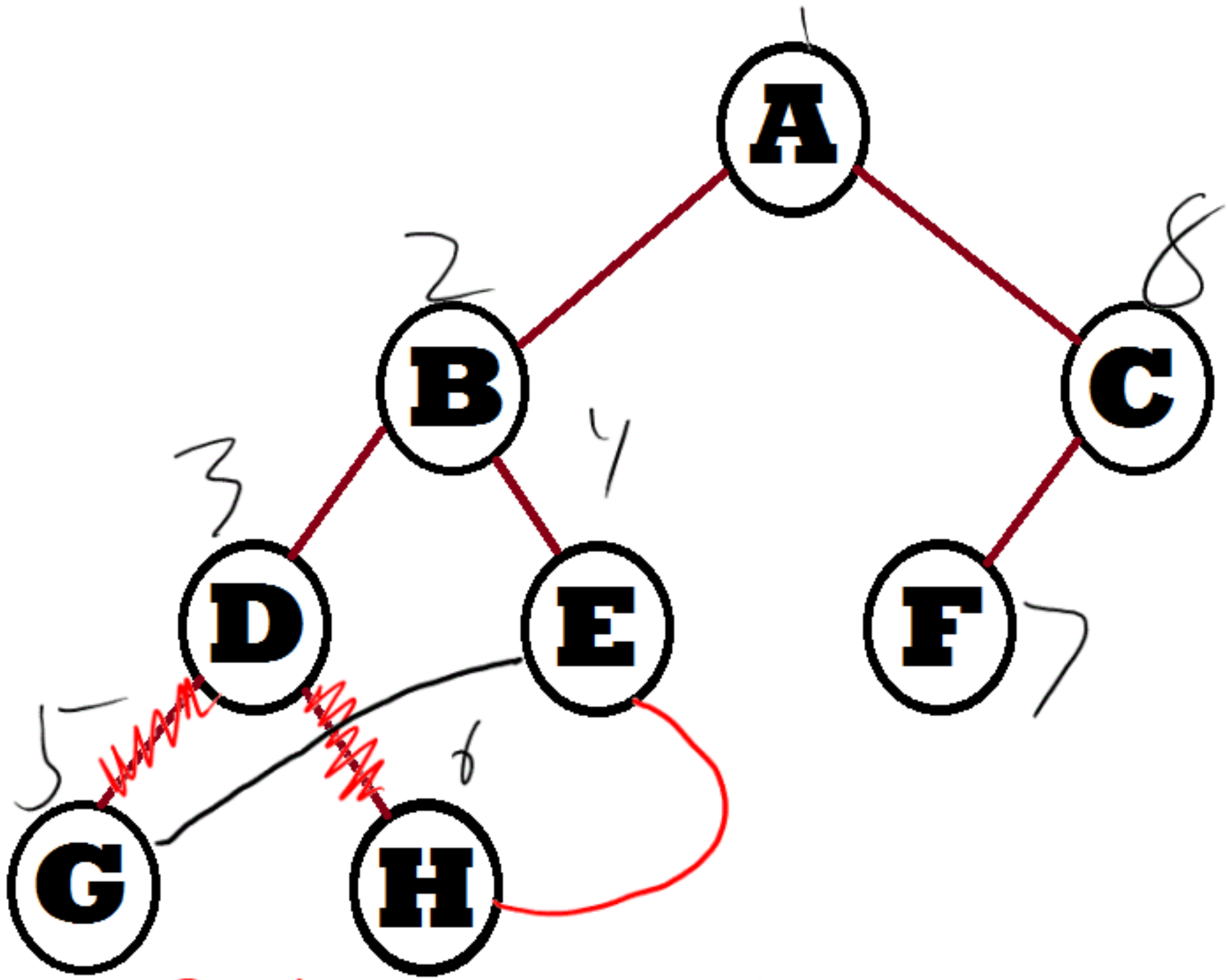
Example

Largest

3	1	7	6	8	1	1	1

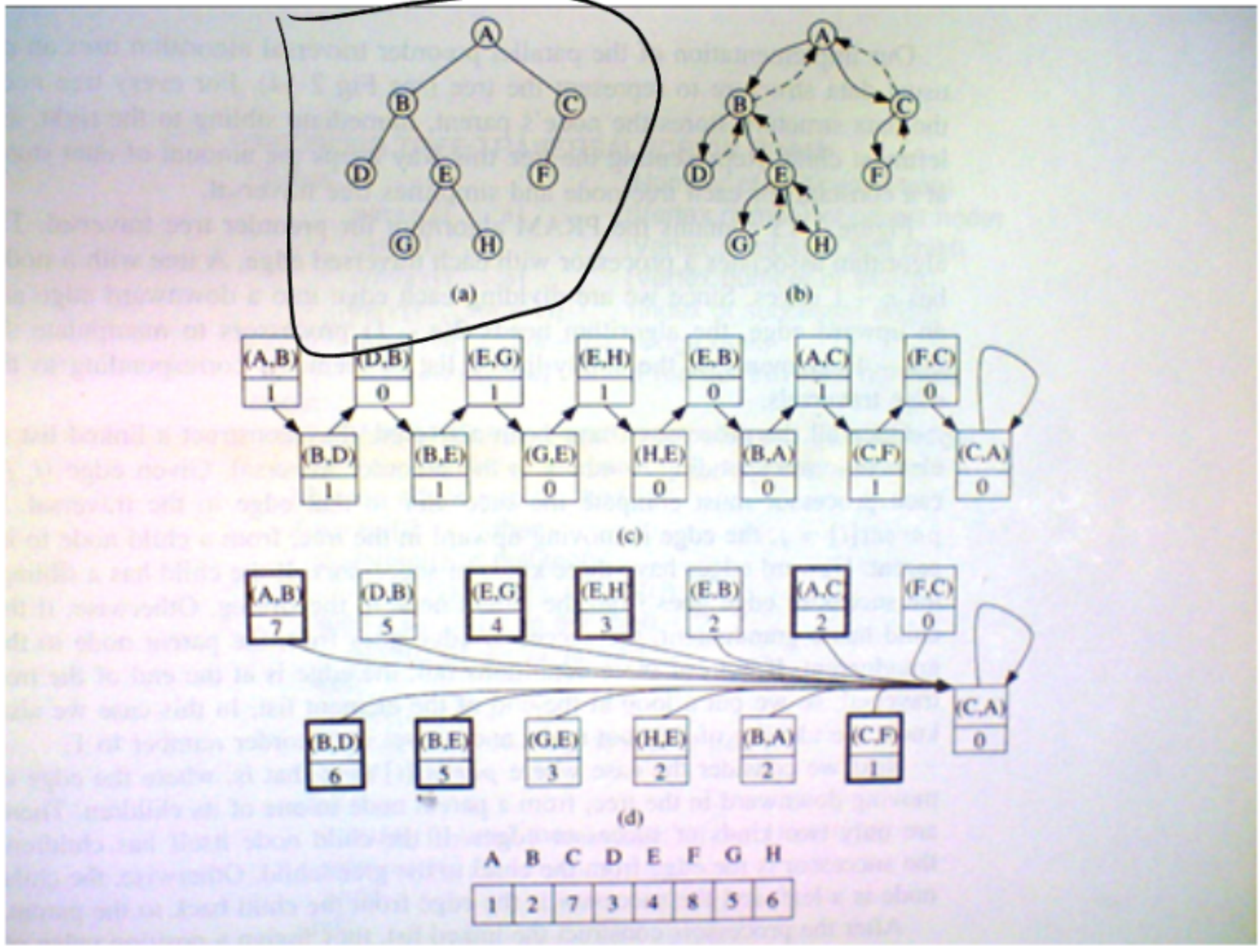
$\log(n)$

Preorder Tree Traversal



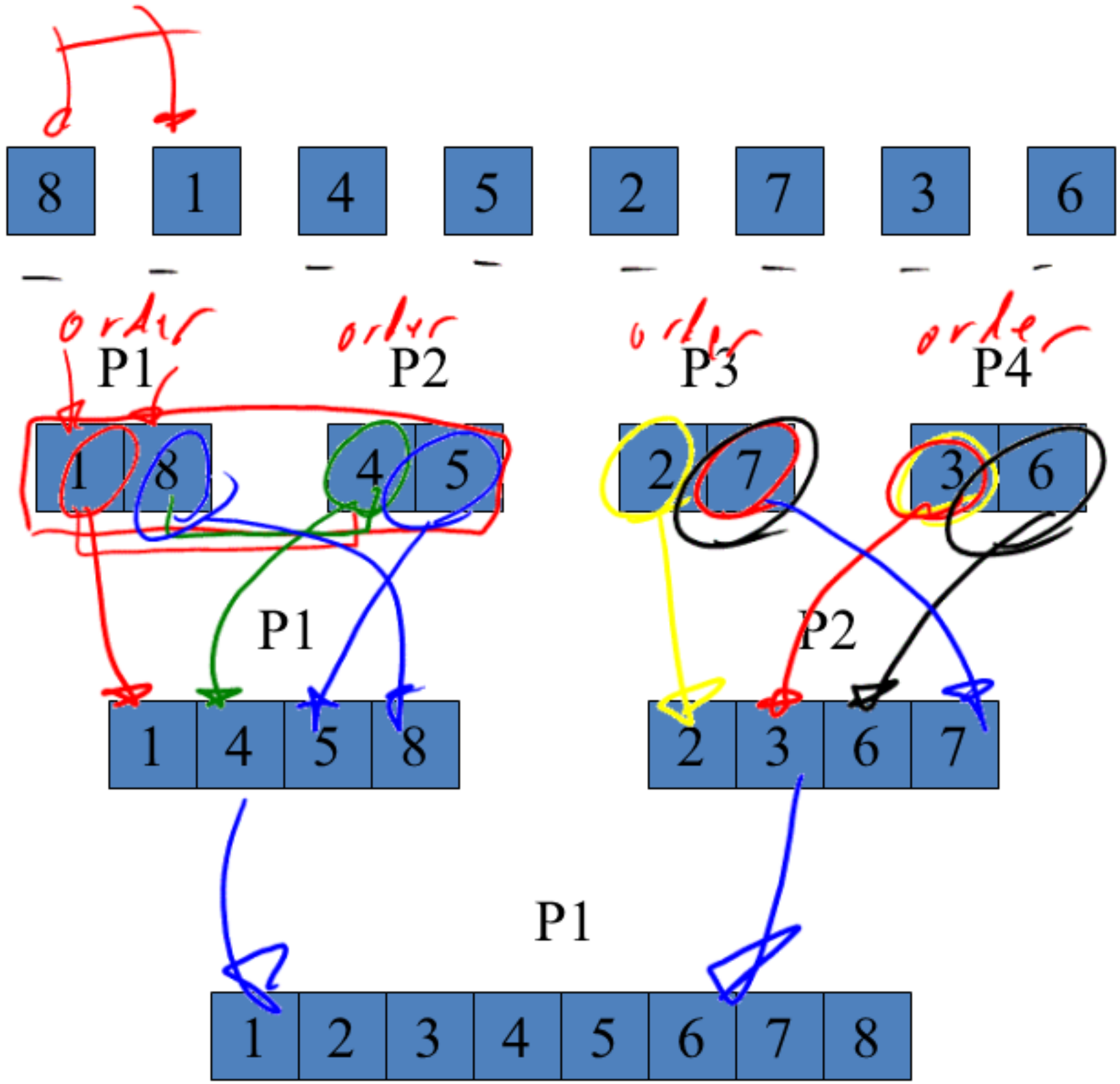
G H D E B F C A

Parallel ALgorithm

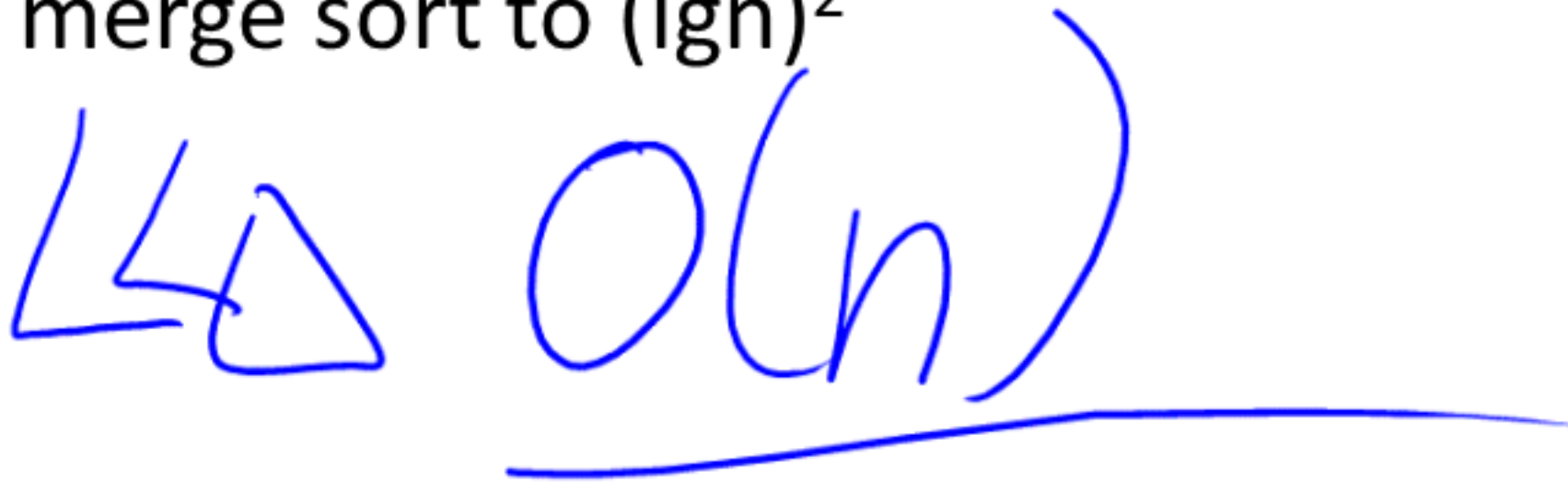


Merge sort in parallel

Example: Merge Sort



- Number of compares
  - $1 + 3 + \ldots + (2^i-1) + \ldots + (n-1)$
  - $\sum_{i=1..lg(n)} 2^i-1 = 2n-2-lgn = \Theta(n)$
- We have improved from nlg(n) to n simply by applying the old algorithm to parallel computing, by altering the algorithm we can further improve merge sort to $(lgn)^2$



O(1) Sorting Algorithm

We assume a CRCW PRAM where concurrent write is handled with addition

```
for(int i=1; i<=n; i++)
{
  for(int j=1; j<=n; j++)
  {
    if(X[i] > X[j])
      Processor Pij stores 1 in memory location mi
    else
      Processor Pij stores 0 in memory location mi
  }
}
```

n x i processors

Each core does this.

an example

0	1	2	3	4	5	6	7
10	4	6	8	12	2	0	16

I,j	0	1	2	3	4	5	6	7
0	0	0	0	0	1	0	0	1
1	1	0	1	1	1	0	0	1
2	1	0	0	1	1	0	0	1
3	1	0	0	0	1	0	0	1
4	0	0	0	0	0	0	0	1
5	1	1	1	1	1	0	0	1
6	1	1	1	1	1	1	0	1
7	0	0	0	0	0	0	0	0
Total	5	2	3	4	6	1	0	7

Lets look at

0 2 4 6 8 10 12 16

Parallel Algorithm

- The Fast Parallel Algorithm
  - Construct a singly linked list of the edges and direction
  - Assign weights to the edges of the newly created linked list
    - 1 for downward
    - 0 for upward
  - Compute the rank of that element
  - Assign the nodes order in the preorder tree traversal