02 Jan 2025 - Algorithms - Week 01

Online notes Jeff Ericson

Methods for efficiently solving computational problems

* Searching query in a PDF → abstraction → T[1] ... T[k] S[1] ... S[l]
  Find i s.t
  T[i] T[i+1] ... T[ ] = S[1]

* Map, querying to get nearest point → abstraction

* travel booking → optimize time / optimize cost

Algorithmic techniques to solve problems

- existing algorithms (C, C++ implementation)
- general techniques

Basic strategy: Induction

i/p : $a_1, a_2, \ldots, a_n$

o/p : $f(a_1, \ldots, a_n)$

Read a_1 $\xrightarrow{\text{compute}}$ $f(a_1)$

Read a_2 $\xrightarrow{\text{compute}}$ $f(a_1, a_2)$

⋮ ⋮

Examples :

1. Finding $\max(a_1, \ldots, a_n)$
2. Finding average
3. Finding # of occurences of a given element.
4. sorting (insertion sort)

Divide and conquer

A. Divide input into 2 or more pieces.

B. Solve on each piece → Recursive

C. combine solutions.

Finding the maximum : $a_1, a_2, \ldots, a_n$

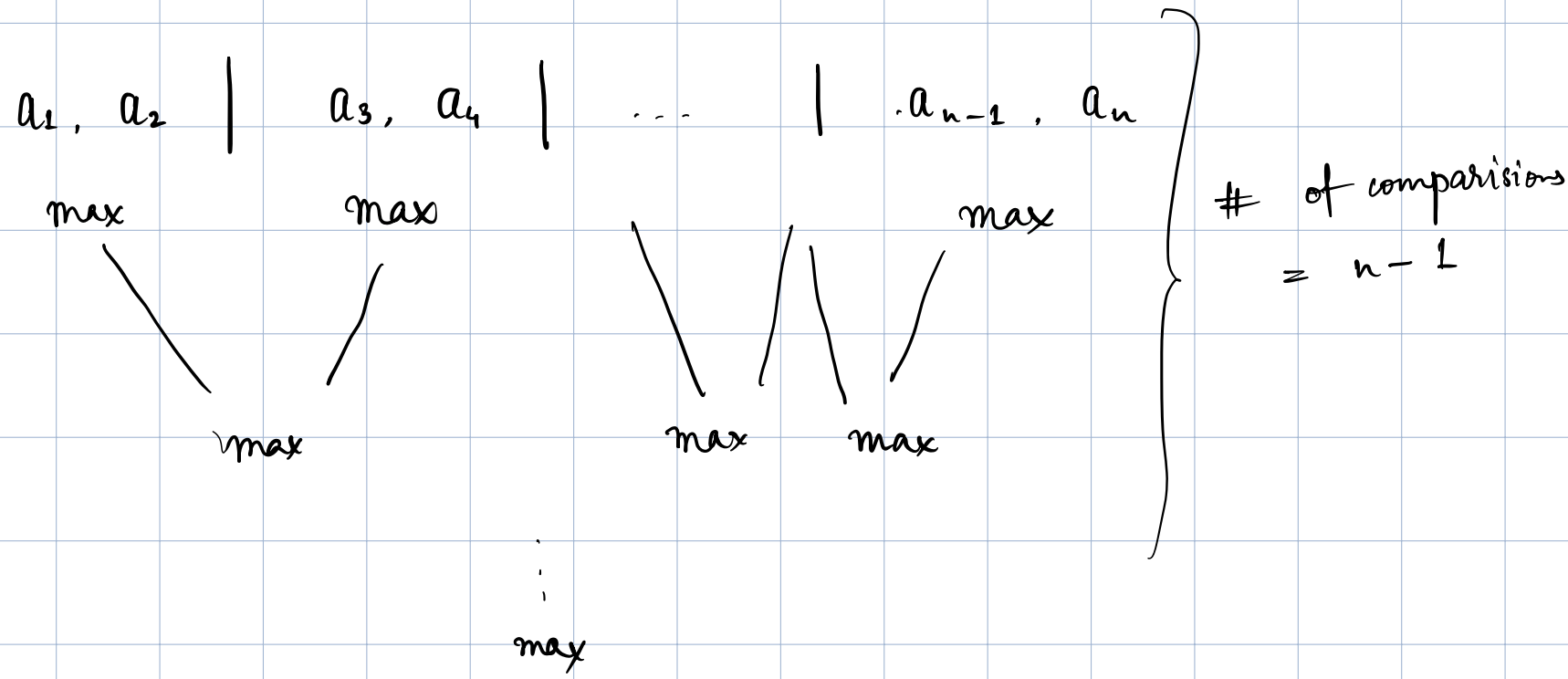
comparisions = $n - 1$

$a_1, a_2, \ldots, a_{\lfloor n/2 \rfloor}$ | $a_{\lfloor n/2 \rfloor + 1}, \ldots, a_n$

$\max = x$ | $\max = y$

return $\max(x, y)$

# of comparisions = $n - 1$



# of comparisions $= n - 1$

always, no matter how recursion is performed

every comparision removes only one element

$T(n)$ = # time steps on an input of size n.

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$$

$$T(1) = 0$$

$$T(2) = 1$$

$$T(3) = T(1) + T(2) + 1$$

$$= 2$$

$$T(4) = T(2) + T(2) + 1$$

$$= 3$$

Finding maximum and minimum of n elements

Induction: $2(n-1)$ comparisions

$a_1, \ldots, a_{\lfloor \frac{n}{2} \rfloor} \mid a_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, a_n$

$x \leq X$ $\qquad$ $y \leq Y$

$\min(x, y)$

$\max(X, Y)$

$T(n) =$

$T(1) = 1$

$T(2) = 1$

$T(3) = 1 + 1 + 1$

$T(4)$

1 | 2 3

1 (2, 3)

1, 2 | 3, 4

(1, 2) (3, 4)

If n is a power of 2

$T(n) = 2T\left(\frac{n}{2}\right) + \sim$

$T(n) = 2T\left(\frac{n}{2}\right) +$

$T(1) = 1$

$T(2) = 1$

$T(4) = 3$

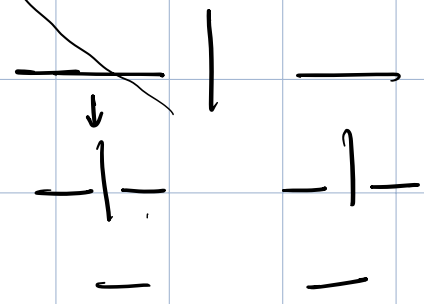
$T(n) = 2T\left(\frac{n}{2}\right) + 1$

$T\left(\frac{n}{2}\right)$

$\frac{}{1} \mid \frac{}{1}$

1

$T(8)$



$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 2$$

$$T(1) = 0$$
$$T(2) = 1$$
$$T(4) = 4$$
$$T(8) = 10$$
$$T(16) = 22$$

$T(n) = an + b$

$a(4) + b = 4$

$a(8) + b = 10$

$4a = 6$

$a = 3/2$

$b = -2$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + c$$

$\Big\downarrow$

solution always

$T(n) = an + b$

$$an + b = an + 2b + c$$

$$b = -c$$

Recap

* Insertion sort

  correctness and efficiency

* time complexity

* **RAM model**

operations that take constant time

find max (A, first, last)

mid = $\frac{f + l}{2}$

x = find max (A, f, m)

y = find max (A, m+1, l)

return max(x, y)

Puzzle : Find max and second - max in $\leq n + \log_2 n + 2$ comparisions

Sorting problems

I/p : $a_1, a_2, \ldots, a_n$

O/p : $b_1 \leq \ldots \leq b_n$

permutation of i/p

Insertion sort : $O(n^2)$

Bubble sort : $O(n^2)$

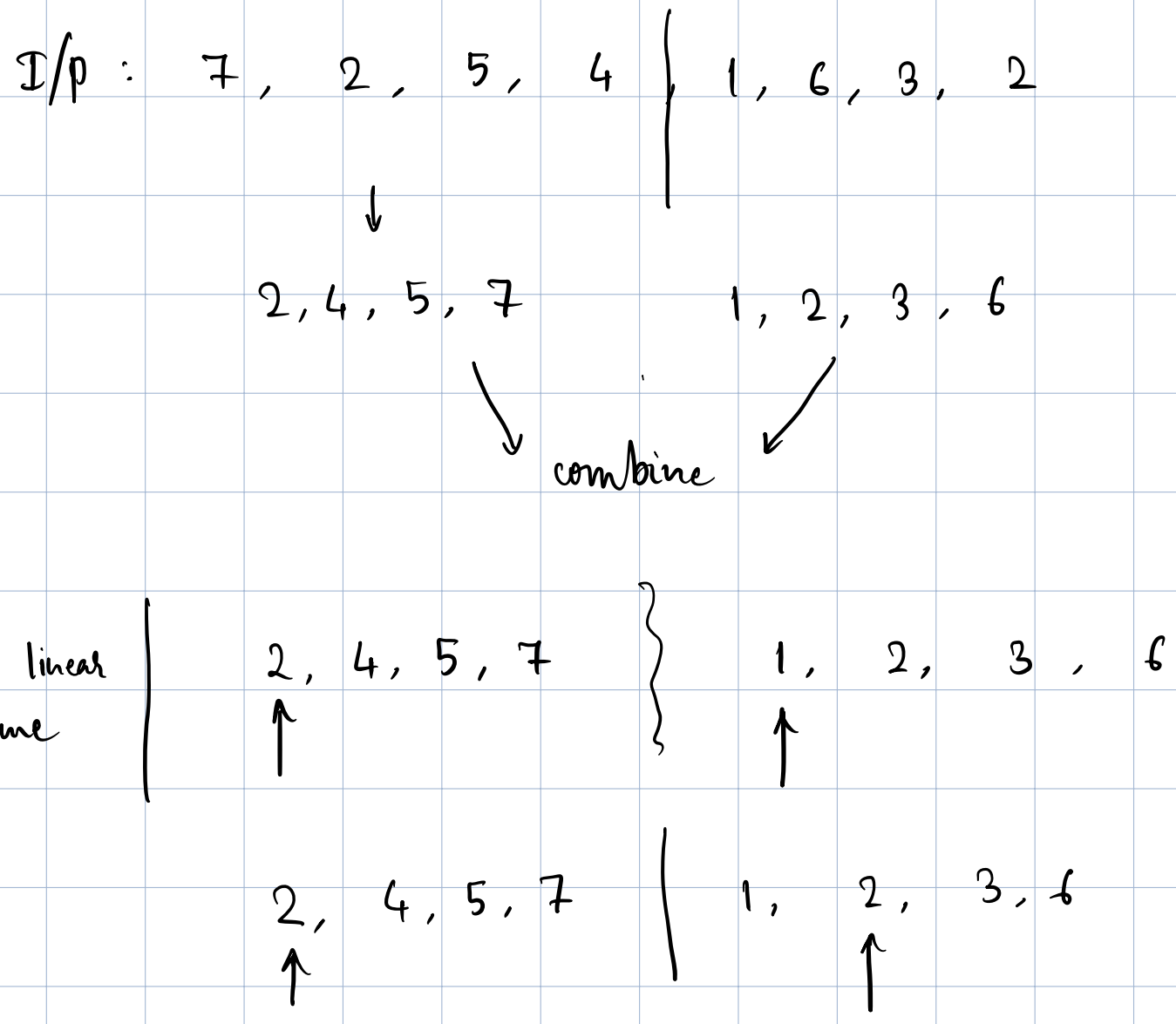
Heapsort : $O(n \log n)$

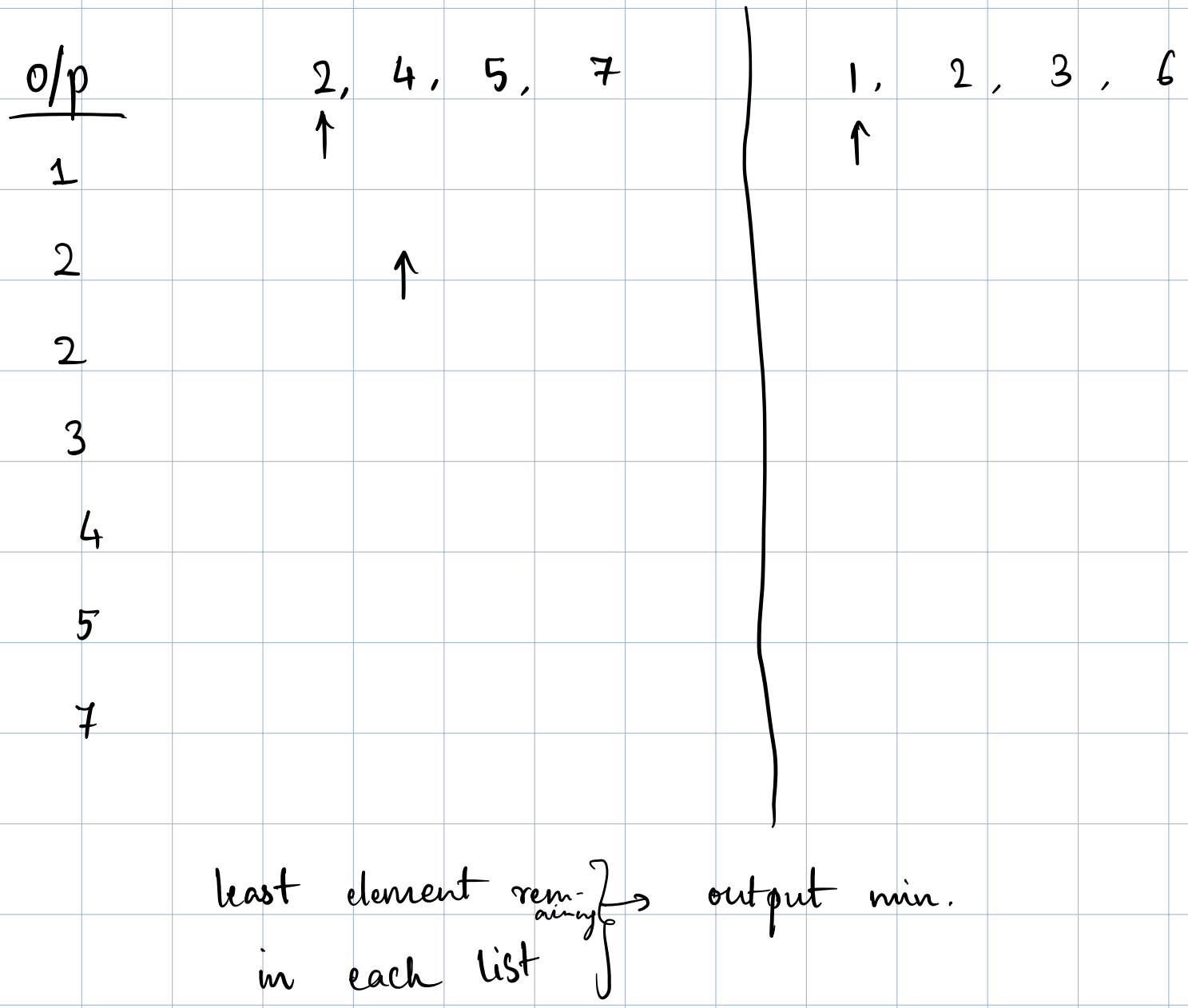
Merge sort : $O(n \log n)$ → divide and conquer

comparisions = 1 + 1

$$T(n) = T\left(\lfloor n/2 \rfloor\right) + T\left(\lceil n/2 \rceil\right) + 2$$

$$T(n) = \frac{3n}{2} - 2$$


I/p : 7, 2, 5, 4 | 1, 6, 3, 2
2, 4, 5, 7
1, 2, 3, 6
combine
in linear time
2, 4, 5, 7
1, 2, 3, 6
2, 4, 5, 7
1, 2, 3, 6


o/p
1
2
2
3
4
5
7
2, 4, 5, 7
1, 2, 3, 6
least element remaining in each list
output min.

$A[1, 2, \ldots, m]$ - sorted

$B[1, 2, \ldots, n]$ - sorted

We can find $A \cup B$ in sorted order in $O(m+n)$ time

$O(m+n)$: one element is output in each step.

$1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$ → merge sort

$\lfloor \frac{n}{2} \rfloor + 1, \ldots, n$ → merge sort

combine the 2 sorted lists

merging 2 sorted lists of size $\frac{n}{2}$ each

$$T(n) = T\left(\lfloor \tfrac{n}{2} \rfloor\right) + T\left(\lceil \tfrac{n}{2} \rceil\right) + O(n)$$

power of 2 assumption can be justified

2^n $2n$