Algorithm Analysis and Design

Follow up from DSA.

- No coding component in this course.
- No assignment submissions.
- Build mathematical perspective to algorithms.

Books:

- "Algorithm design" by Kleinberg and Tardos.
- "Algorithms" by Dasgupta, Papadimitriou and Vazirani.

Grading scheme:

Quiz 1	15
Quiz 2	15
Midsem	30
Endsem	40

At least 4 problem sets.

- Every evaluation consists of 30-35% of questions from* problem sets.

* - related questions

Syllabus:

1. Intro to algorithm design
2. Graph algorithms
3. Greedy algorithms
4. Divide and Conquer
5. Dynamic programming
6. Network flows
7. Computational hardness.

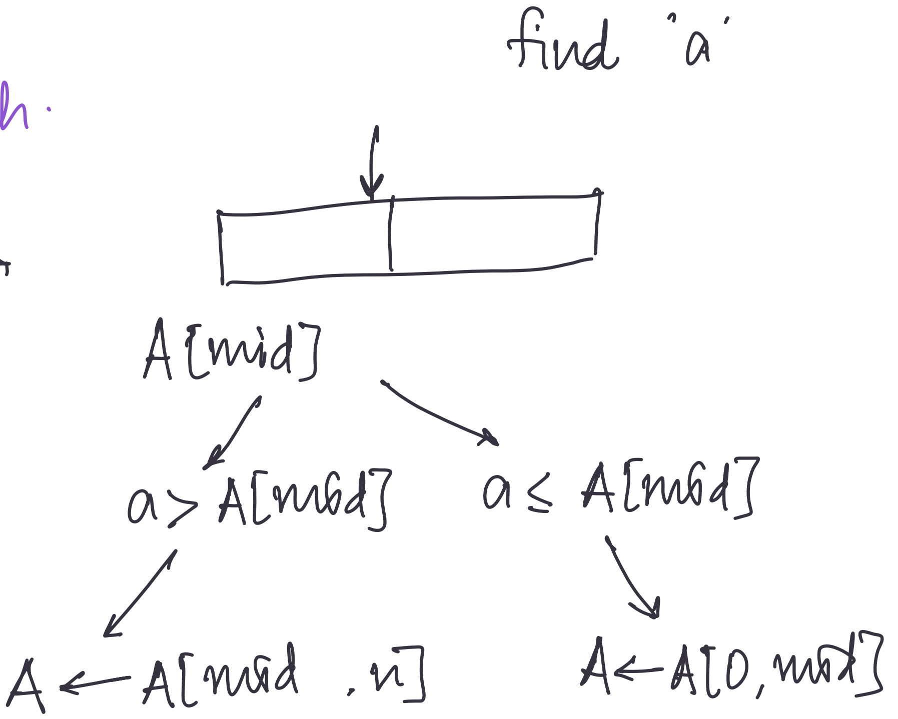
$$x \longrightarrow \boxed{A} \longrightarrow A(x)$$

Resources: Time Space.

Ex: Binary search.

Sorted array →

≤ log n comparisons.



Matrix multiplication:

Each entry of A and B have at most l-bit repr.

$$C = \begin{bmatrix} A \end{bmatrix}_{n\times n} \begin{bmatrix} B \end{bmatrix}_{n\times n}$$

$$C_{ij} = \sum_k A_{ik} \cdot B_{kj}$$

n^2 entries

↳ n multiplications

n additions.

$$\Bigg\} \le n^2(2n) = 2n^3.$$

[Strassen] $c \cdot n^{\log_2 7}$

→ $\underbrace{c \cdot n^{\omega}}_{\text{\# of mult.}}$ $\omega \sim 2.3\ldots$ [Alman, Vassileska Williams]

$O(\ell^2) \leftarrow$ Brute force.

$$a \cdot b = \left(\sum_{i=0}^{\ell-1} a_i \cdot 2^{i-1}\right)\left(\sum_{j=0}^{\ell-1} b_j \cdot 2^{j-1}\right).$$

$(a_0, \dots, a_\ell)$

LSB MSB

[Strassen-Schonhäge]

$O(\ell \log\ell \log\log\ell)$

$O(\ell^{\log_2 3}) \leftarrow$

Karatsuba's integer mult.

$c \cdot n^3 \cdot \ell^{\log_2 3}$

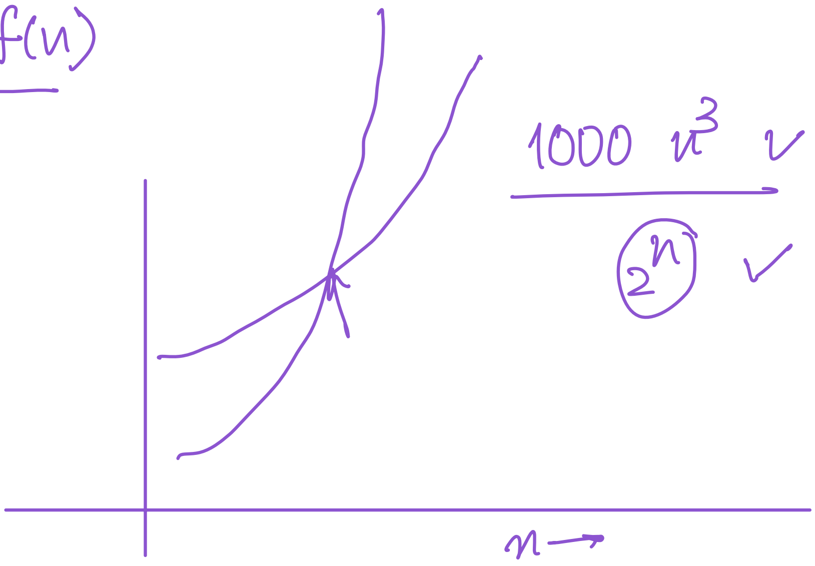
Qu: What does O( ) mean?

"Measure" of how the runtime/complexity (time) grows as a function of input size. $\underline{T(n) = O(f(n))}$

↳ $\exists\, n_0 \in \mathbb{N}$, and a constant $c \in \mathbb{R}$ s.t $\forall\, \underline{n \geq n_0}$,

$\underline{T(n)} \leq \underline{c \cdot f(n)}$

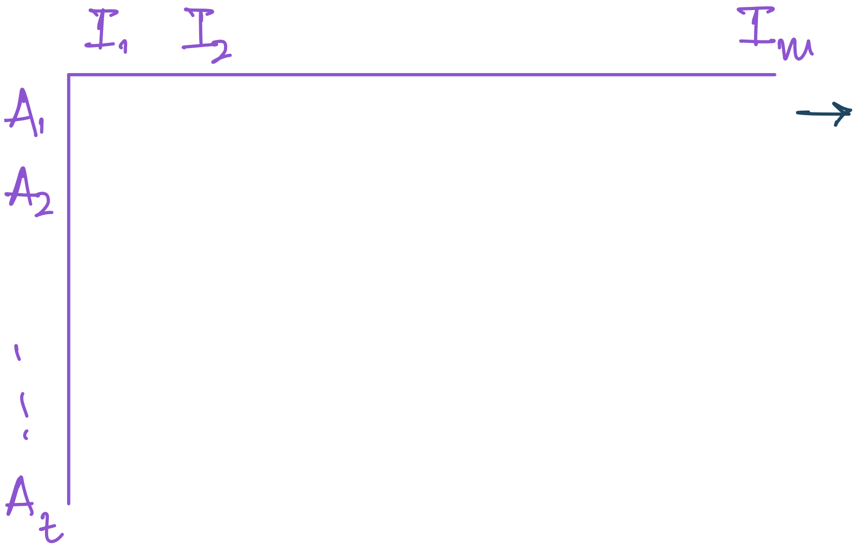
$\underline{O(n^3)}$



$$\frac{1000\, n^3 \ \checkmark}{\boxed{2^n} \ \checkmark}$$

Worst-case analysis:

$$\sum_{i=1}^{m} P_i \cdot T_A(I_i)$$

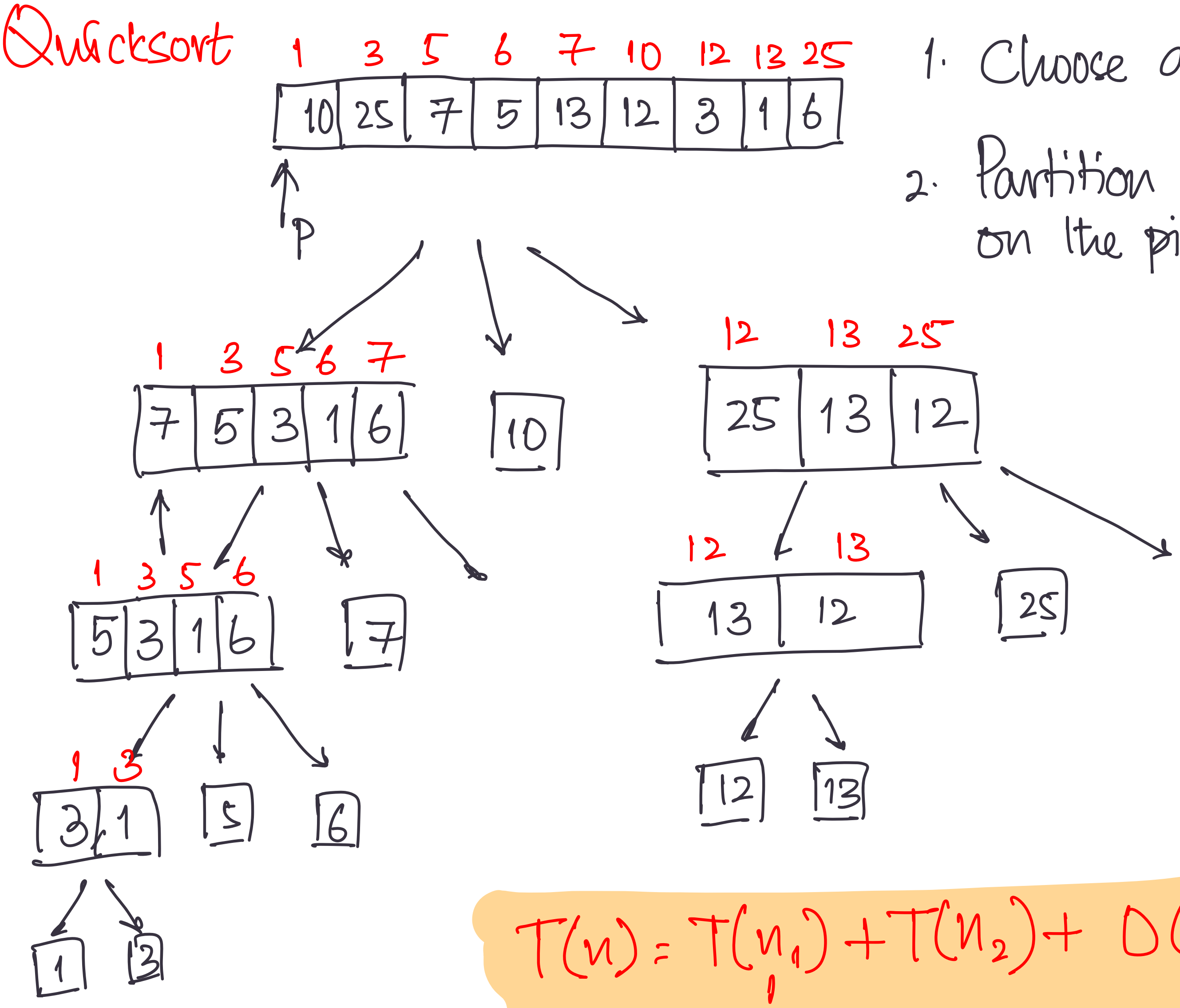


$$\rightarrow \max_{I} \{\overset{\text{time}}{A(I)}\}$$

Quicksort

1. Choose a pivot

2. Partition based on the pivot.



$$T(n) = T(n_1) + T(n_2) + O(n)$$

n_1 = # of elems < P

n_2 = # of elems > P

$T(1) = 1$

"Efficient algorithms"

↳ Anything better than brute-force.

<u>Qu:</u> Count the no. of triangles in a given graph $G=(V,E)$

↙

Pick triples (a,b,c) from the vertex set. } .
Check if they form a triangle.

$\binom{n}{3}\cdot$ (time to check if they form a triangle).

$A, \underline{\underline{A^2}}$

A

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$a_{ik} \leftarrow 0/1$ indicating presence of edge (i,k)

$$\underline{\underline{A^2_{ij}}} = \sum_{k=1}^{n} \underbrace{a_{ik}\cdot a_{kj}}_{} \;\}$$

↳ 1 if and only if edges a_{ik} and a_{kj} exist.

↑

entry gives us # of $i \to j$ <u>paths</u> of length 2.

```
Count = 0
For each i and j with i<j:
        Count += $A^2_{i,j}$   if and if $A_{ij} = 1$.        } $O(n^\omega)$

Return  count/3
```