Ghid recapitulare – Sortari & Order Statistics

Sortari prin comparatii

- **Insertion Sort**
- Bun pt. vectori mici / aproape sortati.
- Worst-case: O(n^2).

- **Merge Sort**
- Divide et impera: divide → conquer → merge.
- Complexitate: O(n log n) mereu.
- Nu e in-place (foloseste memorie suplimentara).

- **Heap Sort**
- Foloseste max-heap.
- BuildHeap: O(n).
- Sortare: n apeluri Heapify → O(n log n).
- In-place, worst-case tot O(n log n).
- Practic: mai lent decat Quicksort.

- **Quick Sort**
- Divide et impera cu pivot.
- Average: O(n log n).
- Worst-case: O(n^2) (ex. sir sortat daca pivot prost).
- Randomizare pivot → elimina caz prost.
- Cel mai rapid in practica.

Limite teoretice

- Toate sortarile prin comparatii au limita Ω(n log n).
- Dovada: arborele de decizie (n! permutari → inaltime minima log(n!) ≈ n log n).

Sortari liniare (fara comparatii)

- **Counting Sort**: O(n + k). Necesita interval mic de valori (1..k). Nu e in-place.
- **Radix Sort**: O(d·(n+k)), cu Counting Sort pe cifre. Devine O(n) daca d constant si k=O(n). Stabil.

- **Bucket Sort**: pentru valori uniforme in [0,1). Expected O(n).

Order Statistics

- **Minim / Maxim**: O(n).
- **Min + Max impreuna**: 3 comparatii pentru 2 elemente → ~1.5n comparatii.
- **Selectia (al i-lea cel mai mic)**:
- Naiv: sortezi → O(n log n).
- Randomized Select → O(n) in medie.
- Median of Medians → O(n) worst-case (teoretic, nu prea folosit practic).

De stiut la examen:

- Diferente cheie intre sortari (merge vs quick, quick vs heap).
- Cand putem face mai bine de O(n log n) (Counting / Radix / Bucket).
- Cum gasim rapid min, max, mediana.
- De ce limitele Ω(n log n) NU se aplica la Counting/ Radix (nu folosesc comparatii).