UNIT II · LESSON 07

Algorithm Analysis & Design

asymptotic notation · recurrences · master theorem · divide & conquer · greedy · dynamic programming · backtracking · branch & bound · P / NP

20
questions in Unit II
Lists · Trees · Sorting · Hashing · Graphs

DEFINITION ALGORITHM PROPERTIES

- Finite sequence of unambiguous steps: **input, output, definiteness, finiteness, effectiveness**
- Analysis: **time** & **space** complexity; best, average, worst case
- **Apriori** (theoretical) vs **aposteriori** (measured) analysis
- Space = fixed part (code, constants) + variable part (stack, dynamic data)

COUNT LOOP COMPLEXITY

Code pattern	Time
`for i=1..n`	O(n)
nested i=1..n, j=1..n	$O(n^2)$
`for (j=1; j<=i; j++)` **inside i loop**	$n(n+1)/2 \rightarrow O(n^2)$
`for (i=1; i<n; i*=2)`	O(log n)
`for (i=n; i>0; i/=2)` **with inner j=1..n**	O(n log n)
`for (i=1; i*i<=n; i++)`	O(√n)
inner j=1..i, i doubling (1,2,4...n)	O(n)

GROWTH ASYMPTOTIC NOTATION

Notation	Meaning	Bound
O(g)	$f \le c \cdot g$ for $n \ge n_0$	upper (worst-case style)
Ω(g)	$f \ge c \cdot g$ for $n \ge n_0$	lower
Θ(g)	$c_1 g \le f \le c_2 g$	tight
o(g)	$f/g \rightarrow 0$	strictly smaller
ω(g)	$f/g \rightarrow \infty$	strictly larger

- Order: **$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < 2^n < n! < n^n$**
- $3n^2 + 5n + 7 = \Theta(n^2)$; $\log_a n = \Theta(\log_b n)$ (any bases)
- $\Sigma\, i = n(n+1)/2 = \Theta(n^2)$; $\Sigma\, 1/i = \Theta(\log n)$; $\Sigma\, 2^i$ $(i \le n) = 2^{n+1} - 1$
- $\log n! = \Theta(n \log n)$

SOLVE RECURRENCES

Recurrence	Solution	Example
T(n) = T(n−1) + 1	O(n)	factorial, linear search (rec.)
T(n) = T(n−1) + n	$O(n^2)$	quick sort worst, selection
T(n) = T(n/2) + 1	O(log n)	binary search

SOLVE RECURRENCES (CONTD.)

Recurrence	Solution	Example
T(n) = 2T(n/2) + 1	O(n)	tree traversal, max of array
T(n) = 2T(n/2) + n	O(n log n)	merge sort
T(n) = 2T(n−1) + 1	$O(2^n)$	Tower of Hanoi
T(n) = T(n−1) + T(n−2)	$O(\varphi^n) \approx 1.618^n$	naive Fibonacci
T(n) = 7T(n/2) + n^2	$O(n^{2.81})$	Strassen

- Methods: **substitution**, **recursion tree**, **master theorem**

LOCAL BEST GREEDY METHOD

- Choose the **locally optimal** choice, never reconsider; needs **greedy-choice property + optimal substructure**

Problem	Greedy rule	Time
Fractional knapsack	highest value/weight first	O(n log n)
Activity selection	earliest **finish** time	O(n log n)
Job sequencing with deadlines	highest profit, latest free slot	$O(n^2)$
Huffman coding	merge two least frequencies	O(n log n)
Optimal merge pattern	merge two smallest files	O(n log n)
MST (Prim, Kruskal)	lightest safe edge	O(E log V)
Dijkstra	nearest unvisited vertex	O(E log V)

T(N) = AT(N/B) + F(N) MASTER THEOREM

- Compare f(n) with $\mathbf{n^{\log_b a}}$

Case	Condition	T(n)
1	$f(n) = O(n^{\log_b a - \varepsilon})$	$\Theta(n^{\log_b a})$
2	$f(n) = \Theta(n^{\log_b a} \log^k n)$	$\Theta(n^{\log_b a} \log^{k+1} n)$
3	$f(n) = \Omega(n^{\log_b a + \varepsilon})$, regularity	Θ(f(n))

- $4T(n/2) + n \to n^2$; $4T(n/2) + n^2 \to n^2 \log n$; $4T(n/2) + n^3 \to n^3$
- $3T(n/4) + n \log n \to n \log n$ (case 3)

SPLIT–SOLVE–COMBINE DIVIDE & CONQUER

Problem	Recurrence	Time
Binary search	T(n/2) + 1	O(log n)
Merge sort	2T(n/2) + n	O(n log n)
Quick sort (avg)	2T(n/2) + n	O(n log n)
Max & min (pairs)	2T(n/2) + 2	**3n/2 − 2** comparisons
Strassen matrix multiply	$7T(n/2) + n^2$	$O(n^{2.81})$
Karatsuba multiply	3T(n/2) + n	$O(n^{1.585})$
Closest pair of points	2T(n/2) + n	O(n log n)

LOCAL BEST GREEDY METHOD (CONTD.)

- **0/1 knapsack cannot be solved greedily** → use DP

FRACTIONAL KNAPSACK EXAMPLE

- W = 50; items (v, w): (60, 10), (100, 20), (120, 30) → ratios 6, 5, 4
- Take items 1, 2 fully (30 kg, 160) + 20/30 of item 3 (80) → **240**
- 0/1 version: best = items 2 + 3 = **220**

SEARCH TREES BACKTRACKING & B&B

- **Backtracking**: DFS of state-space tree, prune when a **constraint** fails
- Examples: **N-queens** (8-queens: 92 solutions), sum of subsets, graph colouring, Hamiltonian cycle, sudoku
- **Branch & bound**: BFS / least-cost search with **bounds** to prune; for optimisation
- Examples: 0/1 knapsack, **TSP**, job assignment

Backtracking	Branch & bound
DFS	BFS / best-first (LC)
feasibility (constraints)	optimisation (bounds)
finds all/any solutions	finds the optimal solution

COMPLEXITY CLASSES P, NP, NP-COMPLETE

- **P**: solvable in polynomial time (sorting, MST, shortest path)
- **NP**: solution **verifiable** in polynomial time; P ⊆ NP
- **NP-hard**: every NP problem reduces to it

OVERLAPPING SUBPROBLEMS DYNAMIC PROGRAMM

- Needs **optimal substructure** + **overlapping subproblems**; store results (**memoisation** top-down or **tabulation** bottom-up)

Problem	Recurrence idea	Time
Fibonacci	$F(n) = F(n-1) + F(n-2)$	$O(n)$
0/1 knapsack	$K(i,w) = \max(K(i-1,w), v_i + K(i-1, w - w_i))$	$O(nW)$ pseudo-poly
LCS	c[i][j] = c[i−1][j−1] + 1 if match, else max	$O(mn)$
Matrix-chain multiplication	m[i][j] = min over k (m[i][k] + m[k+1][j] + $p_{i-1}p_kp_j$)	$O(n^3)$
Floyd–Warshall	$d_k[i][j]$	$O(n^3)$
Bellman–Ford	relax V − 1 times	$O(VE)$
Optimal BST	min over roots	$O(n^3)$
TSP (Held–Karp)	subsets	$O(n^2 2^n)$
Edit distance	insert/delete/replace	$O(mn)$

- Matrix chain: A(10×30) B(30×5) C(5×60): (AB)C = 1500 + 3000 = **4500**, A(BC) = 9000 + 18000 = 27000
- LCS("ABCBDAB", "BDCABA") length = **4** (e.g. BCBA)

COMPLEXITY CLASSES P, NP, NP-COMPLETE (CONTD.)

- **NP-complete** = NP ∩ NP-hard: SAT (Cook, first), 3-SAT, clique, vertex cover, Hamiltonian cycle, TSP (decision), subset sum, 0/1 knapsack (decision), graph colouring
- If any NP-complete problem is in P → **P = NP** (open problem)
- Halting problem: **undecidable**, NP-hard but not in NP

TRAP EXAM TRAPS

- O is an upper bound, not necessarily tight: $n = O(n^2)$ is true
- Greedy fails for 0/1 knapsack and for coin systems like {1, 3, 4} (6 = 3 + 3)
- Master theorem does not apply to $T(n) = T(n-1) + n$ or $2T(n/2) + n/\log n$
- Knapsack DP $O(nW)$ is **pseudo-polynomial**, not polynomial
- Dynamic programming ≠ divide & conquer: DP subproblems **overlap**

Q → A RAPID RECALL

$T(n) = 2T(n/2) + n$	$\Theta(n \log n)$
$T(n) = 8T(n/2) + n^2$	$\Theta(n^3)$
$T(n) = T(n/2) + n$	$\Theta(n)$
Strategy of merge sort	divide & conquer
Strategy of Floyd–Warshall	dynamic programming
Huffman coding strategy	greedy
Fractional knapsack optimal by	greedy (ratio)
Matrix chain for 4 matrices: parenthesisations	5 (Catalan C_3)
First problem proved NP-complete	SAT (Cook, 1971)
8-queens number of solutions	92