Attack order + exam rules

Order: spec → type → ES → greedy/D&C/DP → BT/B&B → LB → NP-c → compromise (approx/rand/**FPT**)

Unqualified "runtime" = **WORST** case; give Θ if provable, else smallest O

- **Any of** $O/\Omega/\Theta$ can describe any of worst/best/avg case. Never pair O↔w.c. or Ω↔best

Use the paradigm **ASKED** even if clumsy; off-paradigm cleverness penalized

Types: optimization (constructive / evaluation), decision, search, counting, enumeration

opt→decision: add bound B; min asks "$\le B$", max asks "$\ge B$"

Proof shapes

Not correct(4): instance x | x valid | alg gives y (show each choice) | correct z better

Optimal substructure(3): build smaller I' from O | split O into pieces | a bad piece could be swapped, beating O — contradiction

Ties: unspecified ⇒ arbitrary; alg correct only if correct for **ALL** tie-breakings; arbitrary ≠ random

Paradigm checklists

ES(5): possibilities | generate all/next | info | extract info | form solution

Greedy(5): preprocess | steps | info for decision | criteria | update info

D&C(4): smaller instances | base cases | dividing | combining

DP(6): recurrence | entry info | bases | table shape | eval order | extract

BT(10): partial soln (PS) | PS@root | extra info | root | non-root | candidate? | can't extend?
| extend | update info | output. B&B(11) = BT + bounding fn as item 4

Bounding fn: max ⇒ **UPPER** bound, min ⇒ **LOWER** bound, on any candidate below PS

Prune when bound worse than best-so-far (max: smaller; min: larger)

DP table craft

Cost = create table + #entries × cost/entry + extract; filling dominates

Entry: decision→T/F; eval→value; search/constr→which smaller instance won (to trace)

MatrixChain $m[i,j] = \min_{i\le k<j}\{m[i,k] + m[k+1,j] + d_i d_{k+1} d_{j+1}\}$, $m[i,i]=0$

LCS $L[i,j] = L[i-1,j-1] + 1$ if $x_i{=}y_j$ else $\max\{L[i-1,j], L[i,j-1]\}$; row0=col0=0

Knapsack $C[i,j] = \max\{C[i-1,j],\ C[i-1,j-w_i] + v_i\}$; $n \times (W{+}1)$, rows increasing

FW $c(i,j,k)$ = cheapest $i \to j$, interiors of index $< k$: $\min\{c(i,j,k-1),\ c(i,k-1,k-1) + c(k-1,j,k-1)\}$; k=0: edge wt / ∞

Order notation

O: $\exists$ real $c>0$, int $n_0 \ge 1$: $f \le cg\ \forall n \ge n_0$. Ω: $f \ge cg$. Θ: $c_1 g \le f \le c_2 g$

"Formal definitions" ⇒ explicit c, n_0; c may be < 1; keep the **LARGER** n_0

Proof craft: UB drop negative terms, $g \ge 1$ absorbs consts; LB split off cg, leftover ≥ 0

Sum: keep dominant term(s); Θ iff every **KEPT** term is Θ; one kept term only O ⇒ whole sum O

Never substitute a value for n or m; once specified, answer as a fn of it

$1 \prec \log\log n \prec \log n \prec \log^2 n \prec n \prec n\log n \prec n^2 \prec n^3 \prec 2^n \prec n! \prec n^n$

Cost model (k = length)

$\Theta(1)$: new empty list, len, $a[i]$ get/set, append, pop() last, 2-input arith/cmp, attribute access

- $\Theta(k)$: [k items], 'in', insert, pop(p), remove, **COPY** (even with '='), max/min of k

$a{+}b$: $\Theta(k{+}\ell)$; $a[i{:}j]$: $\Theta(j{-}i)$; sorted/.sort: $\Theta(k \log k)$; map/filter: $\Theta(k\,f(k))$

Graph/tree: vertices $\Theta(n)$, edges $\Theta(m)$, neighbours/adjacent/edge wt $\Theta(d)$; children $\Theta(c)$

k constant ⇒ that $\Theta(k)$ op is constant time

TARGET NUMBER binary search $\Theta(\log n)$ in place; slicing degrades to $\Theta(n)$

MAX SUBTOTAL D&C $\Theta(n \log n)$: combine = max{L soln, R soln, maxsuf(L)+maxpre(R)}

Recurrences

Master $T = aT(n/b) + f$: $a \ge 1, b > 1$ **CONSTANT**, ignore floors; $x = n^{\log_b a}$
f smaller than x ⇒ $\Theta(x)$; f bigger ⇒ $\Theta(f)$; same size ⇒ $\Theta(x \log n)$
N/A for $T(n{-}1)$, unequal splits ($n/5$ & $7n/10$), non-constant a or b

x: $a{=}1 \Rightarrow 1$; $a{=}b \Rightarrow n$; $4T(n/2) \Rightarrow n^2$; $8T(n/2) \Rightarrow n^3$; $2T(n/4) \Rightarrow \sqrt{n}$; $3T(n/2) \Rightarrow n^{\log 3}$

Iteration(3): unwind to general k | pick k reaching base | closed form

Substitution(4): guess (no order notation) | substitute | simplify | check bases
if cg fails try $cg{-}h$, $h \in O(g)$ ($cn{-}b$, $cn^2{-}dn$); fix constants **LAST**

$T(n{-}1) + \Theta(1) \Rightarrow \Theta(n)$; $T(n/2) + \Theta(1) \Rightarrow \Theta(\log n)$; $T(n/2) + \Theta(n) \Rightarrow \Theta(n)$; $2T(n/2) + \Theta(n) \Rightarrow \Theta(n \log n)$

Lower bounds

Info theory: ℓ possible outputs ⇒ any **COMPARISON-BASED** alg needs $\Omega(\log \ell)$ w.c.
SORTING $\ell{=}n!$ ⇒ $\Omega(n \log n)$; search among n ⇒ $\Omega(\log n)$; yes/no ⇒ $\Omega(1)$

Adversary(3): state strategy | claim bound T = #key ops any correct alg needs | after $T{-}1$ steps ≥ 2 consistent instances with **DIFFERENT** outputs

Hard problems

NP-complete here: TSP DECISION, VERTEX COVER, INDEPENDENT SET, HAM CYCLE,
k-**COLOURING** ($k \ge 3$), KNAPSACK, LONG PATH, JOB SCHED decision (even $m{=}2$)

Reductions

$A \le_p B$ = solve A with poly-many calls to B + poly extra work ⇒ A no harder

Easy B ⇒ Easy A; Hard A ⇒ Hard B. NO converses. Equivalent = both directions

- **To show** Z hard, reduce **KNOWN**-hard $Y \to Z$. Reversed proves **NOTHING**

IS $\le_p$ VC: $f(G,k) = (G, n{-}k)$; IS of size $\ge k$ iff VC of size $\le n{-}k$

NP proof recipes

NP-hard: every $X \in$ NP reduces to Y. NP-complete = in NP **AND NP**-hard

- **ADJECTIVES not classes:** write "Y is in NP", **NEVER** "Y is in NP-complete"

$P \subseteq NP$; one **NPC** problem in P ⇒ P=NP; one **NPC** problem not in P ⇒ P≠NP

In NP(4): certificate + poly size | verifier + poly w.c. time | yes-instance gets "Yes" | not fooled by false certificates

NPC(6): $Z \in$NP | pick **NPC** Y | alg for f: Y-inst→ Z-inst | x yes ⇒ $f(x)$ yes | $f(x)$ yes ⇒ x yes (or contrapositive) | f runs in poly time

- f must map **ALL** instances of Y; need not cover all of Z; step 5 ≠ repeat of step 4

Classes: $P \subseteq ZPP \subseteq RP \subseteq NP$; $ZPP \subseteq RP \subseteq BPP$; $NC \subseteq L \subseteq NL \subseteq P$; $FPTAS \subseteq PTAS \subseteq APX$

- **BPP vs NP UNKNOWN**; RP/BPP = MC 1-/2-sided; ZPP = LV exp poly

Approximation

- **RATIO BOUND** $R(n)$ (not "approx ratio"): $\max\{A/O,\ O/A\} \le R(n)$ for **EVERY** size-n instance

min ⇒ use A/O (upper-bd A, lower-bd O); max ⇒ use O/A (reverse)

Approx alg = feasible soln + provable bound; poly time **NOT** in the definition

Analyze(4): pick measure M | bound A by M | bound O by M | combine, M cancels

Disprove(3): pick instance x of size n | show x valid | $\max\{A/O, O/A\} > R(n)$

Disprove $\Theta(f)$: need a **FAMILY** of arbitrarily large instances, ratio $> g$, $g \ge f$, $g \notin \Theta(f)$

APX = poly-time approx alg with a **CONSTANT** ratio bound

VC 2-approx (taught as GREEDY): repeatedly take **BOTH** ends of an uncovered edge

METRIC/EUCLIDEAN TSP 2-approx: **MST** → full walk → shortcuts; $A \le 2M$, $O \ge M$

Randomized

Las Vegas: always **CORRECT**, **EXPECTED** poly time (no w.c. guarantee)

Monte Carlo: always **POLY** time, correct with high probability

- LV→MC always possible; MC→LV no known general method

false positive = yes on a no-inst; false negative = no on a yes-inst; both ⇒ two-sided

1-sided ⇒ the OTHER answer is CERTAIN: no FN ⇒ trust "No"; no FP ⇒ trust "Yes"

Heuristics / FPT / online

Hill climb: any feasible start | repeat a small improving change | stop = local opt (may be suboptimal)

FPT: $O(g(p) \cdot n^{O(1)})$, g a fn of p **ONLY**; n^k is **NOT FPT**; if p depends on n, not **FPT**

VC-by-k bounded search tree: height $\le k$, 2 children ⇒ $O(2^k n^{O(1)})$; uses BT

Kernel(2): kernel is a yes-instance **IFF** original is; size bounded by fn of p; both steps **FPT**

Online competitive ratio c: cost $\le c\times$ **BEST OFFLINE** cost, for **ANY** input

Math toolkit

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$; $\sum_{i=1}^{n} c^{i-1} = \frac{1-c^n}{1-c}$; $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$

$\log = \log_2$; $\log_a x = \log_b x / \log_b a$; $\log xy = \log x + \log y$; $\log x^k = k \log x$; $n^a n^b = n^{a+b}$; $(n^a)^b = n^{ab}$

perms $n!$; ordered k-seqs $\frac{n!}{(n-k)!}$; $C(n,k) = \frac{n!}{k!(n-k)!}$; 2^n subsets; k^n categorizations

$\log n! \in \Omega(n \log n)$; $C(n,k) \in \Theta(n^k)$