

# Algorithms Cheat Sheet — Lectures 1–8 (Open Notes)

Dense formula/template reference · Copy these templates on the exam · Lectures 1–8

## 1. Foundations (L1–2)

### Core definitions

**Problem:** a computational task specified by input/output relation (e.g., sorting). **Instance:** one concrete input (e.g., [3,1,4]). **Algorithm:** a well-defined computational procedure that takes an instance and produces a correct output for every instance. Correctness = produces right answer; Efficiency = resource use (time/space) as function of input size.

### Insertion Sort — idea

Grow a sorted prefix  $A[1..j-1]$ . At step  $j$ , take  $\text{key} = A[j]$  and insert it into the correct position in the prefix by shifting larger elements right. Like sorting cards in hand. Best  $\Theta(n)$  (already sorted); worst/avg  $\Theta(n^2)$  (reverse sorted / random).

INSERTION-SORT( $A$ ):

```
for j = 2 to n:
    key = A[j]; i = j - 1
    while i > 0 and A[i] > key:
        A[i+1] = A[i]; i = i - 1
    A[i+1] = key
```

### Loop Invariant — 3-part template (COPY THIS)

**Initialization:** It is true before the first iteration.

**Maintenance:** If true before iteration  $i$ , it remains true before iteration  $i+1$ .

**Termination:** When the loop ends, the invariant + exit condition  $\Rightarrow$  desired property (correctness).

Ex (Insertion Sort): At start of iteration  $j$ ,  $A[1..j-1]$  contains the same elements as originally, in sorted order.

### RAM model bullets

- Word size  $\Theta(\lg n)$  bits · constant-time arithmetic/comparison on words · each simple instruction (arith, data move, control) costs  $O(1)$  · memory is a large array of words · ignore caches/pipeline · input size = number of words / elements (not bits, unless bit-complexity matters).

## 2. Proof Toolkit (L3–4)

### Weak Induction — prove $P(n)$ for all $n \geq n_0$

1. **Base:** Prove  $P(n_0)$  directly.

2. **IH:** Assume  $P(k)$  for some  $k \geq n_0$ .

3. **Inductive step:** Prove  $P(k+1)$  using IH.

Ex:  $\sum_{i=1..n} i = n(n+1)/2$ . Base  $n=1$ :  $1=1$ . IH:  $\text{sum to } k$ . Step:  $\text{sum to } k+1 = k(k+1)/2 + (k+1) = (k+1)(k+2)/2$ .

### Strong Induction — IH assumes $P(n_0) \dots P(k)$ all true

1. **Base:** Prove  $P(n_0)$  [sometimes several base cases].

2. **IH:** Assume  $P(j)$  for all  $n_0 \leq j \leq k$ .

3. **Step:** Prove  $P(k+1)$  using any/all of those.

Ex: Every  $n \geq 2$  has a prime factor. Base  $n=2$ . For  $k+1$ : if prime done; else factor  $a \cdot b$  with  $2 \leq a \leq k$ , apply IH to  $a$ .

### Contradiction

1. Assume the negation of the claim. 2. Derive a logical contradiction ( $A \wedge \neg A$ , or violate known fact). 3. Conclude claim is true.

Ex:  $\sqrt{2}$  irrational. Assume  $p/q$  in lowest terms. Then  $2q^2 = p^2 \Rightarrow p$  even  $\Rightarrow p=2r \Rightarrow q$  even  $\Rightarrow$  contradicts lowest terms.

**Counterexample** — to disprove  $\forall x P(x)$ , exhibit one  $x$  where  $\neg P(x)$ .

Ex: "All primes are odd" — counterexample: 2.

**Direct proof** — assume hypothesis  $H$ , chain implications to conclusion  $C$ . No negation.

Ex: If  $n$  even then  $n^2$  even:  $n=2k \Rightarrow n^2=4k^2=2(2k^2)$ .

## 3. Analysis (L4)

**Input size caveats:** Sorting  $\rightarrow n = \# \text{elements}$ . Graphs  $\rightarrow |V|+|E|$ . Integers  $\rightarrow \# \text{bits}$  (bit complexity) if ops not  $O(1)$ . Matrices  $\rightarrow$  dimension or  $\# \text{entries}$ . Always state what  $n$  measures.

**Exact vs asymptotic  $T(n)$ :** Exact counts every primitive op (messy constants). Asymptotic abstracts to growth rate as  $n \rightarrow \infty$ : drop lower-order terms and constant factors  $\rightarrow \text{Big-}O / \Omega / \Theta$ . Worst-case  $T(n) = \max$  over inputs of size  $n$ ; best-case = min; average needs distribution.

## 4. Asymptotics (L4–5)

### Formal definitions

**$O$ :**  $f(n)=O(g(n))$  iff  $\exists c>0, n_0>0$  s.t.  $0 \leq f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$ . (upper bound)

**$\Omega$ :**  $f(n)=\Omega(g(n))$  iff  $\exists c>0, n_0>0$  s.t.  $0 \leq c \cdot g(n) \leq f(n)$  for all  $n \geq n_0$ . (lower bound)

**$\Theta$ :**  $f(n)=\Theta(g(n))$  iff  $\exists c_1, c_2>0, n_0>0$  s.t.  $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \forall n \geq n_0$ . ( $\equiv O \cap \Omega$ )

**little- $o$ :**  $f(n)=o(g(n))$  iff  $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$ . (strictly slower; stronger than  $O$ )

**little- $\omega$ :**  $f(n)=\omega(g(n))$  iff  $\lim_{n \rightarrow \infty} f(n)/g(n) = \infty$ . (strictly faster; stronger than  $\Omega$ )

**Polynomial:** any degree- $d$  poly  $a_d n^d + \dots + a_0$  with  $a_d \neq 0$  is  $\Theta(n^d)$ .

### Growth ladder (slow $\rightarrow$ fast)

$1 \ll \lg \lg n \ll \lg n \ll (\lg n)^k \ll n^\epsilon \ll n \ll n \lg n \ll n^2 \ll n^3 \ll n^k \ll 2^n \ll n^n \ll n!$  ( $\epsilon > 0$ ,  $k$  fixed)

### Property table ( $\checkmark$ = yes, $\times$ = no)

| Prop                                          | $O$          | $\Omega$     | $\Theta$         | $o$          | $\omega$     |
|-----------------------------------------------|--------------|--------------|------------------|--------------|--------------|
| Reflexive ( $f \sim f$ )                      | $\checkmark$ | $\checkmark$ | $\checkmark$     | $\times$     | $\times$     |
| Symmetric ( $f \sim g \Rightarrow g \sim f$ ) | $\times$     | $\times$     | $\checkmark$     | $\times$     | $\times$     |
| Transitive                                    | $\checkmark$ | $\checkmark$ | $\checkmark$     | $\checkmark$ | $\checkmark$ |
| $f=O(g) \wedge g=O(f) \Rightarrow \Theta$     | —            | —            | $\checkmark$ def | —            | —            |

## Proof templates with $c, n_0$

**Prove  $f=O(g)$ :** Find  $c>0, n_0>0$ . Show  $f(n) \leq c \cdot g(n) \forall n \geq n_0$ . Often pick  $c \geq \max$  of leading-coeff ratios / bound lower terms.

**Prove  $f=\Omega(g)$ :** Find  $c>0, n_0$ . Show  $f(n) \geq c \cdot g(n) \forall n \geq n_0$ .

**Prove  $f=\Theta(g)$ :** Prove both  $O$  and  $\Omega$  (two pairs  $(c_1, n_0), (c_2, n_0)$  or one  $n_0$  with both).

**Prove  $f=o(g)$ :** Show  $\lim f/g = 0$  (L'Hôpital / algebra). **Disprove  $O$ :** show  $\forall c, n_0 \exists n \geq n_0$  with  $f(n) > c \cdot g(n)$ .

**Decide prove vs disprove:** If claim is true  $\rightarrow$  construct  $c, n_0$ . If false  $\rightarrow$  counterexample sequence or limit argument.

## Worked examples (COPY THESE)

•  $n^3 = \Omega(n)$ : Need  $c \cdot n \leq n^3 \forall n \geq n_0$ . Take  $c=1, n_0=1$ :  $n \leq n^3$  for  $n \geq 1$ . ✓

•  $n = o(n^2)$ :  $\lim n/n^2 = \lim 1/n = 0$ . ✓ (also  $n=O(n^2)$  but not  $\Theta$ )

•  $5n^3+2n = \Theta(n^3)$ : Lower:  $5n^3+2n \geq 5n^3 \geq 1 \cdot n^3$  for  $n \geq 1 \Rightarrow c_1=1$ . Upper:  $5n^3+2n \leq 5n^3+2n^3 = 7n^3$  for  $n \geq 1 \Rightarrow c_2=7, n_0=1$ . ✓

## 5. Divide & Conquer (L6)

**Pattern: Divide** — split into subproblems · **Conquer** — solve recursively (base: brute force) · **Combine** — merge sub-solutions into answer. Always return values (don't rely on globals).

### Merge Sort

Divide: mid. Conquer: sort left & right. Combine: merge two sorted lists in  $O(n)$ . Recurrence:  $T(n) = 2T(n/2) + O(n)$ ,  $T(1)=\Theta(1) \rightarrow \Theta(n \lg n)$  (all cases).

### Min-Max simultaneous

Find both min and max with fewer than  $2(n-1)$  comparisons. Divide into two halves; recurse; combine with 2 comparisons (min of mins, max of maxes).

```
MIN-MAX(A[lo..hi]):
  if hi - lo <= 1: return (min(A[lo], A[hi]), max(A[lo], A[hi])) # 1 cmp
  mid = (lo+hi)/2
  (mn1, mx1) = MIN-MAX(A[lo..mid]); (mn2, mx2) = MIN-MAX(A[mid+1..hi])
  return (min(mn1, mn2), max(mx1, mx2)) # +2 cmp
# T(n)=2T(n/2)+O(1) → Θ(n) (≈ 3n/2 comparisons)
```

### Inversion count (pairs $i < j$ with $A[i] > A[j]$ ) via modified merge

**Merge rule:** When merging  $L$  and  $R$ , if you take from  $R$  ( $R[j] < L[i]$ ), add  $(|L| - i) = \#$  remaining elements in  $L$  — each forms an inversion with  $R[j]$ . Taking from  $L$  adds 0. Recurrence:  $T(n)=2T(n/2)+O(n) \rightarrow \Theta(n \lg n)$ .

### Strassen matrix multiply

Classic MM:  $T(n)=8T(n/2)+O(n^2) \rightarrow \Theta(n^3)$ . Strassen: 7 recursive multiplies +  $O(n^2)$  adds  $\rightarrow T(n)=7T(n/2)+O(n^2) \rightarrow$  by Master Thm case 1:  $O(n^{\lg 7}) \approx O(n^{2.807})$ .

## 6. Solving Recurrences (L7–8)

### Substitution method steps

1. **Guess** the form (often from tree intuition or Master Thm).
2. **Prove by induction** (strong). For  $O$ : show  $T(n) \leq c \cdot f(n)$ . For  $\Omega$ : show  $T(n) \geq c \cdot f(n)$ .
3. Verify **base cases** (choose  $n_0$  so bases hold; adjust  $c$ ).
4. **Subtract lower-order trick:** If induction fails on lower-order terms, strengthen guess:  $T(n) \leq c \cdot f(n) - d \cdot g(n)$  (pick  $d$  to absorb the leftover).

### Worked: $T(n)=8T(n/2)+n = \Omega(n^3)$

Guess  $T(n) \geq c \cdot n^3$ . IH:  $T(n/2) \geq c \cdot (n/2)^3 = c \cdot n^3 / 8$ .  
 $T(n) \geq 8 \cdot (c \cdot n^3 / 8) + n = c \cdot n^3 + n \geq c \cdot n^3$ . (true for any  $c > 0$ )  
Base: choose  $c$  small enough /  $n_0$  large enough so  $T(n_0) \geq c \cdot n_0^3$ .  $\Rightarrow T = \Omega(n^3)$ .  
(Actually  $\Theta(n^3)$  by Master case 1 / similar 0 proof.)

### Worked: $T(n)=2T(n/2)+n^2 = \Omega(n^2)$ needing $c \leq 2$

Guess  $T(n) \geq c \cdot n^2$ . IH:  $T(n/2) \geq c \cdot (n/2)^2 = c \cdot n^2 / 4$ .  
 $T(n) \geq 2 \cdot (c \cdot n^2 / 4) + n^2 = (c/2) \cdot n^2 + n^2 = (c/2 + 1) \cdot n^2$ .  
Need  $(c/2 + 1) \cdot n^2 \geq c \cdot n^2 \Rightarrow c/2 + 1 \geq c \Rightarrow 1 \geq c/2 \Rightarrow c \leq 2$ .  
Pick  $c=2$  (or any  $0 < c \leq 2$ ); fix base.  $\Rightarrow T = \Omega(n^2)$ . (Also  $O(n^2) \Rightarrow \Theta$ .)

### Recursion-tree method outline

1. Draw tree: root cost  $f(n)$ , children  $f(n/b)$ , ... until leaves  $T(1)$ . 2. Cost per level; #levels =  $\log_b n$ ; #leaves =  $a^{\lg_b n} = n^{\lg_b a}$ . 3. Sum over levels (+ leaf costs)  $\rightarrow$  guess. 4. Prove rigorously by substitution.

**Ex:**  $T(n)=2T(n/2)+cn$ . Levels:  $cn + cn + \dots + cn$  ( $\lg n$  levels) +  $\Theta(n)$  leaves  $\rightarrow \Theta(n \lg n)$ .

### Master Theorem — $T(n)=a T(n/b)+f(n)$ , $a \geq 1, b > 1$

Compare  $f(n)$  to  $n^{\lg_b a}$ :

**Case 1:**  $f(n) = O(n^{\lg_b a - \epsilon})$  for some  $\epsilon > 0 \rightarrow T(n) = \Theta(n^{\lg_b a})$  (leaves dominate)

**Case 2:**  $f(n) = \Theta(n^{\lg_b a} \lg^k n)$ ,  $k \geq 0 \rightarrow T(n) = \Theta(n^{\lg_b a} \lg^{k+1} n)$  ( $k=0 \Rightarrow \Theta(n^{\lg_b a} \lg n)$ )

**Case 3:**  $f(n) = \Omega(n^{\lg_b a + \epsilon})$  for some  $\epsilon > 0$  **AND regularity:**  $a \cdot f(n/b) \leq c \cdot f(n)$  for  $c < 1$  and large  $n \rightarrow T(n) = \Theta(f(n))$  (root dominates)

**When MT fails:** polynomial gap missing (e.g.  $f = n^{\lg_b a} / \lg n$ ); regularity fails;  $a, b$  not constants; floors/ceilings usually OK but weird divides can break; non-polynomial differences.

Quick checks: MergeSort  $a=2, b=2, f=n \rightarrow n^{\lg 2}=n = \Theta(f)$  case2  $\rightarrow \Theta(n \lg n)$ . Strassen  $a=7, b=2, f=n^2 \rightarrow n^{\lg 7} \approx n^{2.81} > n^2$  case1  $\rightarrow \Theta(n^{\lg 7})$ .  $T=2T(n/2)+n^2 \rightarrow n$  vs  $n^2$  case3, regularity  $2(n/2)^2 = n^2/2 < c \cdot n^2 \checkmark \rightarrow \Theta(n^2)$ .

## 7. HW1 Flash Cards

**Second-largest + loop invariant:** Track largest and second-largest through the array. Invariant: after  $i$  elements,  $L$  = max of first  $i$ ,  $S$  = second-max of first  $i$  (or  $-\infty$ ). Init  $i=2$ ; maintain on each new  $A[i]$ ; terminate  $\Rightarrow S$  is second-largest.  $\approx n + \lg n - 2$  comparisons (tournament).

**Strong induction  $n! > 2^n$  (for  $n \geq 4$ ):** Base  $n=4$ :  $24 > 16$ . IH: assume for all  $4 \leq j < k$  that  $j! > 2^j$ . Step:  $(k+1)! = (k+1) \cdot k! > (k+1) \cdot 2^k \geq 2 \cdot 2^k = 2^{k+1}$  (since  $k+1 \geq 5 > 2$ ). Use strong or weak — both work; weak suffices here.

**Contradiction — composite has divisor  $\leq \sqrt{n}$ :** Let  $n$  composite  $\Rightarrow n = a \cdot b$  with  $1 < a \leq b < n$ . If both  $a, b > \sqrt{n}$  then  $a \cdot b > n$  — contradiction. Hence  $a \leq \sqrt{n}$ . (Useful for primality trial division up to  $\sqrt{n}$ .)

**PairwiseCheck Big-O:** Algorithm comparing all pairs  $\rightarrow \Theta(n^2)$  comparisons. If asked Big-O of a pairwise scan:  $O(n^2)$  is correct upper;  $\Theta(n^2)$  if exactly all pairs. Don't claim  $O(n)$  unless early abort is proven for all inputs.

## 8. Exam Traps — Don't Lose Points

- **$c > 0$**  always (positive constants). Never  $c=0$  or negative in  $O/\Omega/\Theta$  proofs.
- Must hold for **all**  $n \geq n_0$  — one counterexample  $n$  kills a claimed  $c, n_0$ . Pick  $n_0$  carefully.
- **Don't confuse  $o$  with  $O$** :  $o$  is strict (limit 0);  $O$  allows equality of growth ( $n=O(n)$  but  $n \neq o(n)$ ).  $\Theta \neq o$ .
- **Don't solve a recurrence unless asked** — if Q only wants the recurrence, write  $T(n)=\dots$  and stop.
- **D&C writeups: return values, not globals**. Pseudocode should return (min,max) / count / array — side effects hide bugs and lose points.
- Base cases matter in induction/substitution — skipping them loses credit even if inductive step is perfect.
- Master Thm Case 3 needs **regularity** — forgetting it is a common deduction.
- $\lg = \log_2$  in CLRS-style courses unless otherwise stated;  $\ln = \ln$ ;  $\log$  usually base 2 or irrelevant in  $\Theta$ .

---

Open-notes exam aid · Lectures 1–8 · Formulas & templates above are meant to be copied · Verify base cases · Show  $c, n_0$  explicitly