The Simple AP® Computer Science Principles Big Idea 2 Cheat Sheet

2.1 Bits, Bytes & Abstraction

Digital Data

Bit (binary digit) = 0 or 1. Smallest piece of every value a computer stores; all data is bits.

Byte = 8 bits → 2^8 = 256 different patterns, from 00000000 to 11111111 (0–255 as numbers).

Abstraction (hiding detail): groups of bits stand for numbers, letters, colors, sound.

Context: same bits, new meaning: 01000001 = 65 as a number, or 'A' as a text letter.

How Many Values?

Bits	Patterns	Range
1	2	0–1
4	16	0–15
8	256	0–255
n	2^n	0 to 2^n–1

+1 bit → doubles the number of values. +2 bits → 4× as many values (not just 2 more).

Bits needed for 100 values → 7 bits, b/c 2^7 = 128 ≥ 100 but 2^6 = 64 is too few to cover them.

Common Encodings

Color (RGB): 8 bits each for red, green + blue → 24 bits per pixel; each color channel is 0–255.

⚠ The program's context (number, letter, color) decides how a bit pattern is read.

✓ "How many bits for N items?" → find the 1st power of 2 that is ≥ N. Never round down.

2.1 Binary ↔ Decimal (DAT-1.C)

Place Values

Decimal (base 10) uses digits 0–9. **Binary** (base 2) uses only 0 + 1; each place is a power of 2.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Binary → Decimal

Method: write place values under each bit; add up only the places that hold a 1. Skip the 0s.

Example: 101101 = 32 + 8 + 4 + 1 = **45**. Rightmost bit = the 1s place, next = 2s, then 4s.

Max value: n bits → all 1s = 2^n–1, e.g., 1111 = 15, 11111111 = 255. 16 needs 5 bits: 10000.

Odd or even? Last bit 1 → odd; last bit 0 → even. Rules out wrong answer choices fast.

Decimal → Binary

Subtract method: take biggest power of 2 that fits, write 1, subtract, repeat; 0 in each unused place.

Example: 77 = 64 + 8 + 4 + 1 → **1001101** (0s go in the 32, 16 + 2 places). Check by adding back.

Compare & Order

Ignore leading 0s. Longer number = bigger; same length → 1st different bit decides.

Append a 0 on the right → value doubles: 101 (5) → 1010 (10) → 10100 (20). Each new 0 = ×2.

⚠ Place values grow right → left. Start counting at the 1s place on the far right.

✓ Ordering MCQ: convert each choice to decimal on scratch paper, then compare them.

2.1 Overflow, Roundoff & Analog Data

Fixed Bits → Limits (DAT-1.B)

Fixed-size integers: many languages use a set # of bits → limited range of values.

Overflow error: answer needs more bits than exist. 4 bits: 15 + 1 → 0000 = 0.

Real-world bugs: counters + game scores that wrap back to 0 after hitting their max.

Roundoff error: real numbers like 1/3 or 0.1 stored only as approximations → small errors add up.

Exam Reference Language

AP pseudocode: integer size limited only by memory → no overflow on exam code.

Other languages w/ fixed bits can overflow. Check which kind of language the question uses.

Analog vs Digital

	Analog	Digital
Values	Smooth	Steps
Example	Music pitch	File bits
Exact?	No	Yes

Analog data changes smoothly: pitch + volume of music, colors in a painting, a runner's position.

Sampling: measure the analog signal at regular intervals → store each sample as bits.

Fix overflow: use more bits, or a language w/ no size limit (like AP pseudocode).

⚠ Overflow = number too BIG for the bits. Roundoff = number too PRECISE for the bits.

✓ Digital data only approximates analog data. More samples → closer, but bigger file.

2.2 Lossless vs Lossy Compression

Why Compress? (DAT-1.D)

Compression: fewer bits to store or send → saves storage space + sends files faster online.

Fewer bits ≠ less info: repeated (redundant) data can shrink w/ nothing lost, e.g., "100×white."

How much it shrinks depends on the data's **redundancy** + the **algorithm** used on it.

The Two Types

	Lossless	Lossy
Original?	Fully rebuilt	Approximation
Size cut	Smaller	Much smaller
Use when	Quality key	Size key
Examples	Text, code	Photos, music

Lossless: original data can always be rebuilt exactly, e.g., bank records, code.

Lossy: drops details people barely notice → much smaller files, but the lost detail can't be undone.

Lossy trade-off: more compression → lower quality, e.g., blurry, blocky photos.

Run-Length Encoding

RLE (lossless): count + value, e.g., AAAABBBCC → 4A3B2C. Decodes back exactly.

Weak case: ABCD → 1A1B1C1D gets **longer**. No repeats → nothing for RLE to save.

⚠ Lossy isn't "bad": streaming video uses it b/c small size matters more than detail.

✓ "Must restore exactly" → lossless. "Smallest file / fastest to send" → lossy compression.

2.3 Data → Information & Metadata

Information (DAT-2.A)

Information = facts + patterns pulled from data → spot trends, make connections, solve problems.

Correlation ≠ causation: 2 things rising together doesn't prove 1 causes the other.

Example: ice cream sales + drownings both rise in summer; hot weather drives both of them.

Data set size matters: bigger data sets can reveal more patterns + more useful information.

Combine sources: 1 data set often isn't enough → merge several sources to reach a conclusion.

Metadata (DAT-2.B)

Metadata = data about data. Photo = data; its date, place + file size = metadata.

Data	Its metadata
Photo	Date, GPS, size
Song	Artist, length
Email	Sender, time

Uses: find, organize + manage data, e.g., search photos by date or by location, songs by artist.

Privacy: metadata alone can reveal a lot, e.g., photo GPS → where someone lives.

Independence: changing or deleting metadata does NOT change the data itself.

⚠ "Found using ONLY metadata?" → only fields like date + size count, never the content.

✓ Before claiming 1 thing causes another, the right answer is "more research needed."

2.3 Data Challenges: Clean, Bias, Scale

Messy Data (DAT-2.C)

Challenges at any size: data to clean, **incomplete** data, **invalid** data, + combining several sources.

Non-uniform: users type the same thing differently, e.g., NY, N.Y., new york.

Cleaning: makes data uniform w/o changing its meaning (e.g., all of them → "New York").

Duplicates: same record entered twice → counted twice. Remove them when cleaning.

Invalid = impossible values (age −4, Feb 31).

Incomplete = missing fields or rows.

Bias

Bias comes from the type or source of data, e.g., an online-only survey misses people w/o internet.

More data ≠ less bias: more rows from the same biased source keep the same bias.

Fix bias: change how or where data is collected → reach a wider, fairer, more representative group.

Scale

Issue	Response
Huge data set	Parallel systems
Growing data	Scalable storage
Many sources	Combine + clean

Scalability: a system's computing power limits how much data it can process + store.

⚠ "Collect a larger sample" does NOT fix bias. Change where or how data is collected.

✓ Cleaning keeps the meaning the same. Deleting rows or changing values isn't cleaning.

2.4 Programs Process Data (DAT-2.D)

Four Core Operations

Process	Example
Transform	Double every item
Filter	Keep band kids
Combine	Sum, max GPA
Visualize	Chart, histogram

Transform: change every element, e.g., add a parent's email to every student record.

Filter: keep only rows that meet a condition, e.g., keep only orders over $50 → spot patterns.

Combine/compare: sum, count or average; find the max or min, e.g., the student w/ top GPA.

Visualize: charts, tables + diagrams share what the data shows w/ others clearly.

Tools

Search tools find info fast in big data sets (by keyword, field or date) w/o reading every row.

Spreadsheets: sort, filter + use formulas → find trends w/o writing a full program.

Can It Be Answered?

Method: list the columns given. If a needed field is missing → the question can't be answered.

Example: columns store, item, price. "Most-sold item?" ✓ "Buyer's age?" × (no age).

⚠ Filter first, then count or average, or your result mixes in the wrong groups.

✓ "Which can be determined?" → check every needed field against the column list.

2.4 Insight from Data (DAT-2.E)

Iterative Analysis

Iterative (repeat + refine): run a program, see the result, adjust it, run again → deeper insight.

Filter + clean first → reliable data → real insight. Garbage in → garbage out, every time.

Patterns emerge when data is sorted, grouped or graphed, e.g., sales by month → summer spike.

Combine, Cluster, Classify

Process	What it does
Combine	Merge sources
Cluster	Group similar
Classify	Label by category

Combining: join data sets on a shared field (like student ID) → answer brand-new questions.

Clustering: group similar items. **Classifying**: sort items into set labels (spam/not spam).

Translate: change how data is represented, e.g., GPS points → a map route → new insight.

Exam Moves

Code + list MCQ: trace the loop; track the result variable on every single pass of the loop.

Order of steps: which sequence (filter → sort → count) works? Test it on 2–3 rows.

Multi-select (pick 2): test each choice on its own; often 2 different methods both work.

⚠ Big Idea 2 = 17–22% of MCQs: binary math + data scenarios are fast, sure points.

✓ Create PT: your program's list = Big Idea 2 in action; explain how it manages complexity.