Study Guide: Divide and Conquer

Ashlin Govindasamy

October 2025

Formulas You Need (at a Glance)

General

$$\text{Floor division: } \left\lfloor \frac{a}{b} \right\rfloor \quad \text{(integer division)}$$

$$\text{Arithmetic series: } \sum_{j=a}^{b} j \;=\; \frac{(a+b)(b-a+1)}{2}, \quad a \le b$$

Q1 (Pricing)

$$f(K) = \sum_{j=1}^{K-1} j \cdot \left\lfloor \frac{K}{j} \right\rfloor$$

$$\text{Quotient (grouping) trick:} \quad \text{If } q = \left\lfloor \frac{K}{i} \right\rfloor, \text{ then for } j \in [i, r], \ \left\lfloor \frac{K}{j} \right\rfloor = q,$$

$$\text{where } r = \left\lfloor \frac{K}{q} \right\rfloor, \text{ and we clamp } r \le K-1.$$

$$\text{Block sum:} \quad \sum_{j=i}^{r} j \cdot q = q \cdot \sum_{j=i}^{r} j \;=\; q \cdot \frac{(i+r)(r-i+1)}{2}.$$

Q2 (Dividing planks)

$$\text{Pieces at length } M: \quad P(M) = \sum_{i=1}^{N} \left\lfloor \frac{L_i}{M} \right\rfloor$$

$$\text{Feasibility: } P(M) \ge K$$

$$\text{Search bounds: } 1 \;\le\; M \;\le\; \min\big(10{,}000{,}000,\ \max_i L_i\big).$$

Algorithm Notes (What to Prove / Remember)

Upper-Bound Binary Search (Max Feasible)

We want the largest integer x such that `feasible(x)` is true. Use:

```
long lo = MIN, hi = MAX;
while (lo < hi) {
    long mid = (lo + hi + 1) >>> 1; // upper mid, avoids infinite loop
    if (feasible(mid)) lo = mid;     // mid works: shift right
    else hi = mid - 1;               // mid fails: shift left
}
System.out.println(lo);
```

Invariants (maintained every loop):

- All values $\le lo$ are feasible.
- All values $\ge hi + 1$ are infeasible.
- Therefore the answer always lies in $[lo, hi]$.

Termination: The interval shrinks; with upper mid, `lo` strictly increases or `hi` strictly decreases. When `lo == hi` we have the maximal feasible.

Q1 (Pricing) Notes

Monotonicity (why binary search works): $f(K)$ is strictly increasing in K (given). Hence if $f(K) \le N$, any $K' < K$ also satisfies $f(K') \le N$.

Efficient $f(K)$ in $O(\sqrt{K})$:

1. Set $i \leftarrow 1$. While $i < K$:
   (a) $q \leftarrow \lfloor \frac{K}{i} \rfloor$ is the constant quotient for a block.
   (b) $r \leftarrow \left\lfloor \frac{K}{q} \right\rfloor$ is the last index with that quotient.
   (c) Clamp to $r \le K - 1$.
   (d) Add block: $q \cdot \dfrac{(i + r)(r - i + 1)}{2}$ using 64-bit.
   (e) Set $i \leftarrow r + 1$ and continue.
2. The number of distinct quotients is $O(\sqrt{K})$, so the whole evaluation is fast.

Overflow safety:

- Use `long` for all partials: `cnt = r-i+1`, `(i+r)*cnt`, and the product with q.
- Compute the arithmetic series as `((i + r) * cnt) / 2` in `long`.
- Optional early-exit: if `total > N` you can return "infeasible" quickly.

Binary search range: $1 \le K \le$ 1,000,000. Use `lo=1`, `hi=1_000_000`.

Edge cases:

- $K = 1 \Rightarrow f(1) = 0$ (empty sum). Works with $N \ge 1$.
- Ensure the loop excludes $j = K$ (sum is $j = 1..K - 1$).

Complexity:

$$\underbrace{O(\sqrt{K})}_{\text{for } f(K)} \times \underbrace{O(\log 10^6)}_{\text{binary search}} \quad \text{per test.}$$

Q2 (Dividing) Notes

Monotonicity (why binary search works): $P(M) = \sum \lfloor L_i / M \rfloor$ is non-increasing in M. If some M works ($P(M) \ge K$), then all $M' < M$ also work.

Feasibility check: Single pass:

1. Accumulate `pieces += Li / M` in `long`.
2. Early stop if `pieces >= K`.
3. Return `pieces >= K`.

Binary search range:

$$1 \le M \le \min\bigl(10{,}000{,}000,\ \max_i L_i\bigr).$$

You cannot cut pieces longer than the longest plank, and the problem caps the answer by 10^7.

Overflow safety & exactness:

- Use integer division only; no floating point is needed.
- Accumulator `pieces` must be `long` (since K can be up to 10^7).

Complexity:

$$\underbrace{O(N)}_{\text{feasibility pass}} \times \underbrace{O(\log 10^7)}_{\text{binary search}} \approx O(N \cdot 24).$$

Typical Pitfalls (Both Questions)

- Using lower-bound mid and getting stuck: always use upper-mid `(lo+hi+1)>>>1`.
- Overflow in arithmetic-series or products: cast to `long` before multiplying.
- Extra print text or missing newline: the marker expects *only* the integer and a trailing newline.
- For Q1, mistakenly summing up to $j = K$ instead of $K - 1$.
- For Q2, setting `hi` larger than $\max L_i$ (it doesn't break correctness if capped by 10^7, but it's cleaner and may prune sooner).

Micro Test Set (Hand Checks)

- **Q1**: Input `30` $\to$ Output `6`. Input `1` $\to$ `1`.
- **Q2**: N=4, $L = [10, 14, 15, 11]$, K=6 $\to$ Output `7`.
- **Q2 edges**: K=1 $\to$ `max(L)`. If $L = [1, 1, 1]$, K=3 $\to$ `1`.

Worked Calculations (Step by Step)

Q1 (Pricing): computing $f(K)$

Recall:

$$f(K) = \sum_{j=1}^{K-1} j \cdot \left\lfloor \frac{K}{j} \right\rfloor.$$

Direct small example ($K = 5$).

$$f(5) = 1 \cdot \lfloor 5/1 \rfloor + 2 \cdot \lfloor 5/2 \rfloor + 3 \cdot \lfloor 5/3 \rfloor + 4 \cdot \lfloor 5/4 \rfloor = 1 \cdot 5 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 1 = 16.$$

Check sample $N = 30$: compare $K = 6$ vs $K = 7$.

$$f(6) = 1 \cdot 6 + 2 \cdot 3 + 3 \cdot 2 + 4 \cdot 1 + 5 \cdot 1 = 6 + 6 + 6 + 4 + 5 = 27 \le 30 \quad \text{(feasible)},$$
$$f(7) = 1 \cdot 7 + 2 \cdot 3 + 3 \cdot 2 + 4 \cdot 1 + 5 \cdot 1 + 6 \cdot 1 = 7 + 6 + 6 + 4 + 5 + 6 = 34 > 30 \quad \text{(not feasible)}.$$

Hence the answer is $K = 6$.

Quotient-grouping example ($K = 20$). Use blocks where $q = \lfloor \frac{K}{j} \rfloor$ is constant for $j \in [i, r]$ with $r = \lfloor \frac{K}{q} \rfloor$ (clamp $r \le K - 1$). For each block, add

$$q \cdot \sum_{j=i}^{r} j = q \cdot \frac{(i + r)(r - i + 1)}{2}.$$

Concrete blocks for $K = 20$:

Block $[i, r]$	$q = \lfloor 20/i \rfloor$	count $(r - i + 1)$	$\sum_{j=i}^{r} j$	$q \cdot$ sum
$[1, 1]$	20	1	1	20
	10	1	2	20
	6	1	3	18
	5	1	4	20
	4	1	5	20
	3	1	6	18
	2	4	$7 + 8 + 9 + 10 = 34$	68
	1	9	$11 + 12 + \cdots + 19 = \frac{(11+19)\cdot 9}{2} = 135$	135

Total: $20 + 20 + 18 + 20 + 20 + 18 + 68 + 135 = 319 = f(20)$.

Binary search decision (illustrative). Suppose $N = 30$ and we probe $K = 6$ then $K = 7$ using the fast $f(K)$ above:

$$f(6) = 27 \le 30 \;\Rightarrow\; \text{move } lo \leftarrow 6; \qquad f(7) = 34 > 30 \;\Rightarrow\; \text{move } hi \leftarrow 6.$$

Stops with $lo = hi = 6$.

Q2 (Dividing planks): feasibility counts

Pieces at target length M:

$$P(M) = \sum_{i=1}^{N} \left\lfloor \frac{L_i}{M} \right\rfloor.$$

Sample instance. $L = [10, 14, 15, 11]$, $K = 6$. Upper bound $hi = \min(10{,}000{,}000,\ \max L_i) = 15$.

Test $M = 8$:

$$\left\lfloor \frac{10}{8} \right\rfloor = 1,\ \left\lfloor \frac{14}{8} \right\rfloor = 1,\ \left\lfloor \frac{15}{8} \right\rfloor = 1,\ \left\lfloor \frac{11}{8} \right\rfloor = 1 \Rightarrow P(8) = 4 < 6 \ \text{ (not feasible)}.$$

Test $M = 7$:

$$\left\lfloor \frac{10}{7} \right\rfloor = 1,\ \left\lfloor \frac{14}{7} \right\rfloor = 2,\ \left\lfloor \frac{15}{7} \right\rfloor = 2,\ \left\lfloor \frac{11}{7} \right\rfloor = 1 \Rightarrow P(7) = 6 \ge 6 \ \text{ (feasible)}.$$

Therefore the maximum feasible is $M = 7$.

Binary search move (upper-bound). If $M = 8$ fails, set $hi \leftarrow 7$; if $M = 7$ works, set $lo \leftarrow 7$. End with $lo = hi = 7$.

Pseudocode (Clear and Concise)

Reusable upper-bound binary search

```
lo = MIN
hi = MAX
while (lo < hi):
    mid = floor((lo + hi + 1) / 2)   // or (lo+hi+1)>>>1 in Java
    if feasible(mid):
        lo = mid
    else:
        hi = mid - 1
answer = lo
```

Q1 (Pricing): $f(K)$ via grouping + feasibility

```
// Returns f(K) in O(sqrt(K))
long f(long K):
    if K <= 1: return 0
    total = 0L
    i = 1L
    while i < K:
        q = K / i                  // integer division
        r = K / q                  // last j with this quotient
        if r >= K: r = K - 1       // j only up to K-1
        cnt = r - i + 1            // number of terms in block
        // sum i..r = (i + r) * cnt / 2  (all in 64-bit)
        segmentSum = ((i + r) * cnt) / 2
        total += q * segmentSum
        i = r + 1
    return total

// Binary search driver: max K with f(K) <= N
long solvePricing(long N):
    lo = 1L
    hi = 1_000_000L
    while (lo < hi):
        mid = (lo + hi + 1) >>> 1
        if f(mid) <= N:
            lo = mid
        else:
            hi = mid - 1
    return lo
```

Notes:

- All intermediates in `long`.
- Off-by-one: clamp $r \leq K - 1$.
- Optional pruning: if `total > N`, you can early-return a sentinel >N.

Q2 (Dividing): feasibility + driver

```
// Is length M feasible? (sum floor(Li/M) >= K)
boolean feasible(long[] L, long K, long M):
    if M <= 0: return false
    pieces = 0L
    for each x in L:
        pieces += x / M
        if pieces >= K: return true   // early stop
    return (pieces >= K)
```

```
// Binary search driver: max M with pieces >= K
long solveDividing(long[] L, long K):
    maxL = max(L)
    lo = 1L
    hi = min(10_000_000L, maxL)
    while (lo < hi):
        mid = (lo + hi + 1) >>> 1
        if feasible(L, K, mid):
            lo = mid
        else:
            hi = mid - 1
    return lo
```

Notes:

- No sorting or floating point needed.
- `pieces` must be `long`.
- Bound `hi` by $\max L_i$ and 10^7.

Complexity Summary

- **Q1**: each $f(K)$ in $O(\sqrt{K})$; binary search adds $O(\log 10^6)$.
- **Q2**: each feasibility check in $O(N)$; binary search adds $O(\log 10^7) \approx 24$.