香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

J263 - Spot-Check Debugging

Daniel Hsieh {QwertyPi}
2026-02-14

Background

Problem idea by kctung

Preparation by __declspec, gasbug

Problem Restatement

- Bob needs to debug L lines of code with N buggy sections
- Each day he checks lines at offsets t, t+S, t+2S, ... (step size S)
- Once he locates a buggy line X, he correct lines X, X-1, ... until a correct line
- The starting offsets for each day are T[1], T[2], ..., T[P], T[1], ...
- Answer Q queries: for each day D, how many lines are debugged?

Subtasks

Task	Attempts	Max	Mean	Std Dev	Subtasks					
J263 - Spot-Check Debugging	73	100	15.78	18.248	10: 54	8: 25	13: 13	19: 9	22: 2	28: 1

First (and only) solved by **wy_24215** (Yang Chun Kit) at **2hr 59m 27s**

Subtask Constraints

Subtask	Points	Constraints
1	10	$L \le 100$ $D_Q \le 100$
2	8	$P = 1$ $U[i] = V[i]$ for $1 \le i \le N$
3	13	$P = 1$ $V[i] - U[i] + 1 \le S$ for $1 \le i \le N$
4	19	$P = 1$
5	22	$N = 1$
6	28	No additional constraints

Subtask 1 (10%): L ≤ 100, D[Q] ≤ 100

Store correctness of every line and simulate

Complexity: O(L) per day, O(D[Q]) days

Expected score: 10 (Cumulative: 10)

Subtask 2 (8%): P = 1, U[i] = V[i]

- Buggy sections are single lines
- Same starting offset every day

- Compute the number of lines that matches the only offset (mod S)
- For each query day, check if enough complete cycles have passed

Complexity: O(N + Q)

Expected score: 8 (Cumulative: 18)

Subtask 3 (13%): P = 1, V[i] - U[i] + 1 ≤ S

- Small buggy section size
- Same starting offset every day

- Key insight: Each section can only be debugged once
- Precompute: For every section, whether it will be hit (how?) and if so, the number of lines fixed
- Store the hit sections in an array and answer queries by indexing it

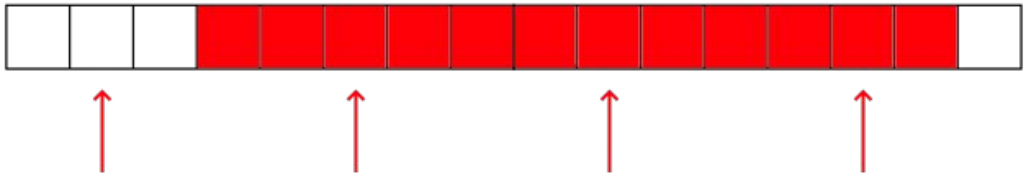
Complexity: O(N + Q)

Expected score: 21 (Cumulative: 31)

Subtask 4 (19%): P = 1

- Same starting offset

- Observation: When we fix a new section, the first day we might fix any number of lines, but from the second day onwards we only fix S lines until we can't fix any lines in the section anymore
- E.g. S=4, T[1]=2: for the red buggy section, we fix 3 lines, then 4 lines, then 4 lines, then leave this section.



Subtask 4 (19%): P = 1

- Fast forwarding
- Iterate on the days
- Track current section index being debugged
- To fast forward d days, simply progress through each section by the previous observation until we run out of days

Complexity: O(N + Q) (won't visit the same section twice)

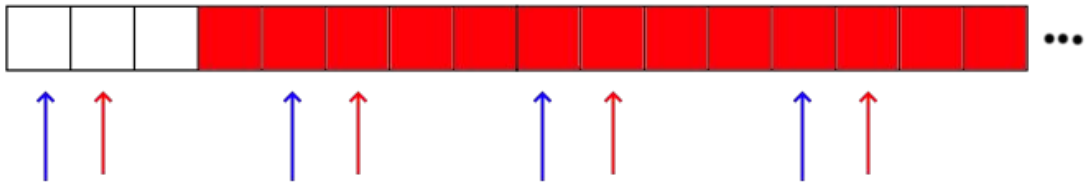
Expected score: 40 (Cumulative: 50)

Subtask 5 (22%): N = 1 (single buggy section)

Stronger Observation: consider each P days as a cycle. The number of lines fixed in each cycle is constant per day in cycle, except the first one. Furthermore, except the first cycle, the total number of lines fixed is a multiple of S.

E.g. if D[1]=2 (red), D[2]=1 (blue), S=4, then

- 1st cycle: R fixes 3 lines, B fixes 3 lines
- 2nd cycle: R fixes 1 line, B fixes 3 lines (total = 4 is a multiple of S)
- ...



Subtask 5 (22%): N = 1 (single buggy section)

- Calculate the lines fixed for the first cycle
- Do it again for the cycles from the second one onward
- Calculate the number of full cycles and handle the ending case carefully in O(P)

Time complexity: O(P+Q)

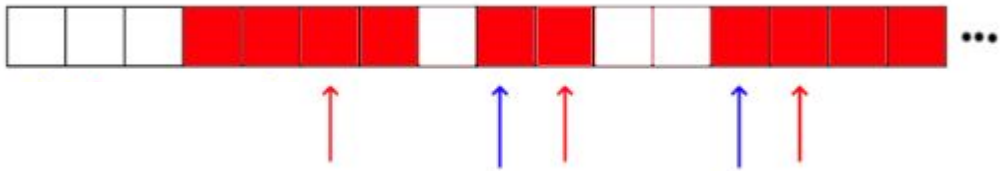
Expected score: 62 (Cumulative: 72)

Full Solution

Notation: Consider the j-th cycle (of each P days). Let the index of the section debugged on the i-th day in the cycle be $F_j[i]$.

E.g. Red = D[1] = 2, Blue = D[2] = 1:

- F_1 = [1, 2]
- F_2 = [2, 3]
- ...



Full Solution

We can do simulation.

- When we simulate day i in the j-th cycle, we set $F_j[i] := F_{j-1}[i]$ initially, and increase it until we hit a buggy section that needs debugging.

However, this is too slow. We can speed up by looking at two cases:

1. We debug the same section for every day in the cycle, i.e. $F_j[1] = \ldots = F_j[P]$.
   → Use subtask 5 to speed up in O(P).
2. We don't debug the same section. This happens at most O(N) times, since the $\min(F_j)$-th section will not be debugged anymore after the next cycle.

Full Solution

> If we don't debug the same section among all days in the cycle j, the $\min(F_j)$-th section will not be debugged anymore after the next cycle.

Why? (Proof Sketch)

- Consider the section size of the $\min(F_j)$-th section after the j-th cycle
- It must be $< S$
- Otherwise, we will debug the same section in cycle j (why?)

Full Solution

So for each day in query, compute the respective F array by fast forwarding + simulation.

- Fast forwarding at most O(N+Q) times, each time just O(1)
- Simulation at most O(N) times, each time amortised O(P)
- Overall time complexity: O(N*P+Q)

Expected score: 100

Implementation Note

- Since the modulo operation should be involved, it is easier to code if indices are 0-based
- So on day i:
  - starting offset = T[i%P]
  - only lines j where j%S == T[i%P] are checked