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

2026 Mini Competition 2 Editorial

2026-03-21

Statistics

Task	Attempts	Max	Mean	Std Dev	Subtasks
M2621 - Fan-Out Tree	70	100	41.228	26.269	23: 64 | 18: 45 | 33: 12 | 26: 8
M2622 - Novel Epilogue	74	100	51.391	35.406	11: 68 | 19: 60 | 16: 31 | 14: 30 | 17: 29 | 23: 22
M2623 - Clearing Pests	78	100	96.846	15.192	7: 77 | 12: 76 | 14: 76 | 31: 75 | 25: 76 | 11: 74
M2624 - Water Cups	68	100	21.088	22.012	6: 58 | 8: 52 | 11: 31 | 17: 4 | 29: 5 | 20: 4 | 9: 4
	78	**400**	**200.987**	**77.002**	

Remarks

- Even if you get accepted in a task, you should still listen to the editorial
- Some contestants overcomplicated the tasks, including but not limited to:
  - Using virtual tree instead of trie in Q1
  - Using segment tree instead of array in Q2

M2621
Fan-Out Tree

The Problem

- A Fan-Out Tree of N layers is defined as follows:
  - Layer 1 has one node as the root
  - Each node on layer i has C[i] children
- Each node can be indexed by the path from the root
- K edges are now added onto the tree
- Find the minimum distance between node X and node Y

Subtask 1 (23%, K = 0, Total length of indices of all nodes ≤ 1000)

- No edges are added
- There is a unique path from X to Y
  - X -> LCA(X, Y) -> Y
- How do we find the LCA?
- LCA(X, Y) is the node represented by the longest common prefix of the indices of X and Y
- The distance between X and Y = dep(X) + dep(Y) - dep(LCA(X, Y))
- Time complexity: O(N)

Subtask 2 (18%, C[i] = 1, Total length of indices of all nodes ≤ 2×10^5)

- The tree before modification is a chain
- Since there are not a lot of nodes (there are at most ~630 nodes), we can build the entire graph directly and run BFS on the graph
- Time complexity: O(N(N + K))

Subtask 3 (33%, Total length of indices in the input ≤ 1000)

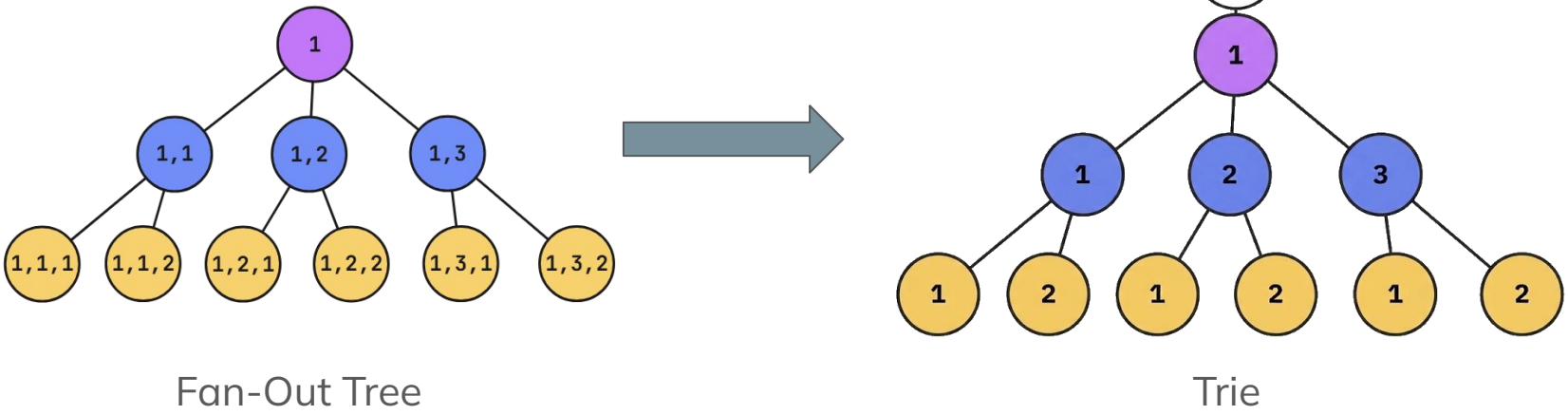
- We are not guaranteed that the graph has less enough nodes to be built
- Notice that only X, Y, the endpoints of the K edges and their ancestors are relevant here
- We can build the graph with these nodes and their ancestors and perform BFS on the graph
- Time complexity: O(NK)

Subtask 4 (26%, Total length of indices in the input ≤ 2×10^5)

- What is the bottleneck for the time complexity?
- We are storing the entire index for each node, hence it takes O(N) to push a node into the queue when we perform the BFS
- Can we make it faster?

Subtask 4 (26%, Total length of indices in the input ≤ 2×10^5)

- We can use a Trie to store the nodes of the graph
- For each relevant node, we insert its index into the trie
  - This essentially also inserts all of its ancestors into the trie



Fan-Out Tree

Trie

Subtask 4 (26%, Total length of indices in the input ≤ 2×10^5)

- We can use a Trie to store the nodes of the graph
- For each relevant node, we insert its index into the trie
  - This essentially also inserts all of its ancestors into the trie
- By the constraints of the task, there can be at most 2×10^5 nodes in the trie
- We can assign an ID for each node in the trie, and perform the solution for subtask 3 with the assigned IDs
- Time complexity: $O(K)$

M2622
Novel Epilogue

Background

Problem Idea and Preparation by `QwertyPi`

Presentation by `QwertyPi`

Fun Fact: This task is inspired by the light novel あそびのかんけい

The author writes 17 pages of epilogue in the first volume!

The Problem

Given a novel of N chapters where the i-th chapter has P[i] pages.

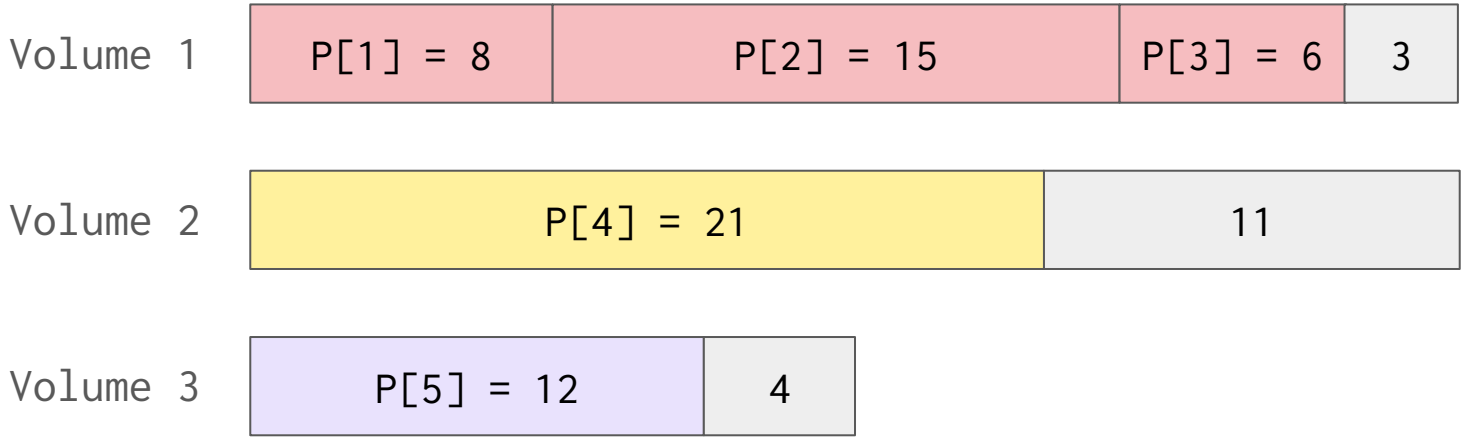
Partition the chapters into volumes so that each volume contains ≤ V pages.

Each volume is padded with epilogue so the number of pages is multiple of 16.

Minimise the total number of pages of epilogue.

Example

```
N = 5    V = 32
P[] = {8, 15, 6, 21, 12}
```



Subtasks

Subtask	Score	N	Additional Constraints
1	11	≤ 500	P[i] > N / 2
2	19	≤ 5000	P[i] mod 16 = 8
3	16	≤ 500	/
4	14	≤ 5000	/
5	17	$\leq 2 \times 10^5$	P[i] mod 16 all equal
6	23		/

Subtask 1 (11%, N ≤ 500, P[i] > V / 2)

Observation. Each volume consists of one chapter only.
So the answer is simply the sum of pages of epilogue over all volumes.

What is the number of epilogue pages for a volume of `X` content pages?

- `(16 - X % 16) % 16`
- Aware of edge cases for modulo!

Expected Score: 11 (Cumulative: 11)

Subtask 2 (19%, N ≤ 5000, P[i] mod 16 = 8)

Now, the length of epilogue in each volume is either 0 pages or 8 pages.

In fact, we can greedily make volumes with no epilogue

- Why? It is always good to consume some chapters without cost!

Therefore, we can simply process the chapters one-by-one:

- If `P[i] + P[i + 1] ≤ V`, then form a volume with chapter `i` and `i + 1`
- Otherwise, form a volume with chapter `i` alone and cost 8 pages

Expected Score: 19 (Cumulative: 19)

Subtask 3 (16%, N ≤ 500)

This task is actually in a very standard DP formulation.
Let `dp[R]` be the minimum number of epilogue pages if we form volumes on chapters 1 to `R`, where `dp[0] = 0`.
Then, we can transit from `dp[L]` to `dp[R]` for `L < R` whenever

```
X := P[L + 1] + P[L + 2] + ... + P[R] ≤ V
```

and update

```
dp[R] := min(dp[R], dp[L] + (16 - X % 16) % 16)
```

There are $O(N^2)$ pairs of `(L, R)`, so this gives us a $O(N^3)$ solution.
Expected Score: 27 (Cumulative: 46)

Subtask 4 (14%, N ≤ 5000)

We can in fact optimise the page count summation with partial sum.

Let `ps[R]` be the number of pages from chapter 1 to chapter `R`.

The transition is optimised to

$$\texttt{dp[R]} = \min_{\texttt{L<R,ps[R]-ps[L]}\le\texttt{V}}(\texttt{dp[L]} + (16 - (\texttt{ps[R]} - \texttt{ps[L]}) \bmod 16) \bmod 16)$$

Time Complexity: $O(N^2)$

Expected Score: 60 (Cumulative: 60)

Subtask 5 (17%, P[i] mod 16 are all equal)

For this constraint, every consecutive 16 chapters form a volume without epilogue. What does this imply?

Observation. If we transit from `dp[L]` to `dp[R]` where `R - L > 16`, then we can transit from `dp[L]` to `dp[R - 16]`, and from `dp[R - 16]` to `dp[R]` instead with exactly same cost.

Therefore, to compute `dp[R]`, we can restrict the possible `L` to between `R - 16 ... R - 1`!

Time Complexity: O(16N)

Expected Score: 36 (Cumulative Score: 77)

Full Solution (DP Optimisation)

Observation. If we transit from `dp[L]` to `dp[R]` **where `R - L > 16`**, then we can transit from `dp[L]` to `dp[R - 16]`, and from `dp[R - 16]` to `dp[R]` instead with exactly same cost.

^ Can we generalise the above observation for subtask 5?

The idea of “splitting” a volume to two volumes sounds useful...

Full Solution (DP Optimisation)

Observation. If we transit from `dp[L]` to `dp[R]` **where there exists a `L < M < R` where `ps[L] % 16 = ps[M] % 16`**, then we can transit from `dp[L]` to `dp[M]` and from `dp[M]` to `dp[R]` instead with exactly same cost.

Therefore, we can simply maintain for each remainder their last occurrence, and only transit from those indices.

Time complexity: O(16N)
Expected Score: 100 (Cumulative: 100)

Alternative Solution (Greedy)

Alternatively, this task can actually be solved with greedy:

- Each step, we greedily form the volume with shortest possible epilogue.
- This can be computed by simply loop through the remainders one-by-one.

Proof: Suppose more optimal transition is `L -> R` and our transition is `M -> R`.

- If `M ≤ L`, then `dp[M] ≤ dp[L] + 15`, implying our transition is not worse.
- Otherwise, if `M > L`, then `L -> M -> R` should have same cost as `L -> R`.

Time Complexity: O(16N)

Expected Score: 100 (Cumulative: 100)

Remarks

- Know and learn standard techniques! Those often give you a base score
- To go beyond, you may think about how the task deviate from the standards, and look for task-specific observations
- Even sometimes you can solve the task without some observations, very often those observations would simplify your implementation by much, for example - you should really not use segment tree to solve this task(!)

- What if the modulo is not 16 but arbitrary instead?
- Turns out this is still solvable in O(N log N)!

M2623
Clearing Pests

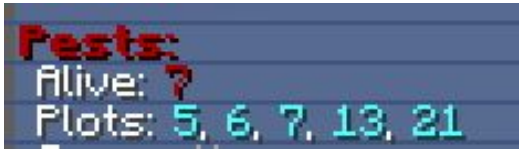
Background

Problem Idea and Preparation by WongChun1234

Presentation by WongChun1234

Inspired by Hypixel Skyblock

Note: do not get into this game, it is quite (very) addicting



In game screenshots

Problem

Given there are N plots and M pests, each plot can have:

- No pests
- Exactly X_i pests
- At least 1 pest

How many possible pest distributions are there?

- None (contradicts with M pests in total)
- One (uniquely determine)
- Multiple

Subtask 1 (7 marks): $N = 1, X_1 \neq -1$

- According to the record, there are X_1 pests in plot 1
- There should be M pests in total (= in plot 1)

- $X_1 = M \Rightarrow$ uniquely determine
- $X_1 \neq M \Rightarrow$ contradiction

Subtask 2 (12 marks): N = 1

- If $X_1 = 1$, there are at least 1 pest in plot 1
- All pests must be in plot 1 anyways as N = 1

- $X_1 \neq -1$
  - $X_1 = M \Rightarrow$ uniquely determine
  - $X_1 \neq M \Rightarrow$ contradiction
- $X_1 = -1$
  - $M \geq 1 \Rightarrow$ uniquely determine
  - $M = 0 \Rightarrow$ contradiction

Subtask 3 (14 marks): $X_i \neq -1$ for all i

- According to the record, there are $X_1 + X_2 + \ldots + X_N$ pests in total
- There should be M pests in total

- $\sum X_i = M \Rightarrow$ uniquely determine
- $\sum X_i \neq M \Rightarrow$ contradiction

Subtask 4 (31 marks): $X_i = -1$ for exactly one i

- Let $X_k = -1$
- There should be at least $\sum_{i \neq k} X_i + 1$ pests in total

- $M \geq \sum_{i \neq k} X_i + 1 \Rightarrow$ uniquely determine
- $M < \sum_{i \neq k} X_i + 1 \Rightarrow$ contradiction

Subtask 5 (25 marks): $N = 2, X_1 = X_2 = -1$

- There should be at least 2 pests in total
- If there are more than 2 pests, there are at least 2 distributions
- E.g. {1, M - 1}, {M - 1, 1}

- $M > 2 \Rightarrow$ multiple
- $M = 2 \Rightarrow$ uniquely determine
- $M < 2 \Rightarrow$ contradiction

Subtask 6 (25 marks): No additional constraints

- Let M' be the number of pests in unknown plots ($M' = M - \sum_{X_i \neq -1} X_i$)
- Similarly, let N' be the number of unknown plots

- M' < N' ⇒ contradiction (each unknown plot must have at least 1 pest)
- M' = N' ⇒ uniquely determine (each unknown plot get 1 pest)
- M' > N' ⇒
  - N' = 0 ⇒ impossible (nowhere to put the extra pests)
  - N' = 1 ⇒ uniquely determine (one unknown plot takes all extra pests)
  - N' > 1 ⇒ multiple

M2624
Water Cups

Problem

Given a rooted tree with N nodes, process Q operations of the following 2 types:

1. Find the K-th element in the post-order of the subtree rooted at node U, considering only nodes that are reachable from U by passing through unblocked edges. (It is guaranteed that K is less than or equal to the number of reachable nodes from U)
2. Toggle the state of an edge (blocked -> unblocked, vice-versa)

Subtask 1 (6 marks): N,Q ≤ 1000, P[i] = i-1 for all 2 ≤ i ≤ n

- It is guaranteed that the tree is a chain. N and Q are rather small.
- For each type 1 operation, naively use a for loop to find the next blocked edge starting from node U.
- Let v be the last node before a blocked edge, the answer we wanted is then (v - K + 1).

Time Complexity: O(NQ)

Expected Score: 6

Subtask 2 (8 marks): N,Q ≤ 1000

- N and Q are rather small
- For each type 1 operation, perform a DFS on the current state of the tree to find the post-order starting from U.
- The answer is the K-th element in the post-order

Time Complexity: O(NQ)

Expected Score: 14

Subtask 3 (11 marks): N,Q ≤ 200000, T[i] = 1

- No type 2 operations (edge will not be blocked).
- Let post[i] be the i-th element in the post-order of the tree.
- Let st[u] and ed[u] be the starting position and the ending position of the subtree of u in the post-order (post[st[u]..ed[u]] contain only and all nodes inside the subtree of u).
- Precompute post[i], st[u] and ed[u] with DFS.
- For each query (U, K), the answer is post[st[U] + K - 1]

Time Complexity: O(N+Q)

Expected Score: 11

Subtask 4 (17 marks): N,Q ≤ 200000, K[i] = 1

Let dist[u] be the number blocked edges on the path from the root to u.

Observation: For all nodes v inside the subtree of node u,

- If v is reachable from u, dist[u] = dist[v]
- If v is not reachable from u, dist[u] < dist[v]

In this subtask, K=1. Our operations can hence be reformulated as:

1. Find node v within the subtree of U with the minimum dist. If multiple such nodes exist, choose the one that appears first in the post order.
2. Increase (Block) / Decrease (Unblock) the dist of all nodes within the subtree of V.

Subtask 4 (17 marks): N,Q ≤ 200000, K[i] = 1

- Convert the problem into a range update range query problem by building a segment tree on the post-order.
- A leaf in the segment tree stores a pair of integers, the dist of the corresponding node and the order of the node in the post-order.
- Each node in the segment tree stores the minimum pair among the range it represents.
- Type 1 operations: find the minimum within the range [st[u], ed[u]].
- Type 2 operations: increase/decrease dist within the range [st[u], ed[u]] with lazy propagation.

Time Complexity: O((N+Q)logN)
Expected Score: 17

Subtask 5 (29 marks): N,Q ≤ 200000, U[i] = 1

- Instead of blocking/unblocking the edges on the original tree, block/unblock the edges on the segment tree
- Introduce a weight to each edge in the segment tree, corresponding to the number of blocked edges on the original tree that blocks this segment tree edge.
- For type 2 operations, increment/decrement the weights of the segment tree edges that connect the nodes that represent the range [st[u], ed[u]] to their corresponding parents.

Subtask 5 (29 marks): N,Q ≤ 200000, U[i] = 1

- In this subtask, since U = 1, all reachable nodes do not pass through any edge with weight > 0 in the segment tree.
- Every node in the segment tree stores the number of leaves that can be reached from the current node by passing through only edges with weight 0.
- For type 1 operations, perform binary search:
  - Check how many nodes are reachable from U = 1 within the range post[1..mid] using the segment tree.
  - When we are traversing the segment tree, return if we encounters a segment tree edge with positive weight.

Time complexity: O(NlogN+Qlog^2N)
Expected Score: 29

Subtask 6 (20 marks): N,Q ≤ 200000

- From subtask 4, we have the observation that all reachable nodes must have the same dist as U.
- For type 1 operations, we first find dist[U] by traversing down the segment tree and summing up the weights of the edge passed from the root to the leaf representing U.
- Then, we perform the same binary search as that in subtask 5.
- When traversing down the segment tree, instead of returning when we encounter an edge with positive weight, return when the sum of edge weights from the root to the current node is greater than dist[U].

Time Complexity: O(NlogN+Qlog^2N)
Expected Score: 91

Subtask 7: No additional constraints

- Get rid of the extra log by performing binary search on segment tree.
- Remember to use fast-IO for large inputs.

Time Complexity: O((N+Q)logN)

Expected Score: 100