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

T26G4 - Mountains

Kodi Siu {bedrockfake}
2026-03-28

Background

Problem Idea by `WongChun1234`

Preparation by `bedrockfake`

Problem Statement

- Input gives you M mountains placed on a N x N grid where the i-th mountain is placed on (X[i], Y[i]) with height H[i]
- Each cell (u, v) on the grid has height H(u, v) = max(H[i] - d(i)) among all mountains i, where d(i) is the manhattan distance between (u, v) and the i-th mountain
- Find an optimal height k such that sum(|H(u, v) - k|) is minimised

Statistics

T26G4 - Mountains	12	49	16	16.733	11: 8	22: 4	16: 1	39: 0	12: 0

Subtask 1 (11%): N, M, H[i] ≤ 100

Observation 1: The optimal k must satisfy k ≤ max(H[i]).

- |H(u, v) - k| will only increase for k > max(H[i]).

- For each cell (u, v), iterate all mountains to calculate its height H(u, v)
  - $O(N^2M)$
- Try all possible k from 1 to max(H[i]), compute sum(|H(u, v) - k|) for all N x N cells
  - $O(N^2\max(H[i]))$
- Total complexity: $O(N^2(M + \max(H[i])))$
- Expected Score: 11

Subtask 2 (22%): N, M ≤ 100

Observation 2: The optimal k is equal to one of the **medians** of the N^2 heights H(u, v).
(*medians: both the $N^2/2$-th and $(N^2+1)/2$-th greatest height for even N)

- When k is equal to one of the medians: As k increases or decreases, there will be at least $\lceil N^2/2 \rceil$ elements (a majority) where |H(u, v) - k| increases, so sum(|H(u, v) - k|) also increases.

- For each cell (u, v), iterate all mountains to calculate its height H(u, v)
  - $O(N^2M)$
- ~~Try all possible k from 1 to max(H[i]), compute sum(|H(u, v) - k|) for all N x N cells~~
- Sort the heights and find one of the medians, compute sum(|H(u, v) - k|) for all N x N cells
  - $O(N^2 \log N^2)$
- Total complexity: $O(N^2(M + \log N^2))$
- Expected Score: 33

Subtask 3 (16%): N, M ≤ 2000

~~For each cell (u, v), iterate all mountains to calculate its height H(u, v)~~

1. Propagate once from top-left to bottom-right:
   - H(u, v) = max(H(u, v), H(u - 1, v) - 1, H(u, v - 1) - 1)

   Propagate again from bottom-right to top-left:
   - H(u, v) = max(H(u, v), H(u + 1, v) - 1, H(u, v + 1) - 1)
2. Alternatively run Dijkstra to compute H(u, v)
   - $O(N^2)$ or $O(N^2 \log N^2)$ depending on implementation

- Sort the heights and find one of the medians, compute sum(|H(u, v) - k|) for all N x N cells
  - $O(N^2 \log N^2)$

- Total complexity: $O(N^2 \log N^2)$
- Expected Score: 49

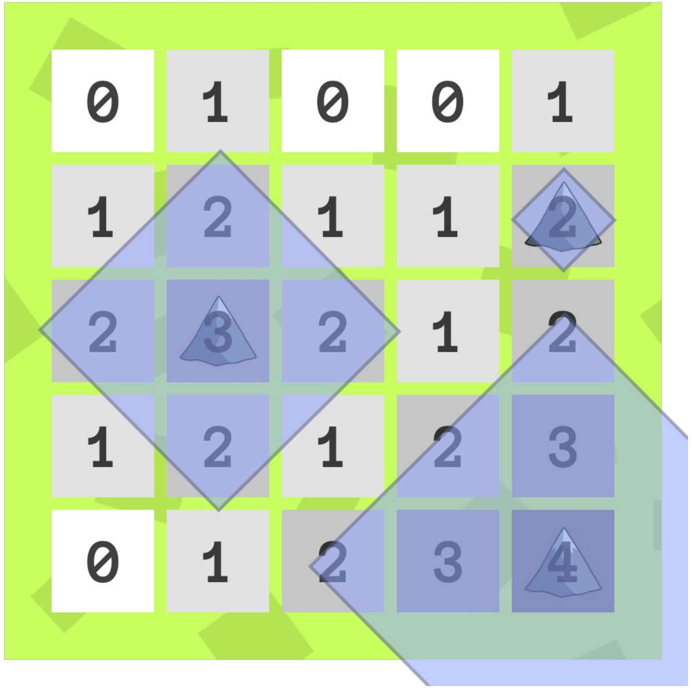
Subtask 4 (39%): Boundary cells have H(u, v) = 0

Idea: Binary search on optimal answer k to find the median

- If at least $N^2/2$ cells have $H(u, v) \geq k$, then median $\geq k$
- If less than $N^2/2$ cells have $H(u, v) \geq k$, then median $< k$

Problem is reduced to calculating no. of cells with $H(u, v) \geq k$.

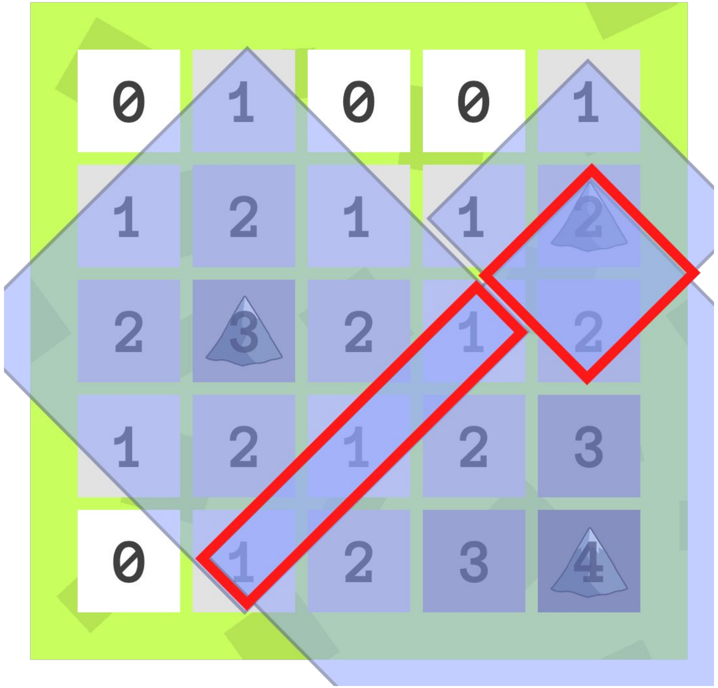
Subtask 4 (39%): Boundary cells have H(u, v) = 0



Observation 3: Cells where H(u, v) ≥ k form orthogonal squares at 45° on the grid.
For every mountain i, cells that are d(i) away will have H(u, v) ≥ H[i] - d(i), forming an orthogonal square of size d(i)+1.

Can we just add up the areas?

Subtask 4 (39%): Boundary cells have H(u, v) = 0

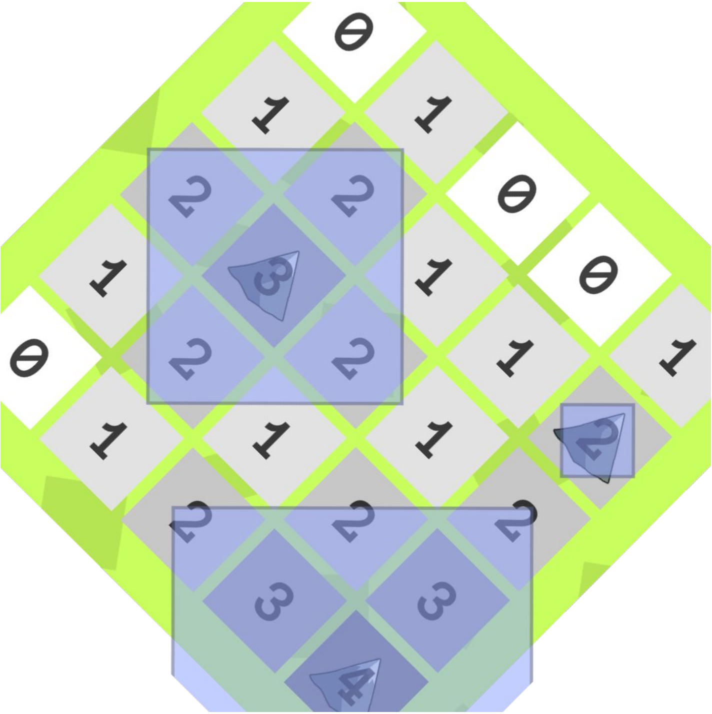


Observation 3: Cells where H(u, v) ≥ k form orthogonal squares at 45° on the grid.
For every mountain i, cells that are d(i) away will have H(u, v) ≥ H[i] - d(i), forming an orthogonal square of size d(i)+1.

NO!

- How to handle overlaps?
- Isn't this a familiar problem?

Subtask 4 (39%): Boundary cells have H(u, v) = 0

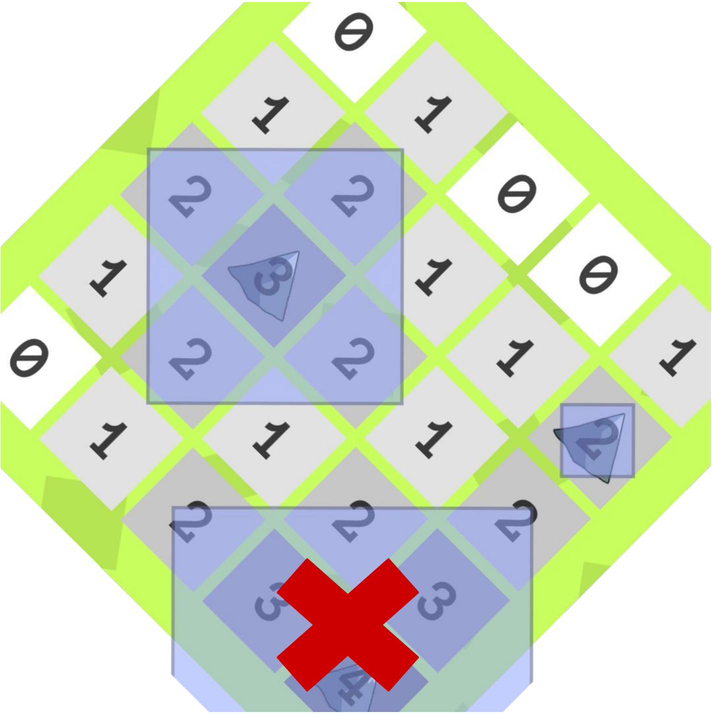


Observation 3: Cells where H(u, v) ≥ k form orthogonal squares at 45° on the grid.
For every mountain i, cells that are d(i) away will have H(u, v) ≥ H[i] - d(i), forming an orthogonal square of size d(i)+1.

Rotate the grid by 45°: the problem simplifies to finding area of rectangles

- M1633 Area of Rectangles
  - Classic problem solved by Sweep Line + Lazy Segment Tree

Subtask 4 (39%): Boundary cells have H(u, v) = 0

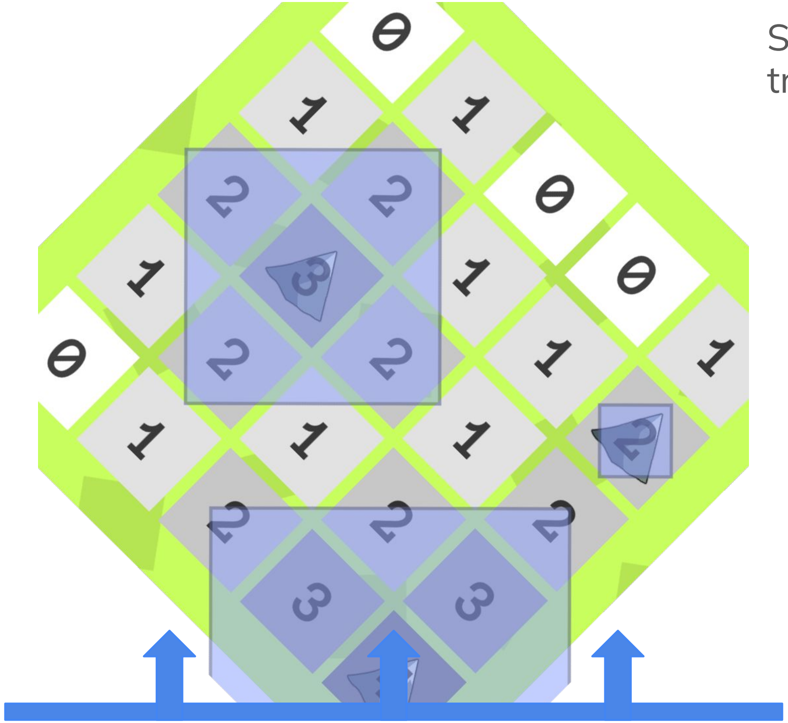


Observation 3: Cells where H(u, v) ≥ k form orthogonal squares at 45° on the grid.
For every mountain i, cells that are d(i) away will have H(u, v) ≥ H[i] - d(i), forming an orthogonal square of size d(i)+1.

Condition: Boundary cells have H(u, v) = 0

- Orthogonal squares will be **fully** in the grid

Subtask 4 (39%): Boundary cells have H(u, v) = 0



Sweep in one direction and maintain lazy segment tree with discretization

- For each node[x] representing range [l, r], maintain `count[x]` = sum(no. of mountain intervals that include i) for i in range [L[l], L[r]-1] (L stores the actual coordinates)

- Calculate `ans[x]`:

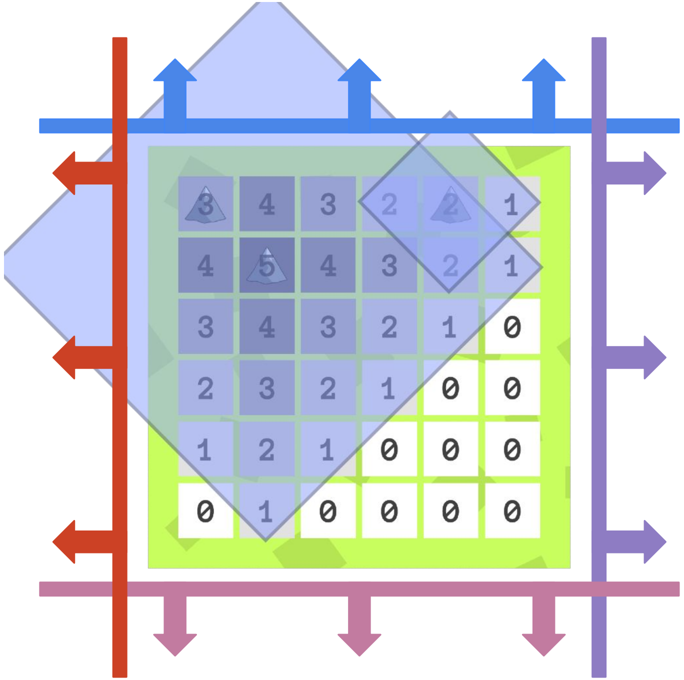
```
if (count[x] > 0) {
    ans[x] = L[r]-L[l]
} else {
    ans[x] = ans[x*2] + ans[x*2+1]
}
```

Subtask 4 (39%): Boundary cells have H(u, v) = 0

- Binary search on optimal answer k to find the median
- Rotate grid, then Sweep line + Lazy Segment Tree to compute area of rectangles
    - Have to discretize the coordinates
    - May need to split parity due to diagonal grid

- Total complexity: O(M logM log(max(H[i])))
- Expected Score: 39 (Cumulative: 88)

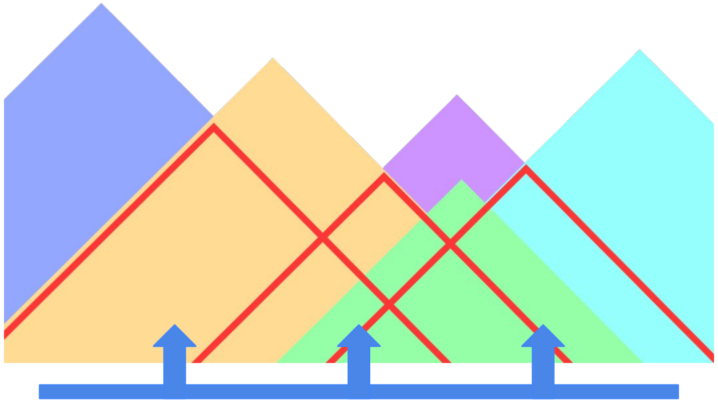
Subtask 5 (12%): No additional constraints



Now we have to handle out of bound orthogonal squares

- Calculate area of rectangles then subtract no. of cells out of bound
- Consider splitting "out of bound" areas into 4 directions

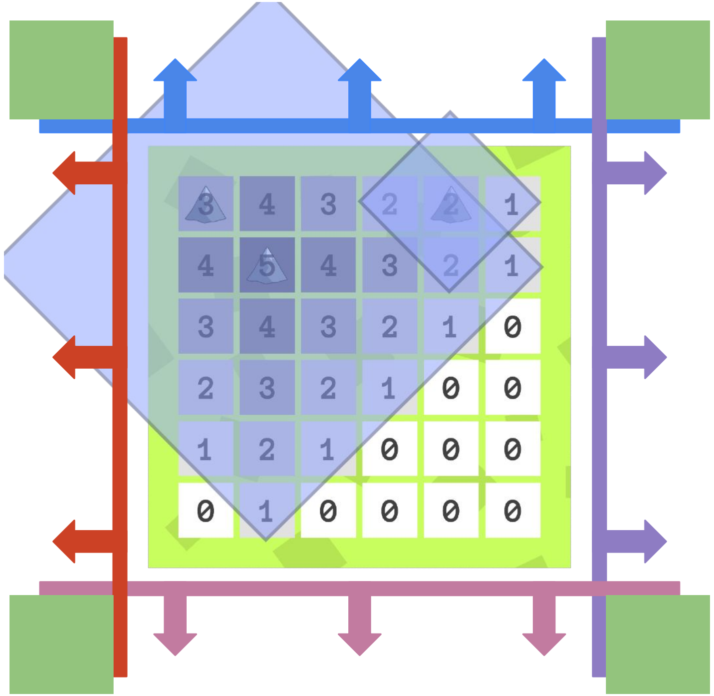
Subtask 5 (12%): No additional constraints



For each side, find the unioned area

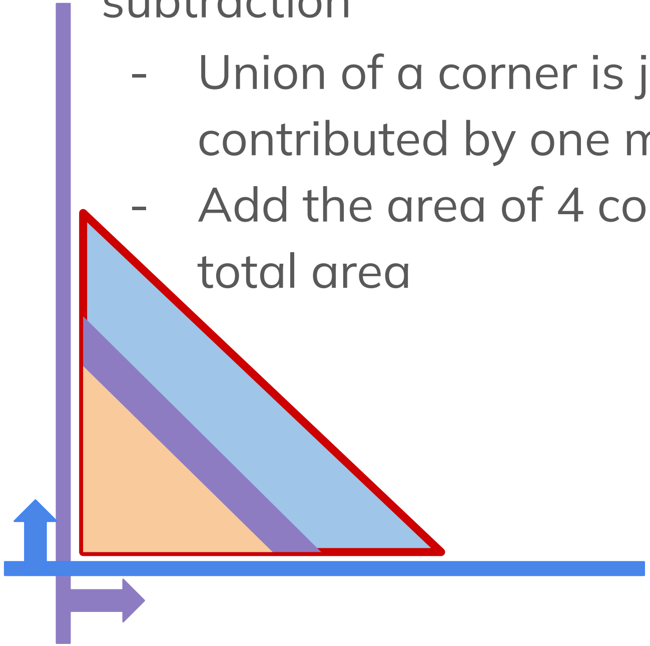
- Only select the mountains that contribute to the union
- Remove duplicate area

Subtask 5 (12%): No additional constraints



Issue: Corners are double counted in subtraction

- Union of a corner is just a triangle contributed by one mountain
- Add the area of 4 corners to the total area



Subtask 5 (12%): No additional constraints

Now we have to handle out of bound orthogonal squares

- Calculate area of rectangles - no. of cells out of bound on each side + no. of cells out of bound on each corner

- Total complexity: O(M logM log(max(H[i])))
- Expected Score: 100

Side note: the constant for this task may be very large if poorly implemented