

# Editorial - Christmas Tree

## Restating the problem precisely

We are given an  $N \times M$  matrix  $T$  of integer brightness values.

A **Christmas Tree** of height  $H > 1$  is defined by choosing a **star** cell  $(i, j)$  and taking a triangular shape:

- It occupies rows  $i, i + 1, \dots, i + H - 1$ .
- On row  $i + d$  (for  $d = 0, 1, \dots, H - 1$ ) it includes columns  $j - d, j - d + 1, \dots, j + d$ .

So the tree is the set of cells

$$(i + d, j + \Delta) \quad \text{for } d = 0..H - 1, |\Delta| \leq d,$$

and it must fit inside the matrix.

## Valid Christmas Trees

For each  $d = 0..H - 2$ , **every** cell on row  $i + d$  inside the tree must be strictly smaller than the **minimum** value among all tree cells on row  $i + d + 1$ .

Formally, if

- $S_d = \{T_{i+d, j-d}, T_{i+d, j-d+1}, \dots, T_{i+d, j+d}\},$
- $S_{d+1} = \{T_{i+d+1, j-(d+1)}, \dots, T_{i+d+1, j+(d+1)}\},$

then

$$\forall x \in S_d : \quad x < \min(S_{d+1}).$$

Observation: the rule for a valid Christmas tree is equivalent to "max < min"

The statement condition

$$\forall x \in S_d : \quad x < \min(S_{d+1})$$

is equivalent to

$$\max(S_d) \leq \min(S_{d+1}).$$

Why?

- If every element of  $S_d$  is  $< \min(S_{d+1})$ , then in particular the largest element of  $S_d$  is also  $< \min(S_{d+1})$ .
- Conversely, if  $\max(S_d) < \min(S_{d+1})$ , then every element of  $S_d$  is  $\leq \max(S_d)$  and hence also  $< \min(S_{d+1})$ .

This equivalence is the key that lets us check each "row boundary" using range maximum/minimum queries.

## Decomposing the tree into row boundaries

A tree of height  $H$  has boundaries between consecutive rows:

- between rows  $i$  and  $i + 1$
- between rows  $i + 1$  and  $i + 2$
- ...
- between rows  $i + H - 2$  and  $i + H - 1$

For a fixed star  $(i, j)$ , at the boundary between rows  $(r, r + 1)$  where  $r = i + d$ , the involved intervals are:

- on row  $r$  (depth  $d$ ): columns  $[j - d, j + d]$
- on row  $r + 1$  (depth  $d + 1$ ): columns  $[j - d - 1, j + d + 1]$

The boundary condition becomes

$$\max(T_{r, j-d..j+d}) < \min(T_{r+1, j-(d+1)..j+(d+1)}).$$

A tree is valid iff **every boundary condition holds** for all depths  $d = 0..H - 2$ .

So we will precompute, for each boundary and each center column, how deep that boundary remains valid.

## Precomputing boundary capacities $b[r][j]$

Fix a boundary between row  $r$  and row  $r + 1$ , and a center column  $j$ .

Define  $b[r][j]$  as the maximum depth  $d \geq 0$  such that

$$\max(T_{r, j-d..j+d}) < \min(T_{r+1, j-(d+1)..j+(d+1)})$$

and both intervals stay inside columns  $1..M$ .

If even  $d = 0$  fails, we set  $b[r][j] = -1$ .

## Why we can binary search on $d$

As  $d$  increases:

- the upper interval grows, so its maximum can only stay the same or increase,
- the lower interval grows, so its minimum can only stay the same or decrease.

Therefore the predicate

$$P(d) : \max(\text{upper interval at depth } d) < \min(\text{lower interval at depth } d + 1)$$

is monotone in  $d$ : once it becomes false, it stays false for all larger  $d$ .

So for each  $(r, j)$  we can binary search  $d$  in  $[0..d_{\max}]$ , where the lower interval needs radius  $d + 1$  to fit:

$$d_{\max} = \min(j - 2, M - j - 1).$$

## Range max/min in $O(1)$ with sparse table RMQ

To evaluate  $P(d)$  we need:

- a range maximum on row  $r$ ,
- a range minimum on row  $r + 1$ .

For each row separately build sparse tables:

- $\text{stMax}[k][i] = \max$  on  $[i, i + 2^k - 1]$ ,
- $\text{stMin}[k][i] = \min$  on  $[i, i + 2^k - 1]$ ,

and precompute  $\log_2$  values for lengths.

$$P(d) : \max(\text{upper interval at depth } d) < \min(\text{lower interval at depth } d + 1)$$

## Complexity of this phase

There are  $(N - 1) \cdot M$  pairs  $(r, j)$ .

For each we binary search over  $d$ , which takes  $O(\log M)$  steps, each step using  $O(1)$  RMQ.

So this phase is

$$O(NM \log M).$$

## From boundary capacities to maximum height per star

Now we want, for each star position  $(i, j)$ , the maximum height  $H_{\max}(i, j)$  of a valid tree.

Turning the boundary condition into a start-row condition

At boundary  $(r, r + 1)$ , a star at row  $i$  meets this boundary at depth

$$d = r - i.$$

The boundary is passable iff

$$r - i \leq b[r][j].$$

Rearrange:

$$i \geq r - b[r][j].$$

Define

$$L[r][j] = \begin{cases} r - b[r][j], & b[r][j] \geq 0, \\ r + 1, & b[r][j] = -1. \end{cases}$$

Then a star starting at row  $i$  can pass boundary  $r$  iff

$$i \geq L[r][j].$$

Fix a column  $j$  and solve all start rows  $i$  in  $O(N)$

Fix a column  $j$ . Consider the array  $L[1..N - 1]$  for that column.

For a given start row  $i$ , we want the farthest boundary index  $e \geq i$  such that all boundaries in  $[i..e]$  are passable:

$$\forall r \in [i..e]: \quad L[r] \leq i.$$

Equivalently,

$$\max_{r \in [i..e]} L[r] \leq i.$$

If this holds, then the tree can go down to row  $e + 1$ , so:

- number of crossed boundaries is  $e - i + 1$ ,
- height is boundaries + 1, i.e.

$$H = (e - i + 1) + 1 = e - i + 2.$$

We can compute all these maxima efficiently with a two-pointer window and a monotonic deque that maintains the sliding maximum of  $L$ :

- Maintain a window of boundaries  $[i..e]$ ,
- Keep a deque of indices with decreasing  $L$  values (front is the maximum),

- Increase  $e$  while the current maximum stays  $\leq i$ ,
- Move  $i$  forward and pop outdated indices.

This yields  $O(N)$  per fixed column  $j$ , so  $O(NM)$  for all columns.

## Horizontal fit limit

Even if the vertical conditions allow a large height, the triangle must fit horizontally.

At height  $H$ , the bottom row has radius  $H - 1$ , so we need

$$H - 1 \leq \min(j - 1, M - j),$$

which implies

$$H \leq 1 + \min(j - 1, M - j).$$

So we clamp the computed height by this limit.

## From maximum height to answers for every $k$

Let  $H_{\max}(i, j)$  be the maximum height possible at star  $(i, j)$ .

Important monotonic fact:

If a star supports height  $h$ , it also supports every smaller height  $k \leq h$  (just take the top  $k$  rows; all required boundary inequalities are a subset).

Therefore, the answer for "height exactly  $k$ " equals

$$\max\{T[i][j] \mid H_{\max}(i, j) \geq k\}.$$

We compute:

- `best[h]` = maximum  $T[i][j]$  among stars with  $H_{\max}(i, j) = h$ ,
- suffix maximum

$$\text{ans}[k] = \max(\text{best}[k + 1], \text{best}[k]),$$

- output `ans[k]` for  $k = 2..N$  (or  $-1$  if no value exists / it is impossible).

## Edge cases

If  $M < 3$ , no height  $\geq 2$  tree is possible. This is exactly subtask one for 3 points.