Towers

Yan Hao

22 March 2022

Towers

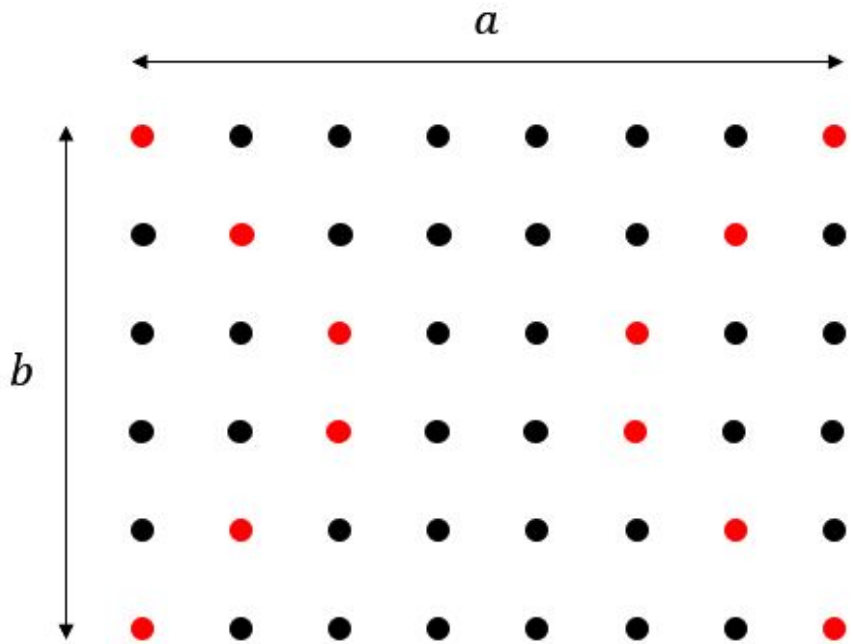
- We have n distinct points on a 2D plane
- Build some towers on these points satisfying the following properties:
  - Every row has at most 2 towers
  - Every column has at most 2 towers
  - Every point is either a tower, or lies on a horizontal or vertical line segment containing 2 towers
- It is always possible to do it, but how?

Subtask 1 & 2

- Subtask 1 ($N \leq 3$)
- If all 3 towers lie in the same row or same column, pick first and last
- Else, choose all 3
- Subtask 2 ($N \leq 16$)
- Guess and check: for every possible assignment of 1 and 0, determine if valid

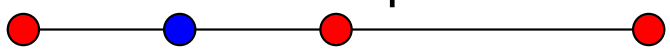
Subtask 3

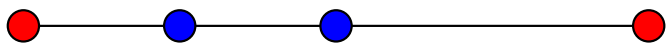
- We have a rectangle grid
- The 4 corners of the rectangle must be towers, since it is impossible to cover them
- Once we remove the outermost layer, we have a smaller rectangle grid
- Proceed your way inwards



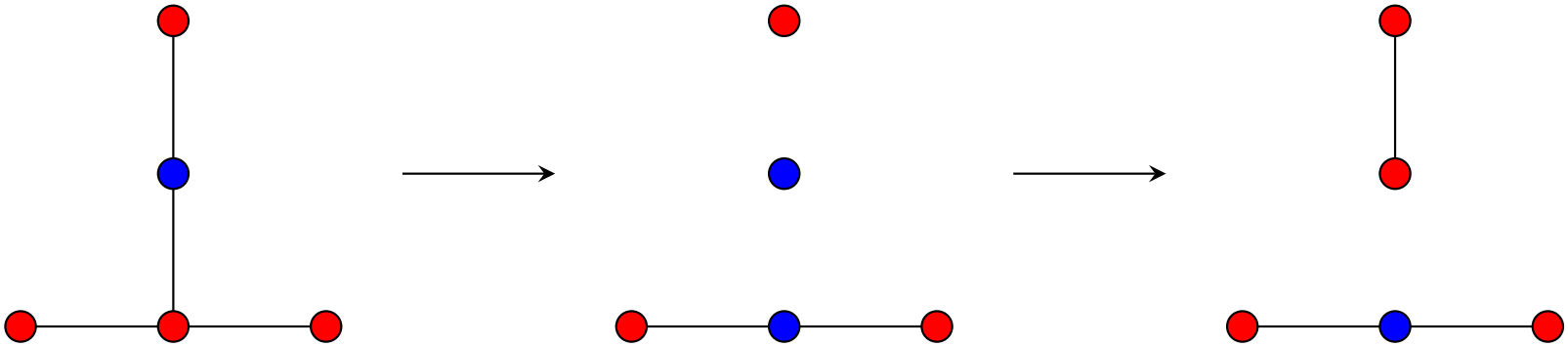
Subtask 4

- Every column has at most two points
- Lets introduce some notation
- A point is horizontally covered if there exists a tower to its left and a tower to its right
- Wishful thinking: can we make every point a tower?
- No, because some towers might be horizontally covered
- For every horizontally covered tower, we can remove the tower
- Will a covered point become uncovered? No!



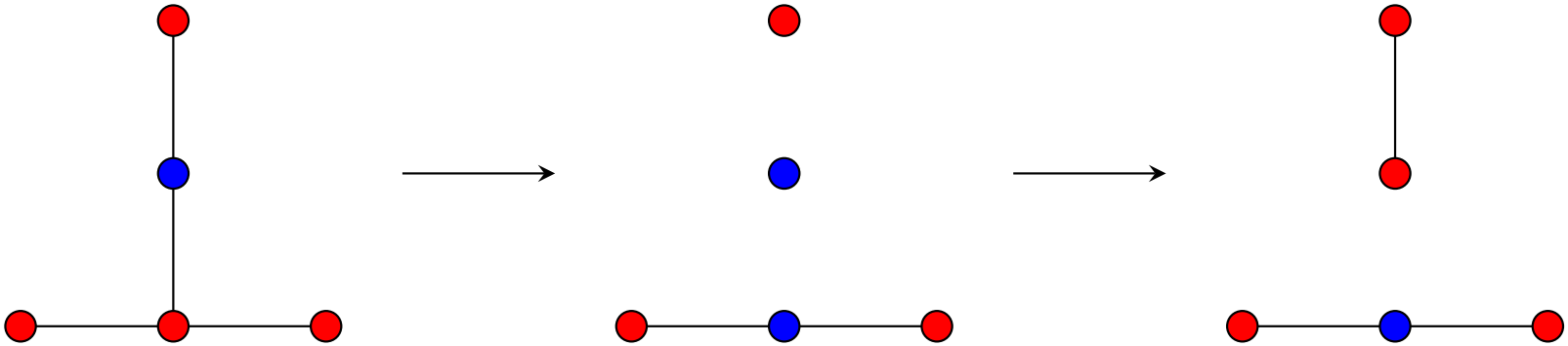


Subtask 5



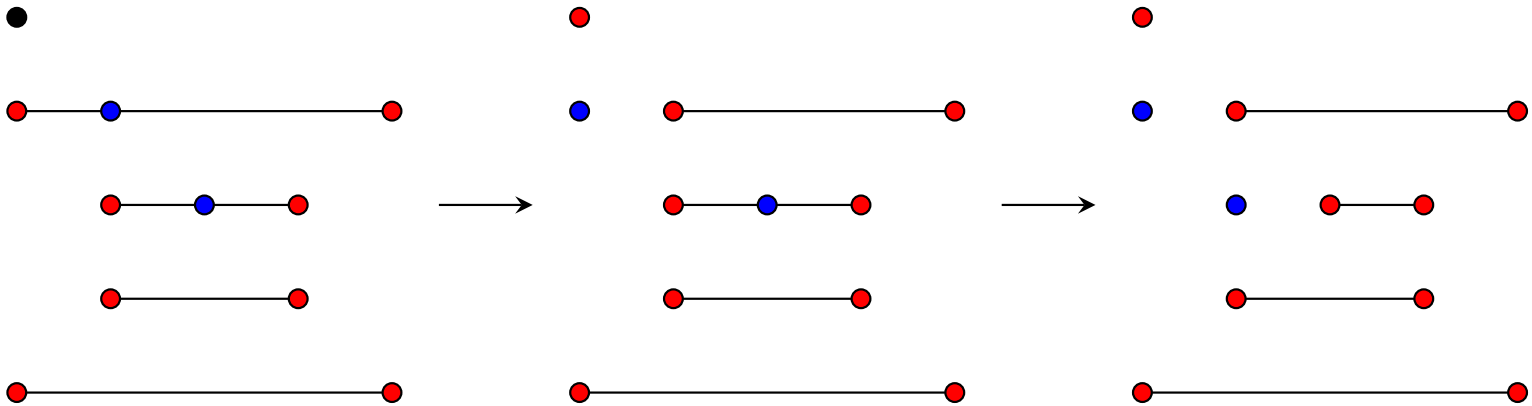
- Use idea from previous subtask
- If we remove a tower, can it cause some other point to be uncovered?
- Possibly, but we can simply convert the newly uncovered point into a tower
- So effectively, we have 'moved' a tower vertically
- Does this process terminate? Unclear.
- Idea: start with a configuration that does not have 3 towers in a row, but may have 3 towers in a column
- When executing this algorithm, towers can only 'move' left and right

Subtask 5



- Idea: start with a configuration that does not have 3 towers in a row, but may have 3 towers in a column
- When executing this algorithm, towers can only 'move' left and right
- We maintain the invariant that there is only ≤ 2 towers in a row
- For every row, consider the line segment joining the two towers
- The length of this line segment decreases, so this must terminate

Subtask 5



- Length of horizontal covering segments must always decrease, so this must terminate
- But how long does this algorithm take?
  - Let the 'length' be the number of points that it covers

Subtask 6 & 7

- We need to do this more efficiently
- One way is to keep a set for every row and column
- Unfortunately, for $n = 10^6$ the memory limits may not allow you to do so
- Use priority queue instead of set, less memory used
- Alternatively, store the leftmost and rightmost point for every row instead