Bombs - Solution

Encho Mishinev

Graph Interpretation

In this problem it is rather useful to model the boxes as a graph. Each box will correspond to a node and two boxes will be connected if they can be removed immediately by a single bomb, i.e. they have a clear line of sight horizontally or vertically. This graph is not static once we start placing bombs – two boxes that have several other boxes inbetween them may not have an edge at first, but get one once those intermediate boxes have been removed.

We will talk about two **types** of edges in this graph - horizontal and vertical. As the name suggests, they indicate whether the connected boxes are in the same row or column. If A and B are two boxes that we know have an edge in the graph, then by "the $A - B$ bomb" we will mean any bomb that removes both boxes. Such bomb always exists because we are told that the cells around a box are always empty.

Each node in this graph has at most 4 neighbours - one in each direction. Thus it can be built in $O((N + M)log(N + M))$ by sorting the positions of all boxes and rocks and scanning them in some direction.

Finding an optimal solution

At first glance it is hard to tell if a certain solution is optimal. However, consider the graph interpretation and split the graph into connected components. It is clear that a single bomb cannot remove more than two boxes. Furthermore, if two components are not connected in the initial graph, it is rather obvious they will not be connected at any later point. The implication is that the theoretical optimal we can hope for in a single connected component of size K is to use $\lceil \frac{K}{2} \rceil$ bombs. As we will see, essentially all solutions rely on this bound.

Subtask 1 [8 points]

$N, M \leq 10$

The intention is to bruteforce a solution. We can always try to place a bomb if it destroys at least two boxes, and once that's not possible the remaining destruction of singles is forced. Since we have no intuition about what pairs of boxes to remove, we just try all possible solutions.

Subtask 2 [10 points]

$N, M \leq 2000$, *There are at most two boxes in a single row or a single column*

Consider the graph interpretation. With the special restriction in this subtask each node has at most two neighbours - this implies that the graph consists entirely of disjoint cycles and paths.

Knowing the optimal bound, it is easy to see how to achieve it here. We simply follow along the path or cycle (for a cycle we can start at any point) and remove adjacent pairs of boxes. No new edges can be formed in this graph as we progress due to the restriction. Thus for any path or cycle of length K we use precisely $\lceil \frac{K}{2} \rceil$ bombs, which is optimal.

Subtask 3 [18 points]

$N, M \leq 2000$, *Only rows 2 and 4 contain non-empty cells*

There are many ideas that may work for this subtask, here we describe one such greedy strategy.

We can think of the rows as chunks of boxes separated by rocks. Let the leftmost rock in row 2 be at position R_a and the first rock in row 4 be at position R_b. Without loss of generality take $R_a \leq R_b$. Now let there be S_a boxes left of R_a in row 2, and S_b boxes left of R_b in row 4.

Our greedy strategy is as follows:

1. If S_a is even, we remove those boxes using $\frac{S_a}{2}$ bombs by removing consecutive pairs.
2. If S_a is odd:
   (a) If there is a vertical connection between any of the S_a boxes to any of the S_b boxes, use one bomb to remove such a pair, then remove the rest of the $S_a - 1$ boxes in the top row with $\frac{S_a-1}{2}$ bombs.
   (b) If there is no vertical connection, remove all S_a boxes using $\frac{S_a-1}{2} + 1$ bombs by removing consecutive pairs and using one bomb for the final box left.

After one application of the rules above we have eliminated "a chunk of boxes followed by a rock" and we can repeat the logic considering the next rock.

Intuitively speaking, the greedy strategy is correct because since $R_a \leq R_b$, all boxes in the top row left of R_a can only be affected by some vertical connection with the bottom row. Since parity is all that matters for the optimal bound, we make use of such vertical connection only if it is necessary. One can also formally prove this greedy by considering the graph interpretation.

There are likely many other ideas that are correct for the structure in this subtask.

Subtask 4 [26 points]

$N, M \leq 2000$

Consider again the graph interpretation and the optimal lower bound. In order to solve this subtask we must suppose that **it is always possible to destroy a connected component of size** K **in** $\lceil \frac{K}{2} \rceil$ **bombs**. This turns out to be true. The contestant may either intuitively guess this, prove it non-constructively, or prove it by an inefficient construction, but in any case it immediately gives a polynomial solution. The statement implies that we can easily compute the number of bombs needed as it is always equal to $\frac{N+X}{2}$ where X is the number of components in the described graph that are of odd size. One can then find valid pairs by simply repeating:

1. Count the number of odd components in the graph
2. Find a pair of boxes that can be destroyed without increasing the number of odd components
3. Remove that pair, update the graph, and repeat 2

Once there is no such pair in step 2 then the graph must have no edges at all, and we can remove the remaining single boxes with one bomb each.

There are $O(N)$ pairs to consider in step 2 and removing a pair affects only a constant number of edges and nodes, so one can predict the number of odd components in $O(1)$. As such the algorithm can be implemented in $O(N^2)$ without any deeper understanding.

In theory one can try to improve this solution to achieve full score by being smarter about finding those pairs in step 2, but it is quite likely that by doing so one ends up inventing the intended full score constructive algorithm.

Subtask 5 [38 points]

The full solution is based on a linear constructive algorithm that achieves the optimal bound.

Without loss of generality we will only describe the solution to a single connected component, since the solutions for multiple ones are fully independent. We start by finding an arbitrary spanning tree, e.g. by running DFS.

We will iteratively remove pairs of nodes from our component. If the component is of size 1 or 2, then the solution is trivial using a single bomb. Otherwise, let us have a component of size $S \geq 3$. There exists a node A such that it is not a leaf, but all of its children are leaves. Let P be the parent of A and let's consider A's children. Since the degree of any node is at most 4, there may be at most 3 children. The cases are:

1. *A has a single child B*
   In this case our **first** bomb can be $A - B$. Now we can remove both A and B from the tree, and since A has no other children, the remaining tree is still connected and a valid spanning tree of size $S - 2$. We place the $A - B$ bomb and then find the optimal bombs for the remaining tree.

2. *A has some pair of children B and C, such that both edges $A - B$ and $A - C$ are of the same type (vertical or horizontal)*
   This implies that if we were to remove A, the remaining graph would have the edge $B - C$. In this case we can just take our **last** bomb to be $B - C$. We then remove both B and C and solve the remaining tree of size $S - 2$. It is clearly still connected because B and C were leaves. Once we have the optimal bombs for the remaining graph, we **append** the $B - C$ bomb.

3. *A has exactly two children B and C, such that $A - B$ and $A - C$ are edges of different type*
   Since there are only two types of edges, it follows that either $A - B$ or $A - C$ is of the same type as $P - A$. Without loss of generality let $P - A$ and $A - B$ be of the same type. We can take the $A - C$ bomb as our **first** and remove them, adding a new edge between P and B. Thus, the remaining tree is still connected and is of size $S - 2$. We can then just solve for this tree to find the remaining bombs.

The above three cases cover all possibilities for A's children, and thus iteratively applying this yields a full solution in which each bomb removes 2 boxes, except when the component consists of a single node. This is clearly optimal.

It is important to note that since in case 2 we choose our **last** bomb, it may seem that ignoring B and C while constructing the remaining solution is not sound – after all, they will not have been removed during our solution. However, for them to ever be an issue the situation must be such that we have an edge in the tree that in reality is being blocked by B or C. However, the only case that ever introduces a new edge is case 3, and it preserves the invariant of no edge being blocked. Since case 2 itself removes only leaves, none of the remaining edges are blocked, either.

The solution can be implemented in $O(N)$ with a single DFS once the graph is constructed. The total complexity is technically $O((N+M)log(N+M))$ because constructing the graph initially requires sorting all boxes and rocks.