Text editor

Terminology

There are three main types of moves:

- Key presses that keep the cursor within a line or columns, which we'll call grid movements.
- Vertical key presses that move the cursor past the end of a line, causing it to snap to the end of the line, which we'll call fall movements.
- Key presses that move the cursor between the start of one line and the end of another, which we'll call jump movements.

Slow solutions

One of the easiest possible solutions is to use DFS. This is obviously correct, but extremely slow when the lines are too long.

You may notice that the vast majority of columns aren't very interesting. In fact, it never makes sense to change direction unless you're in a columns that's either:

- the first column
- the last column of a line
- the initial or target column

This makes it possible to construct a compressed graph with $\mathcal{O}(N^2)$ nodes and vertices and use Dijkstra's algorithm to find the shortest path in $\mathcal{O}(N^2 \log N)$.

Full solution

When trying to solve some interesting cases by hand, you may notice that it's often very useful to move to the first column, since you can quickly reach the end of any line from there, which allows you quick access to many columns. Therefore, we can split the path into two cases: those that visit the first column and those that don't.

Paths that don't visit the first column

If a path doesn't visit the first column, it cannot make use of jump movements. Therefore, we only have to consider grid and fall movements. You may notice that it never makes sense to make

horizontal grid movements and then use a fall movement afterwards. Similarly, its never needed to use horizontal grid movements before reaching the target line.

Therefore, we only need to consider paths that consist the following steps:

1. Move vertically to some line i, potentially using fall movements along the way.
2. Move vertically to line e_l.
3. Move horizontally to column e_c.

It's possible that some or all of those steps will involve 0 key presses.

If you compute the prefix and suffix minimums of the line lengths, you can enumerate all possible candidates for line i and calculate the length of the resulting path for each in linear time.

Paths that do visit the first column

You may notice that it's never necessary to visit the first column multiple times. This follows from the fact that getting from line i to line j requires a minimum of $|i - j|$ key presses.

Therefore, we only need to consider paths that consist the following steps:

1. Somehow move to the first column.
2. Move vertically within the first column.
3. Somehow move to the target position.

There are two main ways to reach the first column:

1. Move vertically to some line i, then use left key presses to move to the start of line i.
2. Move vertically to some line i, then use right key presses (the last of which is a jump) to move to the start of line $i + 1$.

You can also use a fall movement to reach the start of a line of length 0, but that's really just a special case of case 1.

We can again use prefix and suffix minimums to figure out what column the cursor would end up in after using vertical movements to reach a given column i. Using that we can easily compute the length of the shortest path to reach every position in column 1.

There are two ways to reach the target position from the first column:

1. Use right key presses to reach the target position from $(e_l, 1)$.
2. Jump to the end of some line j, move vertically to line e_l and use horizontal grid movements to get to the correct solution.

Again, we can enumerate all candidate lines j and compute the length of the resulting path for each in linear time.