--- Day 22: Sand Slabs ---

Enough sand has fallen; it can finally filter water for Snow Island.

Well, **almost**.

The sand has been falling as large compacted **bricks** of sand, piling up to form an impressive stack here near the edge of Island Island. In order to make use of the sand to filter water, some of the bricks will need to be broken apart - nay, **disintegrated** - back into freely flowing sand.

The stack is tall enough that you'll have to be careful about choosing which bricks to disintegrate; if you disintegrate the wrong brick, large portions of the stack could topple, which sounds pretty dangerous.

The Elves responsible for water filtering operations took a **snapshot of the bricks while they were still falling** (your puzzle input) which should let you work out which bricks are safe to disintegrate. For example:

```
1,0,1~1,2,1
0,0,2~2,0,2
0,2,3~2,2,3
0,0,4~0,2,4
2,0,5~2,2,5
0,1,6~2,1,6
1,1,8~1,1,9
```

Each line of text in the snapshot represents the position of a single brick at the time the snapshot was taken. The position is given as two `x,y,z` coordinates - one for each end of the brick - separated by a tilde (`~`). Each brick is made up of a single straight line of cubes, and the Elves were even careful to choose a time for the snapshot that had all of the free-falling bricks at **integer positions above the ground**, so the whole snapshot is aligned to a three-dimensional cube grid.

A line like `2,2,2~2,2,2` means that both ends of the brick are at the same coordinate - in other words, that the brick is a single cube.

Lines like `0,0,10~1,0,10` or `0,0,10~0,1,10` both represent bricks that are **two cubes** in volume, both oriented horizontally. The first brick extends in the `x` direction, while the second brick extends in the `y` direction.

A line like `0,0,1~0,0,10` represents a **ten-cube brick** which is oriented **vertically**. One end of the brick is the cube located at `0,0,1`, while the other end of the brick is located directly above it at `0,0,10`.

The ground is at `z=0` and is perfectly flat; the lowest `z` value a brick can have is therefore `1`. So, `5,5,1~5,6,1` and `0,2,1~0,2,5` are both resting on the ground, but `3,3,2~3,3,3` was above the ground at the time of the snapshot.

Because the snapshot was taken while the bricks were still falling, some bricks will **still be in the air**; you'll need to start by figuring out where they will end up. Bricks are magically stabilized, so they **never rotate**, even in weird situations like where a long horizontal brick is only supported on one end. Two bricks cannot occupy the same position, so a falling brick will come to rest upon the first other brick it encounters.

Here is the same example again, this time with each brick given a letter so it can be marked in diagrams:

```
1,0,1~1,2,1   <- A
0,0,2~2,0,2   <- B
0,2,3~2,2,3   <- C
0,0,4~0,2,4   <- D
2,0,5~2,2,5   <- E
0,1,6~2,1,6   <- F
1,1,8~1,1,9   <- G
```

At the time of the snapshot, from the side so the `x` axis goes left to right, these bricks are arranged like this:

```
 x
012
.G. 9
.G. 8
... 7
FFF 6
..E 5 z
D.. 4
CCC 3
BBB 2
.A. 1
--- 0
```

Rotating the perspective 90 degrees so the `y` axis now goes left to right, the same bricks are arranged like this:

```
 y
012
.G. 9
.G. 8
... 7
.F. 6
EEE 5 z
DDD 4
..C 3
B.. 2
AAA 1
--- 0
```

Once all of the bricks fall downward as far as they can go, the stack looks like this, where `?` means bricks are hidden behind other bricks at that location:

```
 x
012
.G. 6
.G. 5
FFF 4
D.E 3 z
??? 2
.A. 1
--- 0
```

Again from the side:

```
 y
012
.G. 6
.G. 5
.F. 4
??? 3 z
B.C 2
AAA 1
--- 0
```

Now that all of the bricks have settled, it becomes easier to tell which bricks are supporting which other bricks:

- Brick `A` is the only brick supporting bricks `B` and `C`.
- Brick `B` is one of two bricks supporting brick `D` and brick `E`.
- Brick `C` is the other brick supporting brick `D` and brick `E`.
- Brick `D` supports brick `F`.
- Brick `E` also supports brick `F`.
- Brick `F` supports brick `G`.
- Brick `G` isn't supporting any bricks.

Your first task is to figure out **which bricks are safe to disintegrate**. A brick can be safely disintegrated if, after removing it, **no other bricks** would fall further directly downward. Don't actually disintegrate any bricks - just determine what would happen if, for each brick, only that brick were disintegrated. Bricks can be disintegrated even if they're completely surrounded by other bricks; you can squeeze between bricks if you need to.

In this example, the bricks can be disintegrated as follows:

- Brick `A` cannot be disintegrated safely; if it were disintegrated, bricks `B` and `C` would both fall.
- Brick `B` **can** be disintegrated; the bricks above it (`D` and `E`) would still be supported by brick `C`.
- Brick `C` **can** be disintegrated; the bricks above it (`D` and `E`) would still be supported by brick `B`.
- Brick `D` **can** be disintegrated; the brick above it (`F`) would still be supported by brick `E`.
- Brick `E` **can** be disintegrated; the brick above it (`F`) would still be supported by brick `D`.
- Brick `F` cannot be disintegrated; the brick above it (`G`) would fall.
- Brick `G` **can** be disintegrated; it does not support any other bricks.

So, in this example, **5** bricks can be safely disintegrated.

Figure how the blocks will settle based on the snapshot. Once they've settled, consider disintegrating a single brick; **how many bricks could be safely chosen as the one to get disintegrated?**

To begin, get your puzzle input.

Answer: [Submit]

You can also [Share] this puzzle.