There are many important principles that can improve a change someone will achieve a considered successful life. But one of most important I think is stable and the ability to keep track of our own project. But to be honest, I am kind of very hot-head and my natural childlike personality keep drag me to new ideas. Let ‘s see can I bypass myself and finish these aoc challenges or not

Part 1

```
In[1]:= input = "..." ;
```

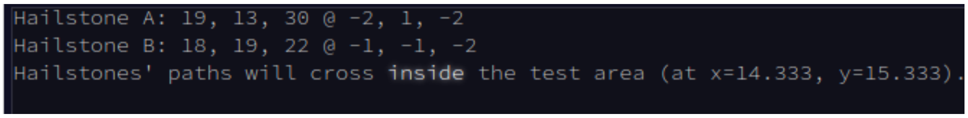
Well, uhm, the pattern, the first thing I feel when I read this challenge is it directly refer to mathematic, it just like, plot 2 linear vectors on the 2D plane and check if the crossed or not. Well check the full content here https://adventofcode.com/2023/day/24 .

We test with the small input first to make clear my explain

```
In[2]:= smallInput = "..." ;

In[3]:= smallInput // StringTrim // StringSplit[#, "\n"] & // TableForm
Out[3]//TableForm=
19, 13, 30 @ -2,  1, -2
18, 19, 22 @ -1, -1, -2
20, 25, 34 @ -2, -2, -4
12, 31, 28 @ -1, -2, -1
20, 19, 15 @  1, -5, -3
```

We will decode how to find the cross point {14.333,15.333} between line 1 and 2 of this example



```
Hailstone A: 19, 13, 30 @ -2, 1, -2
Hailstone B: 18, 19, 22 @ -1, -1, -2
Hailstones' paths will cross inside the test area (at x=14.333, y=15.333).
```

x and y in each line will transform base on the system of linear equations

$x = x_{\text{init}} + x_v$*time
$y = y_{\text{init}} + y_v$*time

It not hard to see init position, velocity of x and y, I mean in the first hailstone A: {19,13} and {-2,-1} is constants, do some mathematic by divide two linear equation will remove time changing variable and this system will become relation of x and y

$$\frac{x - x_i}{y - y_i} = \frac{x_v}{y_v}$$

I will anotation init by i and velocity by v for short

$$(x - x_i)\ y_v\ = x_v\ (y - y_i)$$

$\Rightarrow x\, y_v - x_i\ \ y_v = x_v y - x_v\, y_i$

$\Rightarrow x\, y_v - y\, x_v = x_i\, y_v - y_i\, x_v$

Okie, so the equation of line 1 is x + 2y = 19*(1) - 13*(-2) = 45 and line 2 is -x +y = -18 +19 = 1

Let solve the equation , wolfram lang have built in function

```
In[4]:= Solve[x + 2 y == 45 && -x + y == 1 , {x, y}] // N
Out[4]= {{x → 14.3333, y → 15.3333}}
```

```
In[5]:= N[{{x → 43/3 , y → 46/3}}]
Out[5]= {{x → 14.3333, y → 15.3333}}
```

Perfect, exactly like the example

```
In[51]:= findCrossPoint[stone1_, stone2_] := Module[{},
           {xi1, yi1, zi1, xv1, yv1, zv1} = ToExpression /@ (StringSplit[stone1, {"@", ","}]);
           {xi2, yi2, zi2, xv2, yv2, zv2} = ToExpression /@ StringSplit[stone2, {"@", ","}];
           EchoLabel["stone1"]@ {xi1, yi1, zi1, xv1, yv1, zv1};
           EchoLabel["stone2"]@ {xi2, yi2, zi2, xv2, yv2, zv2};
           res = Solve[x * yv1 - y * xv1 == xi1 * yv1 - yi1 * xv1 &&
                 x * yv2 - y * xv2 == xi2 * yv2 - yi2 * xv2, {x, y}] // N;
           EchoLabel[res]@res
          ]
```

```
In[7]:= smallInputSplit = smallInput // StringTrim // StringSplit[#, "\n"] &
Out[7]= {19, 13, 30 @ -2,  1, -2, 18, 19, 22 @ -1, -1, -2,
         20, 25, 34 @ -2, -2, -4, 12, 31, 28 @ -1, -2, -1, 20, 19, 15 @  1, -5, -3}
```

```
In[8]:= findCrossPoint[smallInputSplit[[1]], smallInputSplit[[2]]]
      » 19, 13, 30 @ -2,  1, -2 {19, 13, 30, -2, 1, -2}
      » 18, 19, 22 @ -1, -1, -2 {18, 19, 22, -1, -1, -2}
Out[8]= {{x → 14.3333, y → 15.3333}}
```

Path 2 and 3 never crossed, let check

```
In[9]:= {{x → 14.333333333333334`, y → 15.333333333333334`}}[[1]] // Values
Out[9]= {14.3333, 15.3333}
```

```
In[10]:= findCrossPoint[smallInputSplit[[2]], smallInputSplit[[3]]]
      » 18, 19, 22 @ -1, -1, -2 {18, 19, 22, -1, -1, -2}
      » 20, 25, 34 @ -2, -2, -4 {20, 25, 34, -2, -2, -4}
Out[10]=
        {}
```

Good .

We will never have {{}} as a result. It mean the full set solution, because our two init point and two velocity always different. Let check the real input with 300 lines

```
In[11]:= inputSplit = input // StringTrim // StringSplit[#, "\n"] &;
```

```
In[12]:= Counts[{#[[1]], #[[2]]} & /@
           (ToExpression /@ StringSplit[#, {"@", ","}] & /@ inputSplit)] // Values // Counts
Out[12]=
         <|1 → 300|>
```

```
In[13]:= Counts[{#[[4]], #[[5]]} & /@
           (ToExpression /@ StringSplit[#, {"@", ","}] & /@ inputSplit)] // Values // Counts
Out[13]=
         <|1 → 300|>
```

Okie, now I need to apply findCrossPoint to all pair in the input. Oh , in mathematic ways, we don't loop them, instead that we generate all the possible pair first and apply the function later on each pair. The pair must be unordered, that mean if we have {a,b} we don't need {b,a}

```
In[14]:= Subsets[inputSplit, {2}]
Out[14]=
         {{251454256616722, 382438634889004, 18645302082228 @ 43, -207, 371,
            289124150762025, 364325878532733, 278169080781801 @ -73, -158, -13},
          {251454256616722, 382438634889004, 18645302082228 @ 43, -207, 371,
            268852221227649, 10710819924145, 258969710792682 @ 41, 192, 62}, ··· 44 846 ··· ,
          {279172409384999, 332554952875949, 532315024821330 @ -19, -122, -394,
            245968014205034, 384015060139844, 157918607855134 @ -79, -167, 300},
          {319402351711330, 263034821469280, 329511641002098 @ -167, 75, -142,
            245968014205034, 384015060139844, 157918607855134 @ -79, -167, 300}}

         Size in memory: 11.3 MB | + Show more | Show all | Iconize ▾ | Store full expression in notebook
```

```
In[15]:= crossPoints = findCrossPoint @@@ Subsets[inputSplit, {2}] // QuietEcho
Out[15]=
         {{{x → 2.65733×10^14, y → 3.13699×10^14}}, {{x → 2.99175×10^14, y → 1.52712×10^14}},
          {{x → 2.78127×10^14, y → 2.54039×10^14}}, {{x → 2.60744×10^14, y → 3.3772×10^14}},
          {{x → 2.74191×10^14, y → 2.72983×10^14}}, {{x → 2.80388×10^14, y → 2.43155×10^14}},
           ··· 44 838 ··· , {{x → 2.67643×10^14, y → 2.58525×10^14}}, {{x → 2.68483×10^14, y → 2.85903×10^14}},
          {{x → 2.73262×10^14, y → 4.41713×10^14}}, {{x → 2.71683×10^14, y → 2.84466×10^14}},
          {{x → 3.07417×10^14, y → 5.13913×10^14}}, {{x → 2.11633×10^14, y → 3.11434×10^14}}}

         Size in memory: 11.5 MB | + Show more | Show all | Iconize ▾ | Store full expression in notebook
```

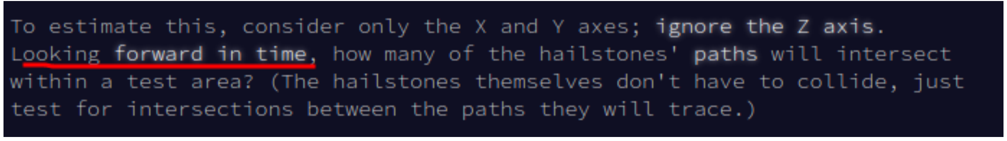
```
In[16]:= Module[{
           val = {x, y} /. #[[1]]
         },

         2 * 10^14 ≤ val[[1]] ≤ 4 * 10^14 && 2 * 10^14 ≤ val[[2]] ≤ 4 * 10^14
        ] & /@ Select[crossPoints, Length[#] ≠ 0 &] // Counts

Out[16]=
<|True → 21817, False → 23026|>
```

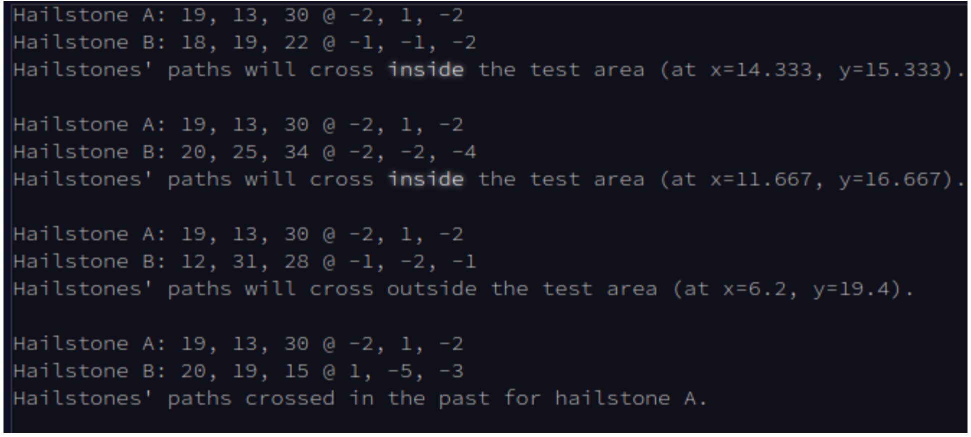
Oops, oh my gosh, what wrong, something I was missing here.

I just check with small input in scratchpad, and what I see here



To estimate this, consider only the X and Y axes; **ignore the Z axis**. Looking **forward in time**, how many of the hailstones' **paths** will intersect within a test area? (The hailstones themselves don't have to collide, just test for intersections between the paths they will trace.)

The author give the most confuse description I have ever seen. With example



```
Hailstone A: 19, 13, 30 @ -2, 1, -2
Hailstone B: 18, 19, 22 @ -1, -1, -2
Hailstones' paths will cross inside the test area (at x=14.333, y=15.333).

Hailstone A: 19, 13, 30 @ -2, 1, -2
Hailstone B: 20, 25, 34 @ -2, -2, -4
Hailstones' paths will cross inside the test area (at x=11.667, y=16.667).

Hailstone A: 19, 13, 30 @ -2, 1, -2
Hailstone B: 12, 31, 28 @ -1, -2, -1
Hailstones' paths will cross outside the test area (at x=6.2, y=19.4).

Hailstone A: 19, 13, 30 @ -2, 1, -2
Hailstone B: 20, 19, 15 @ 1, -5, -3
Hailstones' paths crossed in the past for hailstone A.
```

I have no idea what is meaning “crossed in the past of of”. Extremely confused, the problems is not yet hard, it took time because he make things so wordy, I try to guess here. The case of {19,13...@ -2,1..} and {20,19..@1,-5..}. If you check the result of crossed position, you will see it is {{x→21.4444,y→11.7778}}.The problem stay at x->21.44 at stone A, because velocity of stone A is negative, why the cross point position is somewhat higher . So he is meaning, the point of cross must suitable with the velocity. The cross point here somewhat stay in the opposite side of movement. Oh yeah, understood, let add some condition, but we must creative a bit, there is no way I do thing like check signs of each velocity and init points etc..... Uhm, let see, the result of (des - src)/veloc always have positive sign

```
In[91]:= checkIfAheadOfTime[{x1_, y1_, v11_, v12_},
           {x2_, y2_, v21_, v22_}, {xc_, yc_}] := Module[{},
           EchoLabel["check ahead of time"]@{{x1, y1, v11, v12}, {x2, y2, v21, v22}, {xc, yc}};
           If[(xc - x1)/v11 > 0 ∧ (xc - x2)/v21 > 0 ∧ (yc - y1)/v12 > 0 ∧ (yc - y2)/v22 > 0,
            True,
            False
           ]
          ]
```

```
In[89]:= findCrossPoint[stone1_, stone2_] := Module[{},
           {xi1, yi1, zi1, xv1, yv1, zv1} = ToExpression /@ (StringSplit[stone1, {"@", ","}]);
           {xi2, yi2, zi2, xv2, yv2, zv2} = ToExpression /@ StringSplit[stone2, {"@", ","}];
           EchoLabel["stone1"]@ {xi1, yi1, zi1, xv1, yv1, zv1};
           EchoLabel["stone2"]@ {xi2, yi2, zi2, xv2, yv2, zv2};
           res = Solve[x * yv1 - y * xv1 == xi1 * yv1 - yi1 * xv1 &&
               x * yv2 - y * xv2 == xi2 * yv2 - yi2 * xv2, {x, y}] // N;
           EchoLabel["res"]@res;

            ⎧ {}                                                              Length[res] =
            ⎨ ⎧ res  checkIfAheadOfTime[{xi1, yi1, xv1, yv1},
            ⎩ ⎨        {xi2, yi2, xv2, yv2}, res⟦1⟧ // Values]               True
              ⎩ {}   True

          ]
```

```
In[97]:= crossPoints = findCrossPoint @@@ Subsets[inputSplit, {2}] // QuietEcho
Out[97]=
 {{{x → 2.65733×10^14, y → 3.13699×10^14}}, {{x → 2.99175×10^14, y → 1.52712×10^14}},
  {{x → 2.78127×10^14, y → 2.54039×10^14}}, {{x → 2.60744×10^14, y → 3.3772×10^14}},
  {{x → 2.74191×10^14, y → 2.72983×10^14}}, {{x → 2.80388×10^14, y → 2.43155×10^14}}, {}, {},
  … 44834 … , {{x → 1.66197×10^14, y → 3.3184×10^14}}, {{x → 2.03595×10^14, y → 2.94442×10^14}},
  {{x → 2.67643×10^14, y → 2.58525×10^14}}, {{x → 2.68483×10^14, y → 2.85903×10^14}}, {},
  {{x → 2.71683×10^14, y → 2.84466×10^14}}, {}, {{x → 2.11633×10^14, y → 3.11434×10^14}}}

 Size in memory: 7.4 MB | + Show more | Show all | Iconize ▾ | Store full expression in notebook
```

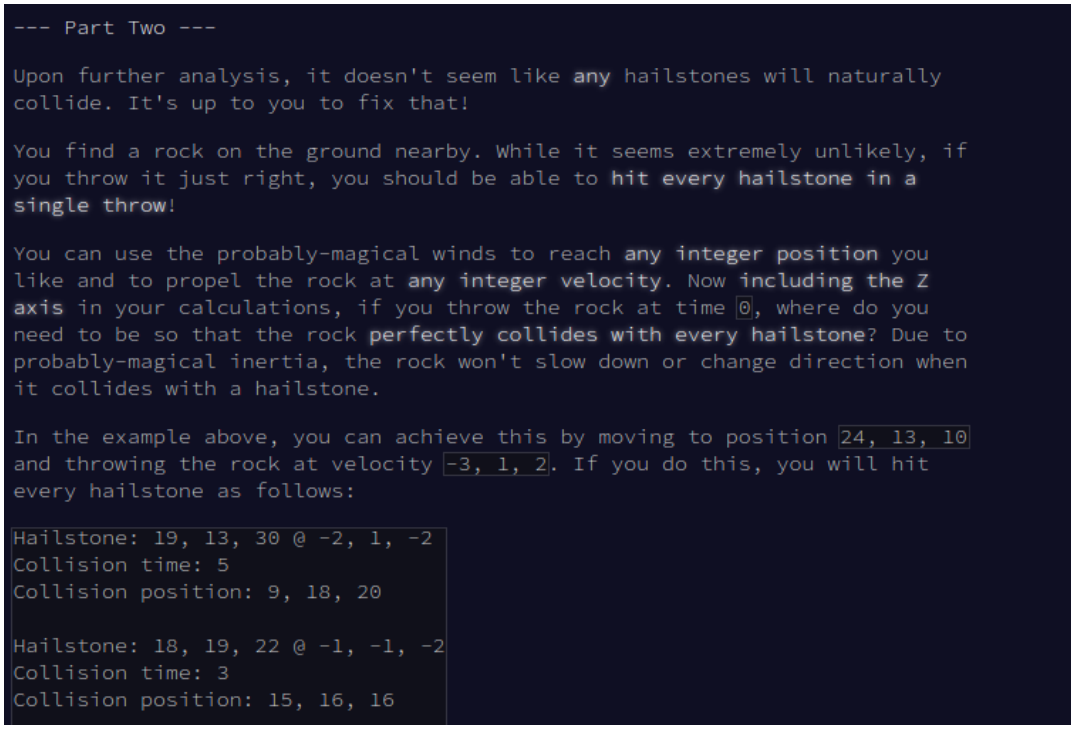
In[98]:=
```
Module[{
    val = {x, y} /. #[[1]]
   },

  2 * 10^14 ≤ val[[1]] ≤ 4 * 10^14 && 2 * 10^14 ≤ val[[2]] ≤ 4 * 10^14
 ] & /@ Select[crossPoints, Length[#] ≠ 0 &] // Counts
```

Out[98]=
```
<|True → 15 889, False → 9324|>
```

Oh my gosh, finally!

Part 2



```
--- Part Two ---
```

Upon further analysis, it doesn't seem like **any** hailstones will naturally collide. It's up to you to fix that!

You find a rock on the ground nearby. While it seems extremely unlikely, if you throw it just right, you should be able to **hit every hailstone in a single throw**!

You can use the probably-magical winds to reach **any integer position** you like and to propel the rock at **any integer velocity**. Now **including the Z axis** in your calculations, if you throw the rock at time `0`, where do you need to be so that the rock **perfectly collides with every hailstone**? Due to probably-magical inertia, the rock won't slow down or change direction when it collides with a hailstone.

In the example above, you can achieve this by moving to position `24, 13, 10` and throwing the rock at velocity `-3, 1, 2`. If you do this, you will hit every hailstone as follows:

```
Hailstone: 19, 13, 30 @ -2, 1, -2
Collision time: 5
Collision position: 9, 18, 20

Hailstone: 18, 19, 22 @ -1, -1, -2
Collision time: 3
Collision position: 15, 16, 16
```

Oops, i am stuck here

Scratchpad

2 In

In[∘]:=
```
Quit[]
```

```
In[18]:= Names["Global`*"]

Out[18]=
{autoCompletion, button, button$, c, cells, cells$, crossPoints, e, file, files,
 findCrossPoint, height, i, i1, i2, input, inputSplit, lines, lines$, listOfLists,
 listOfLists$, listPickers, listPickers$, next, notebook, notebookName,
 notebooks, notebook$, patt, pos, position, position$, pos$, section,
 sectionNames, sectionNames$, SectionPicker, section$, ShowNotebook, smallInput,
 smallInputSplit, stone1, stone2, str, str$, style, suck, todo, trash, v, v1,
 v2, val, VerminAddCompletion, VerminAutoCompletion, VerminConfigFileOpen,
 VerminGrep, VerminMakeNotebookIndex, VerminOpenTodo, width, x, y, z}

In[19]:= x
Out[19]=
x

In[77]:= findCrossPoint[smallInputSplit⟦2⟧, smallInputSplit⟦3⟧]
   » stone1 {18, 19, 22, -1, -1, -2}
   » stone2 {20, 25, 34, -2, -2, -4}
Out[77]=
{}

In[86]:= {{x → 21.444444444444443`, y → 11.777777777777779`}}⟦1⟧ // Values
Out[86]=
{21.4444, 11.7778}

In[93]:= a = findCrossPoint @@@ Subsets[smallInputSplit, {2}]
   » stone1 {19, 13, 30, -2, 1, -2}
   » stone2 {18, 19, 22, -1, -1, -2}
   » res {{x → 14.3333, y → 15.3333}}
   » check ahead of time {{19, 13, -2, 1}, {18, 19, -1, -1}, {14.3333, 15.3333}}
   » stone1 {19, 13, 30, -2, 1, -2}
   » stone2 {20, 25, 34, -2, -2, -4}
   » res {{x → 11.6667, y → 16.6667}}
   » check ahead of time {{19, 13, -2, 1}, {20, 25, -2, -2}, {11.6667, 16.6667}}
   » stone1 {19, 13, 30, -2, 1, -2}
   » stone2 {12, 31, 28, -1, -2, -1}
   » res {{x → 6.2, y → 19.4}}
   » check ahead of time {{19, 13, -2, 1}, {12, 31, -1, -2}, {6.2, 19.4}}
   » stone1 {19, 13, 30, -2, 1, -2}
   » stone2 {20, 19, 15, 1, -5, -3}
   » res {{x → 21.4444, y → 11.7778}}
   » check ahead of time {{19, 13, -2, 1}, {20, 19, 1, -5}, {21.4444, 11.7778}}
   » stone1 {18, 19, 22, -1, -1, -2}
   » stone2 {20, 25, 34, -2, -2, -4}
   » res {}
   » stone1 {18, 19, 22, -1, -1, -2}
   » stone2 {12, 31, 28, -1, -2, -1}
   » res {{x → -6., y → -5.}}
   » check ahead of time {{18, 19, -1, -1}, {12, 31, -1, -2}, {-6., -5.}}
   » stone1 {18, 19, 22, -1, -1, -2}
   » stone2 {20, 19, 15, 1, -5, -3}
   » res {{x → 19.6667, y → 20.6667}}
   » check ahead of time {{18, 19, -1, -1}, {20, 19, 1, -5}, {19.6667, 20.6667}}
   » stone1 {20, 25, 34, -2, -2, -4}
   » stone2 {12, 31, 28, -1, -2, -1}
   » res {{x → -2., y → 3.}}
   » check ahead of time {{20, 25, -2, -2}, {12, 31, -1, -2}, {-2., 3.}}
   » stone1 {20, 25, 34, -2, -2, -4}
   » stone2 {20, 19, 15, 1, -5, -3}
   » res {{x → 19., y → 24.}}
   » check ahead of time {{20, 25, -2, -2}, {20, 19, 1, -5}, {19., 24.}}
   » stone1 {12, 31, 28, -1, -2, -1}
   » stone2 {20, 19, 15, 1, -5, -3}
   » res {{x → 16., y → 39.}}
   » check ahead of time {{12, 31, -1, -2}, {20, 19, 1, -5}, {16., 39.}}
```

```
In[94]:= val = a
Out[94]=
{{{x → 14.3333, y → 15.3333}}, {{x → 11.6667, y → 16.6667}},
 {{x → 6.2, y → 19.4}}, {}, {}, {{x → -6., y → -5.}}, {}, {{x → -2., y → 3.}}, {}, {}}

In[63]:= a =.

In[64]:= a
Out[64]=
a
```

```
In[29]:= a = {{{x → 14.333333333333334`, y → 15.333333333333334`}},
            {{x → 11.666666666666666`, y → 16.666666666666668`}}, {{x → 6.2`, y → 19.4`}},
            {{x → 21.444444444444443`, y → 11.777777777777779`}}, {},
            {{x → -6.`, y → -5.`}}, {{x → 19.666666666666668`, y → 20.666666666666668`}},
            {{x → -2.`, y → 3.`}}, {{x → 19.`, y → 24.`}}, {{x → 16.`, y → 39.`}}}

Out[29]=
         {{{x → 14.3333, y → 15.3333}}, {{x → 11.6667, y → 16.6667}}, {{x → 6.2, y → 19.4}},
          {{x → 21.4444, y → 11.7778}}, {}, {{x → -6., y → -5.}}, {{x → 19.6667, y → 20.6667}},
          {{x → -2., y → 3.}}, {{x → 19., y → 24.}}, {{x → 16., y → 39.}}}

In[36]:= {x, y} /. a⟦1⟧ // Flatten

Out[36]=
         {14.3333, 15.3333}

In[39]:= val

Out[39]=
         val

In[40]:= x

Out[40]=
         x

In[41]:= y

Out[41]=
         y

In[47]:= val

Out[47]=
         val

In[95]:= Module[{
                val = {x, y} /. #⟦1⟧
              },
              EchoLabel["val"]@val;
              7 ≤ val⟦1⟧ ≤ 27 && 7 ≤ val⟦2⟧ ≤ 27
             ] & /@ Select[a, Length[#] ≠ 0 &] // Counts

      » val {14.3333, 15.3333}

      » val {11.6667, 16.6667}

      » val {6.2, 19.4}

      » val {-6., -5.}

      » val {-2., 3.}

Out[95]=
         <|True → 2, False → 3|>
```

```
In[21]:= 200 000 000 000 000
Out[21]=
         200 000 000 000 000

In[22]:= 400 000 000 000 000
Out[22]=
         400 000 000 000 000

In[23]:= ScientificForm[N[400 000 000 000 000, 15]]
Out[23]//ScientificForm=
```

$4.00000000000000 \times 10^{14}$

```
In[24]:= ScientificForm[N[200 000 000 000 000, 15]]
Out[24]//ScientificForm=
```

$2.00000000000000 \times 10^{14}$

```
In[25]:= 2
Out[25]=
         2

In[26]:= SetDirectory["~/nhannht-projects/aoc2023/"]
Out[26]=
         /home/vermin/nhannht-projects/aoc2023

In[27]:= DictionaryLookup["achi" ~~ __]
Out[27]=
         {achier, achiest, achievable, achieve, achieved, achievement,
           achievements, achiever, achievers, achieves, achieving, aching, achingly}

In[28]:= NotebookSave[EvaluationNotebook[], FileNameJoin[{Directory[], "24.nb"}]]

In[221]:=
         VerminExportKeepSyntaxHighLight[]

In[222]:=
         Export[FileNameJoin[{Directory[], "24.pdf"}], EvaluationNotebook[]]
```