Read the data

```
In[127]:= SetDirectory[NotebookDirectory[]];

In[128]:= fname = "example.txt";

In[129]:= (in = ReadList[fname, String]) ~Take~ 10 // TableForm

Out[129]//TableForm=
          initial state: #..#.#..##......###...###
          ...## => #
          ..#.. => #
          .#... => #
          .#.#. => #
          .#.## => #
          .##.. => #
          .#### => #
          #.#.# => #
          #.### => #

In[130]:= replace[c_] := StringReplace[c, {"." → "0", "#" → "1"}] // ToExpression

In[131]:= pad[list_, l_, r_] := PadRight[PadLeft[list, Length[list] + l], Length[list] + l + r];

In[132]:= readInput[fname_] :=
           Module[{initialString, initial, rules, data, zeros, rulesList},
            data = ReadList[fname, String];
            initialString = StringReplace[data[[1]], "initial state: " → ""];
            initial = replace /@ (initialString // Characters);
            rulesList = Map[StringSplit[#, " => "] &, Drop[data, 1]];
            rules = {replace /@ Characters[#[[1]]] → replace[#[[2]]]} & /@ rulesList;
            rules = Flatten[rules, 1];
            zeros = {{_, _, _, _, _} → 0};
            {Join[rules, zeros], initial}
           ]

In[133]:= {rules, initial} = readInput[fname];

In[134]:= rules

Out[134]= {{0, 0, 0, 1, 1} → 1, {0, 0, 1, 0, 0} → 1, {0, 1, 0, 0, 0} → 1,
           {0, 1, 0, 1, 0} → 1, {0, 1, 0, 1, 1} → 1, {0, 1, 1, 0, 0} → 1, {0, 1, 1, 1, 1} → 1,
           {1, 0, 1, 0, 1} → 1, {1, 0, 1, 1, 1} → 1, {1, 1, 0, 1, 0} → 1, {1, 1, 0, 1, 1} → 1,
           {1, 1, 1, 0, 0} → 1, {1, 1, 1, 0, 1} → 1, {1, 1, 1, 1, 0} → 1, {_, _, _, _, _} → 0}
```

Part 1

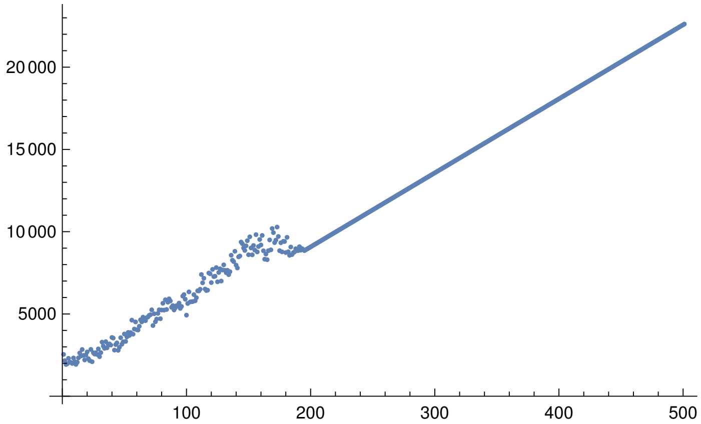
```
In[35]:= params1 = {"left" → 10, "right" → 20};

In[38]:= padded = pad[initial, "left" /. params1, "right" /. params1];
```

```
In[39]:= out = CellularAutomaton[rules, padded, 20];

In[45]:= indexCount[row_, params_] := Module[{indexed},
           indexed = MapIndexed[{#2[[1]] - ("left" /. params) - 1, #} &, row];
           Plus @@ Select[indexed, #[[2]] == 1 &][[All, 1]]
          ]

In[46]:= indexCount[out[[-1]], params1]

Out[46]= 2281
```

Part 2

```
         params2 = {"left" → 500, "right" → 1000};

In[112]:= padded = pad[initial, "left" /. params2, "right" /. params2];

In[120]:= out  =  CellularAutomaton[rules,  padded,  500];

In[121]:= (list = indexCount[#, params2] & /@ out) // Length

Out[121]= 501

In[122]:= ListPlot[list]
```

Out[122]=



```
In[117]:= Drop[list, 1] - Drop[list, -1]

Out[117]= {-378, -232, 43, 330, -230, -21, -61, 334, -213, -179, 134, 266, 293, -179, 392, -359,
           -275, 297, 190, -411, -115, 680, -752, 549, -80, 78, -102, 337, -474, 248, 638,
           -238, -139, 405, -376, 194, 42, -66, 461, -38, -730, 336, 102, -452, 186, 580,
           -395, 146, 474, -449, 276, 270, -187, 178, 757, -854, 311, 433, -474, -17, 228,
           394, -126, 286, -83, -129, 180, 45, 95, 39, 299, -957, 717, -484, 164, 343, 219,
           -542, 526, 410, -413, 628, -592, 446, 203, -131, -386, 105, -264, 128, 133, -16,
           170, -321, 118, 626, 95, -282, -968, 709, 706, -610, 13, 7, 424, -376, 189, 419,
           -8, 103, 896, -507, 279, -656, -90, 20, 1043, -47, -535, 802, -437, 22, 525, -864,
           566, 224, -744, 673, 311, -332, -145, 148, -266, 181, 1000, -302, -86, 627, -842,
           -174, 677, 52, 850, -102, -261, -150, 283, 310, -853, 1090, -676, -419, 567, -279,
           939, -1051, 329, 426, -333, 581, -934, -511, 313, -343, 549, 652, -602, 1296,
-250, -608, 152, 794, -573, -862, 481, -555, 642, -12, -670, 926, -879, -213,
503, -462, 116, 43, 199, -110, -3, 238, -213, 98, -58, -67, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45}
```

```
In[124]:= n500 = list[[-1]]

Out[124]= 22 620
```

```
In[125]:= n500 + 45 * (50 000 000 000 - 500)

Out[125]= 2 250 000 000 120
```