GNOME SORT

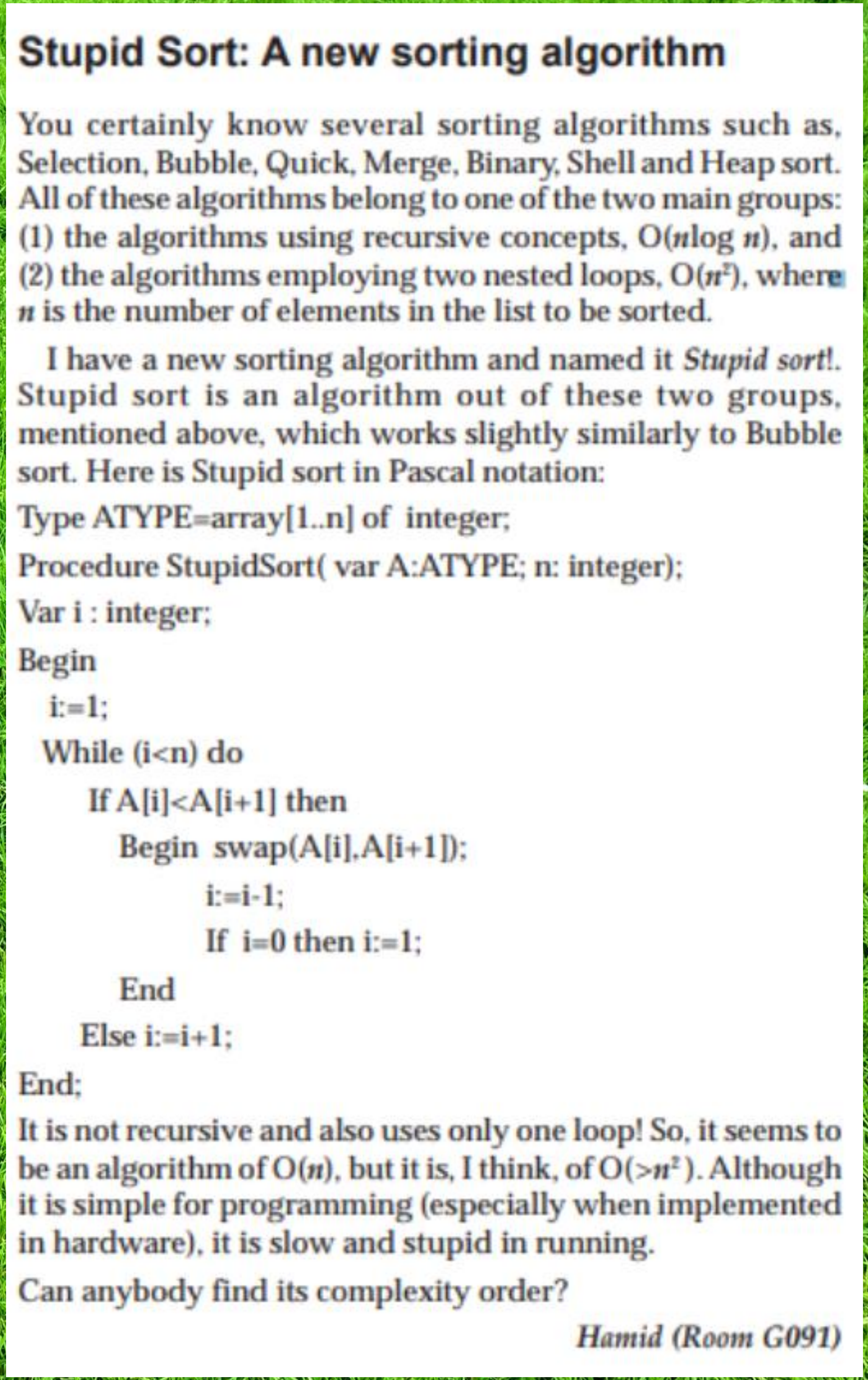
Brought to you by Team Stupid:
Raymond Aya-ay, Jomer Barcenilla,
Erik Celdran, Kevin Marte

Hamid Sarbazi-Azad

Professor of Computer Science and Engineering at Sharif University of Technology

- **Gnome sort was originally proposed by the Iranian computer scientist in 2000.**

- **The sort was first called stupid sort and then later described by Dick Grune as a gnome sort.**



Stupid Sort: A new sorting algorithm

You certainly know several sorting algorithms such as, Selection, Bubble, Quick, Merge, Binary, Shell and Heap sort. All of these algorithms belong to one of the two main groups: (1) the algorithms using recursive concepts, $O(n\log n)$, and (2) the algorithms employing two nested loops, $O(n^2)$, where n is the number of elements in the list to be sorted.

I have a new sorting algorithm and named it *Stupid sort*!. Stupid sort is an algorithm out of these two groups, mentioned above, which works slightly similarly to Bubble sort. Here is Stupid sort in Pascal notation:

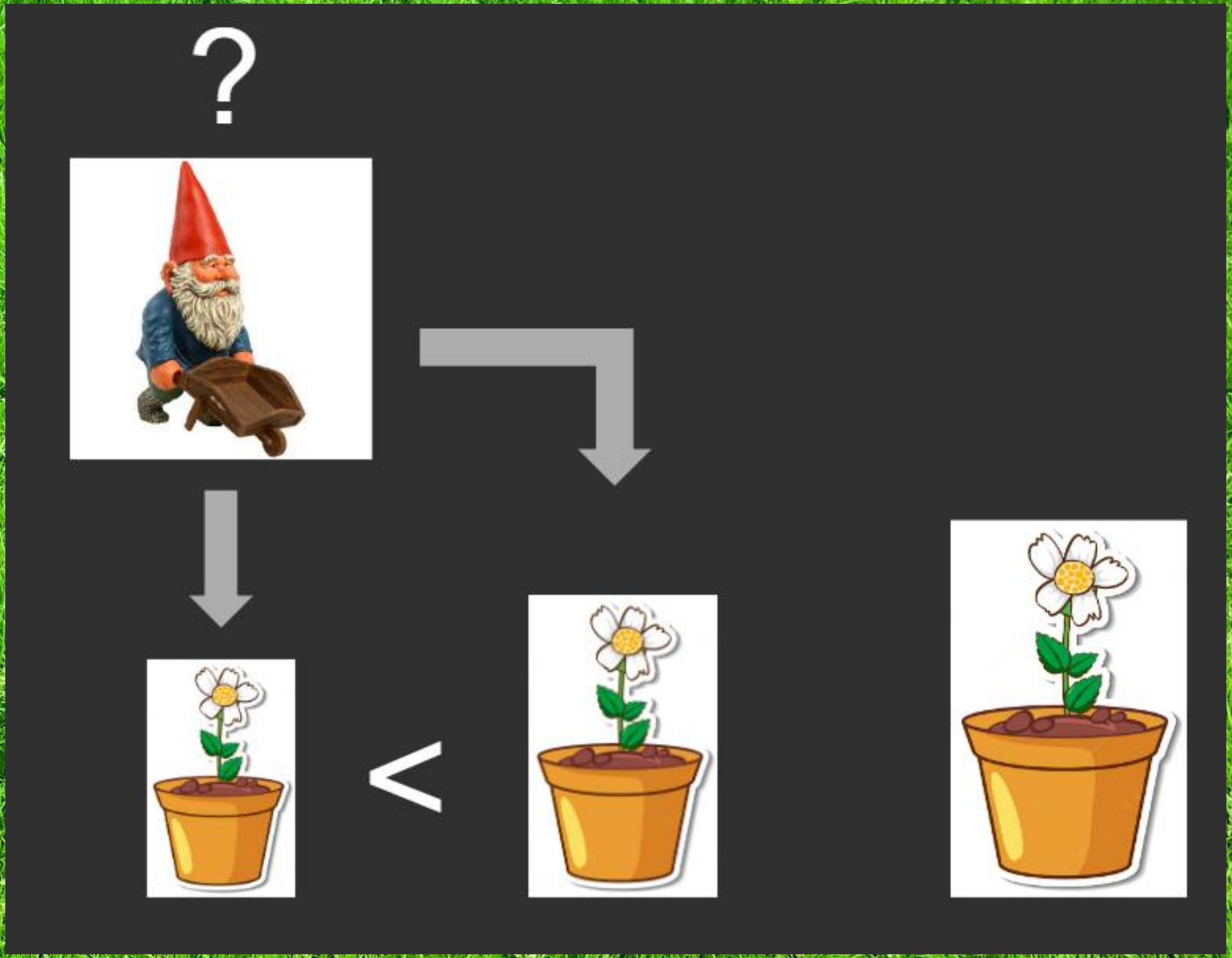
```
Type ATYPE=array[1..n] of  integer;
Procedure StupidSort( var A:ATYPE; n: integer);
Var i : integer;
Begin
  i:=1;
  While (i<n) do
      If A[i]<A[i+1] then
        Begin  swap(A[i],A[i+1]);
               i:=i-1;
               If  i=0 then i:=1;
        End
      Else i:=i+1;
End;
```

It is not recursive and also uses only one loop! So, it seems to be an algorithm of $O(n)$, but it is, I think, of $O(>n^2)$. Although it is simple for programming (especially when implemented in hardware), it is slow and stupid in running.

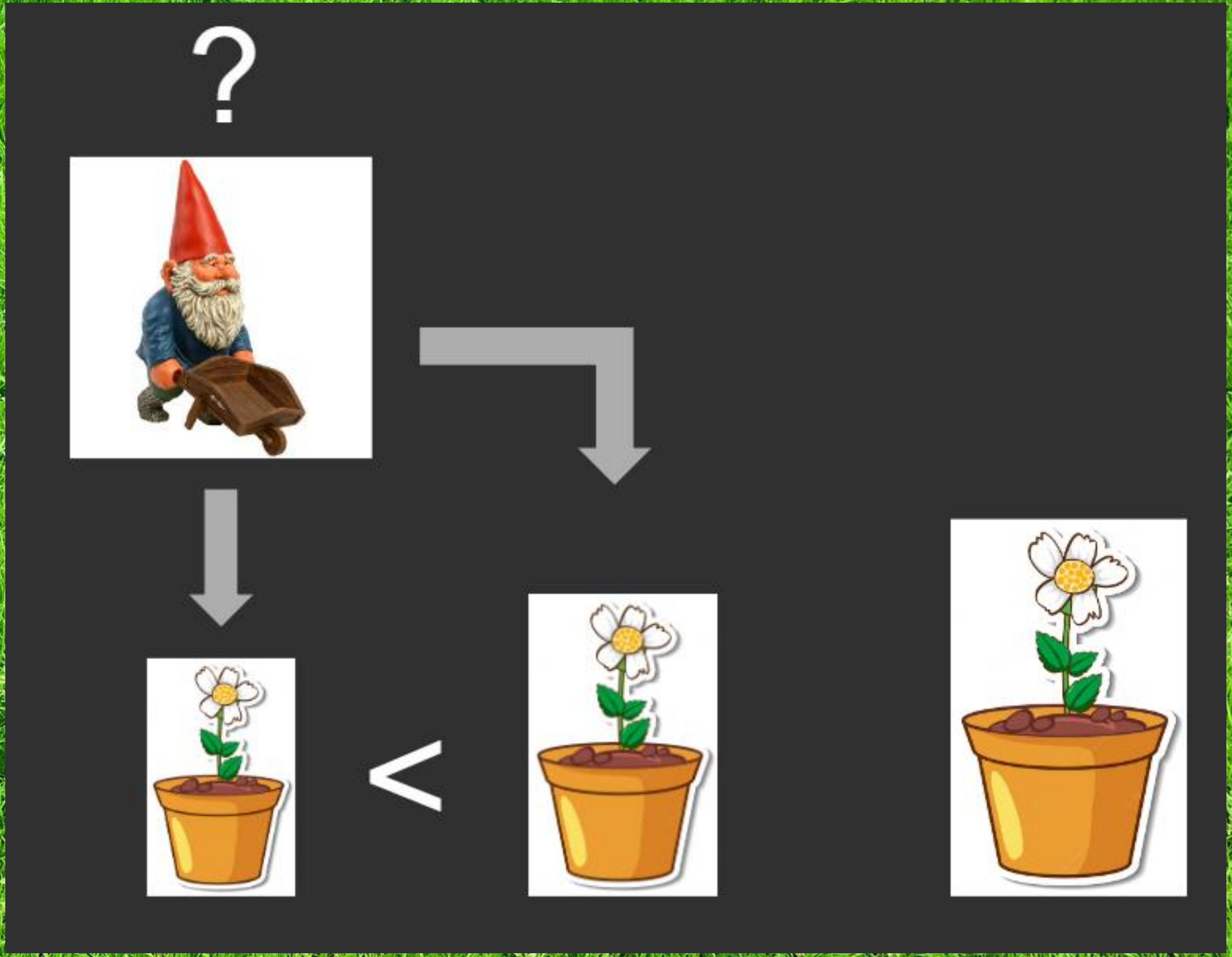
Can anybody find its complexity order?

Hamid (Room G091)

Gnome Sort is based on the technique used by the standard Dutch Garden Gnome



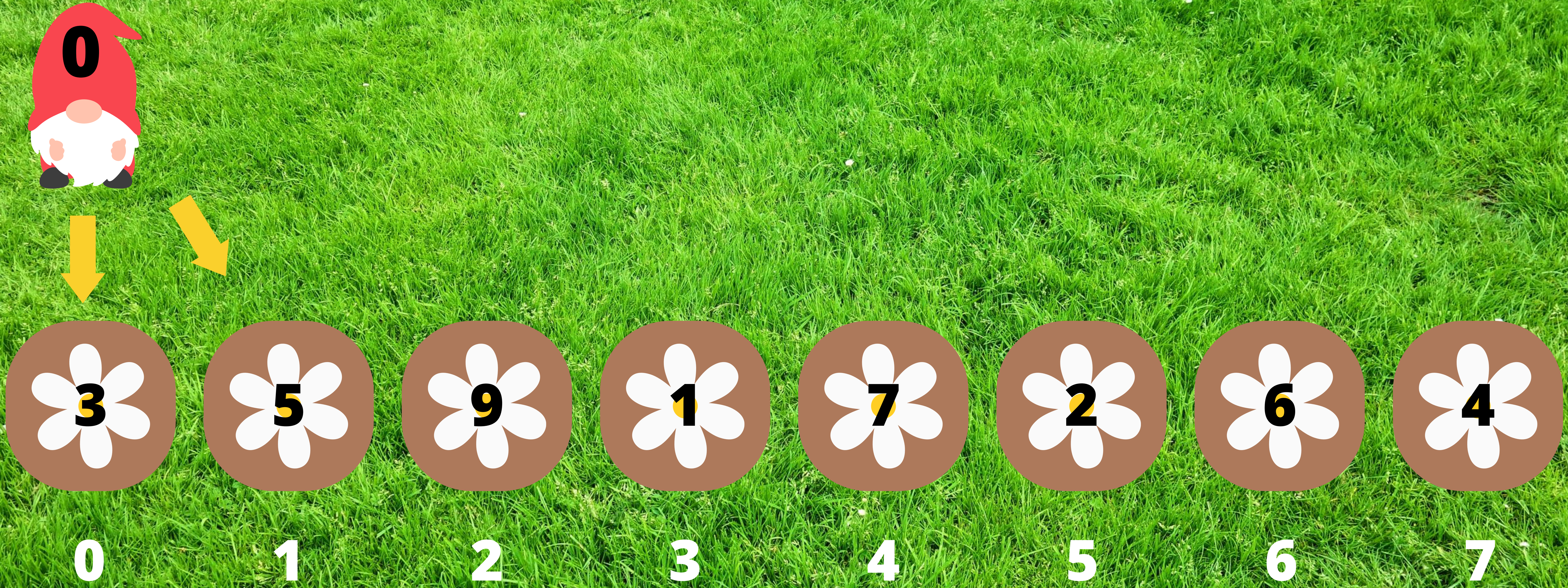
Basically, he looks at the flowerpot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise, he swaps them and steps one pot backward.



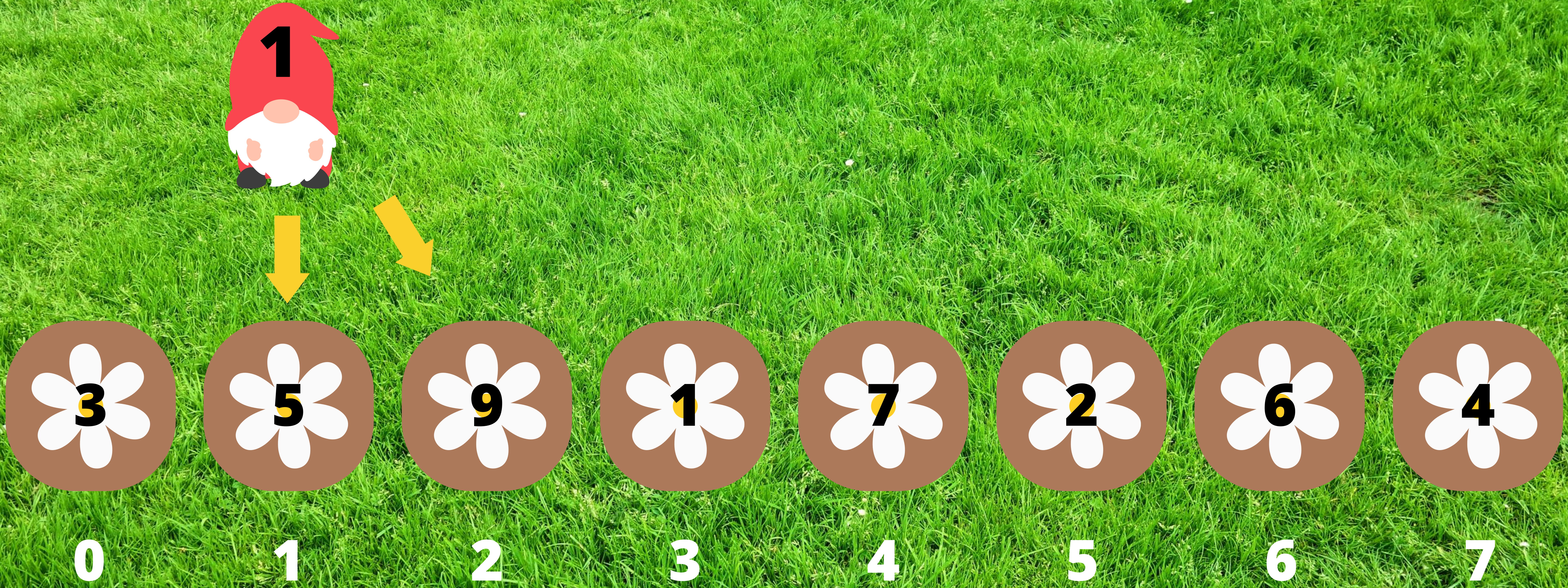
SIMULATION

Let's Sort my flowers

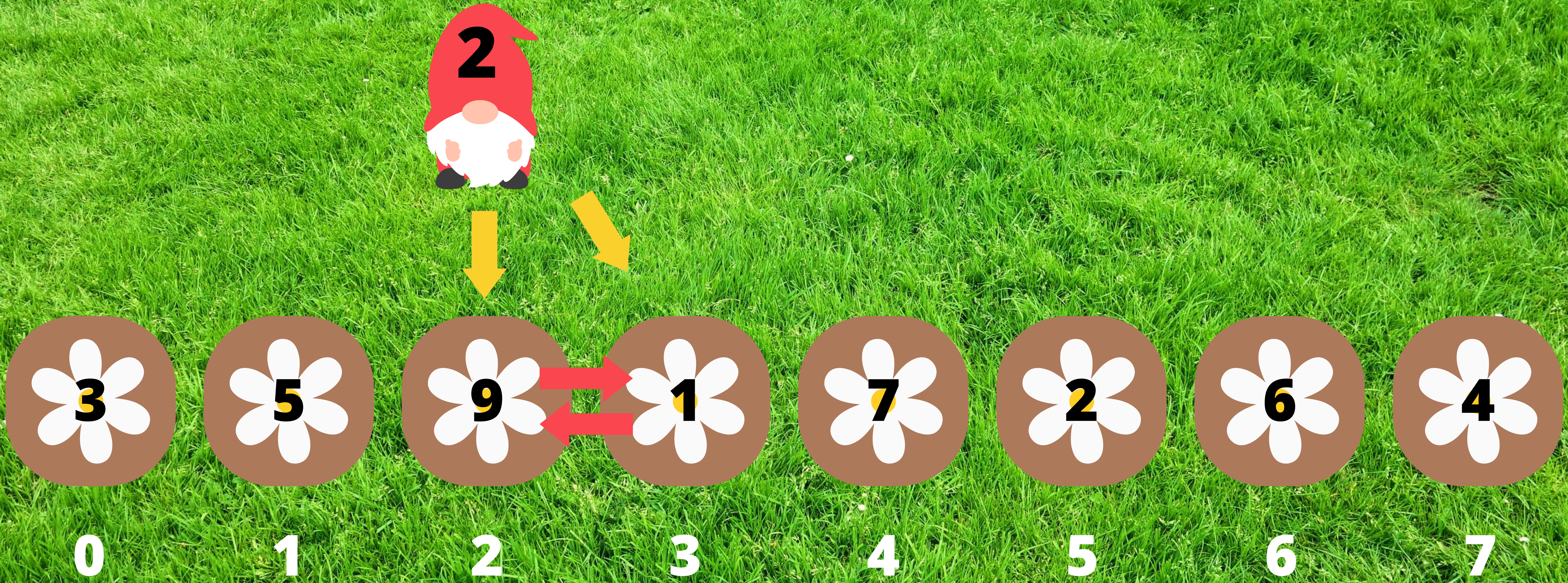
SIMULATION



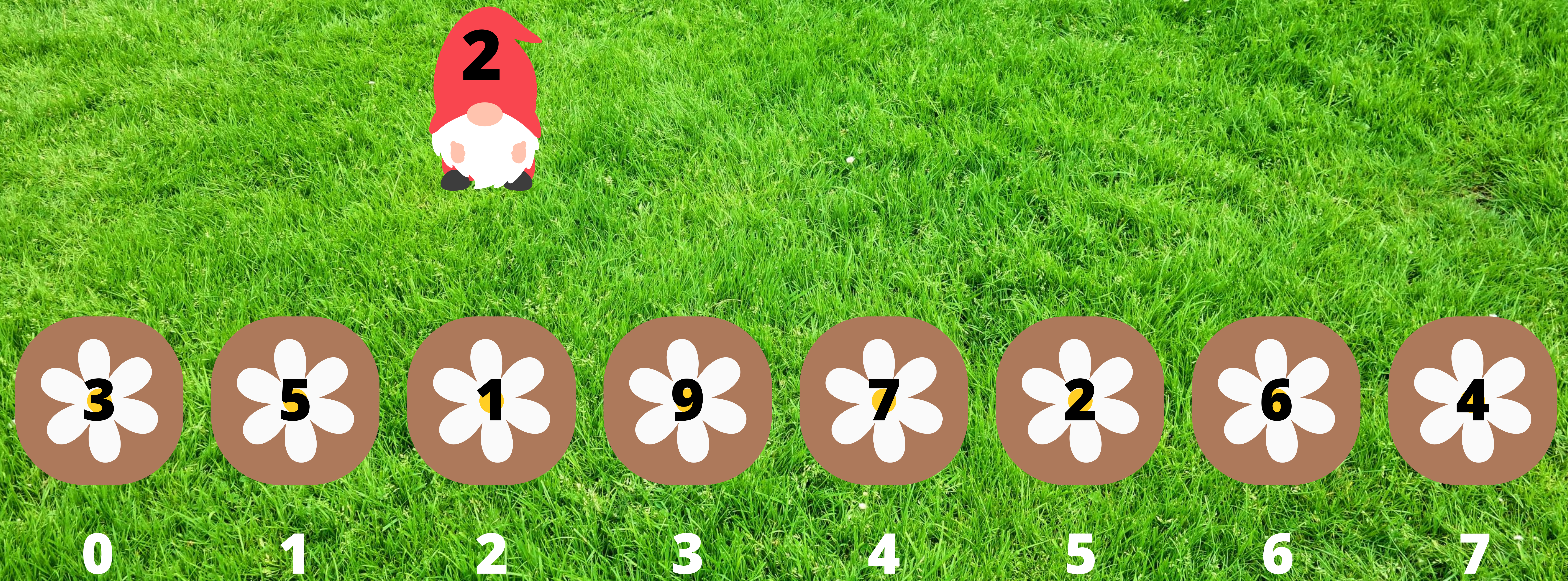
SIMULATION



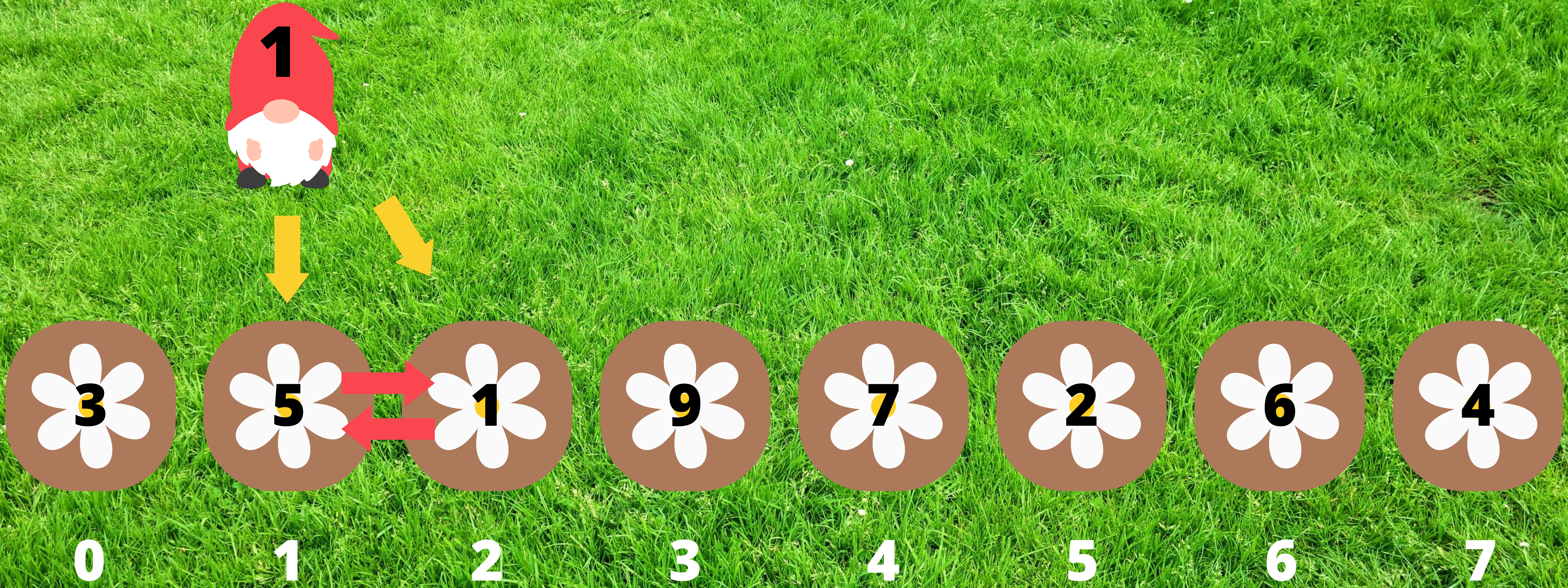
SIMULATION



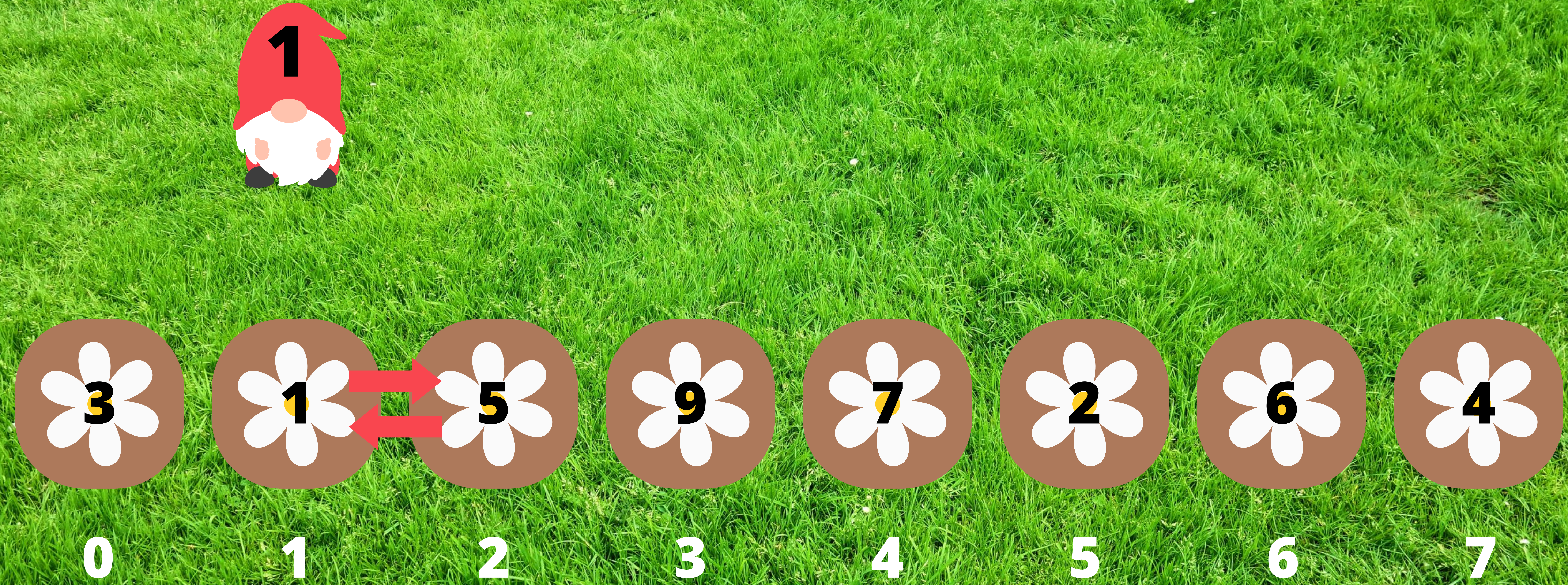
SIMULATION



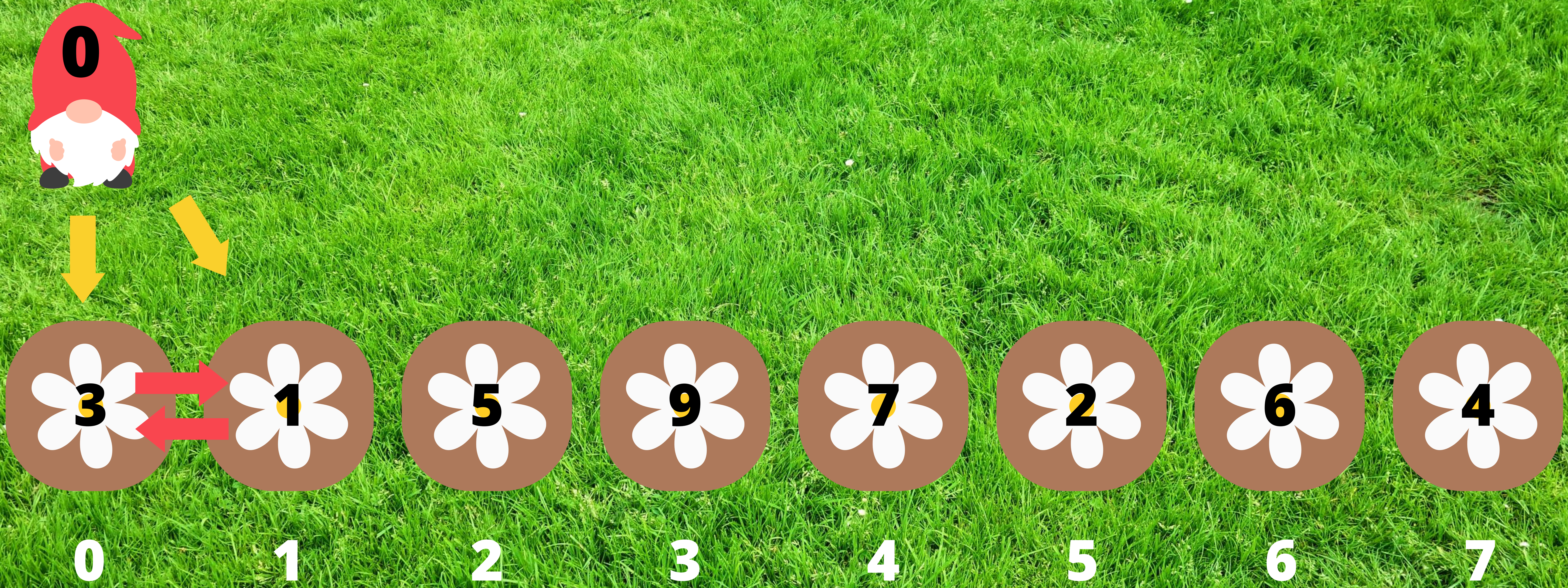
SIMULATION



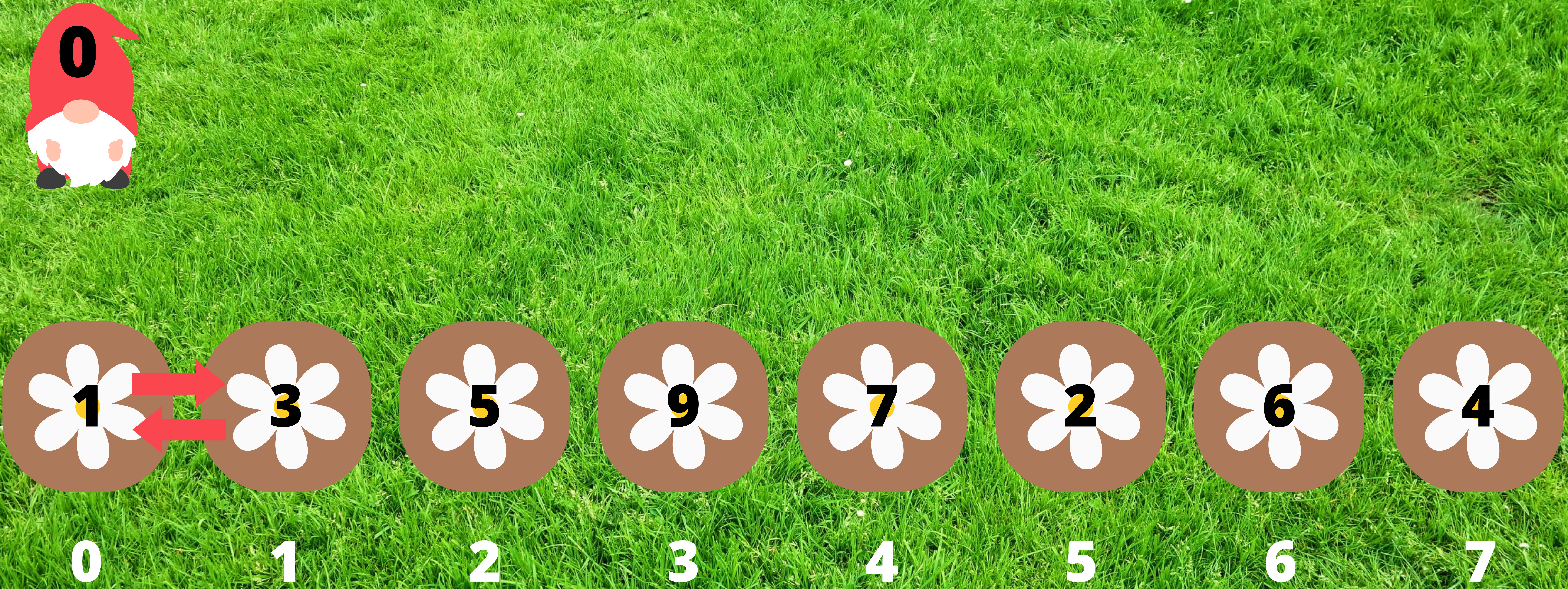
SIMULATION



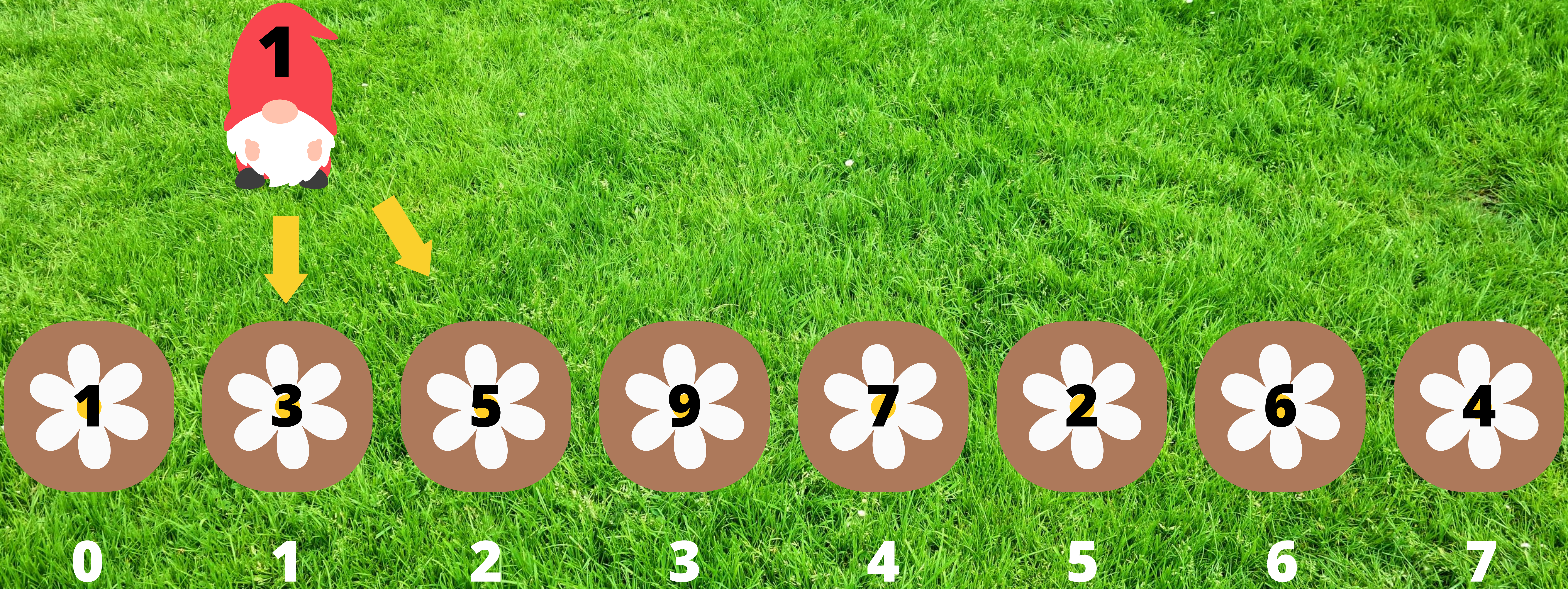
SIMULATION



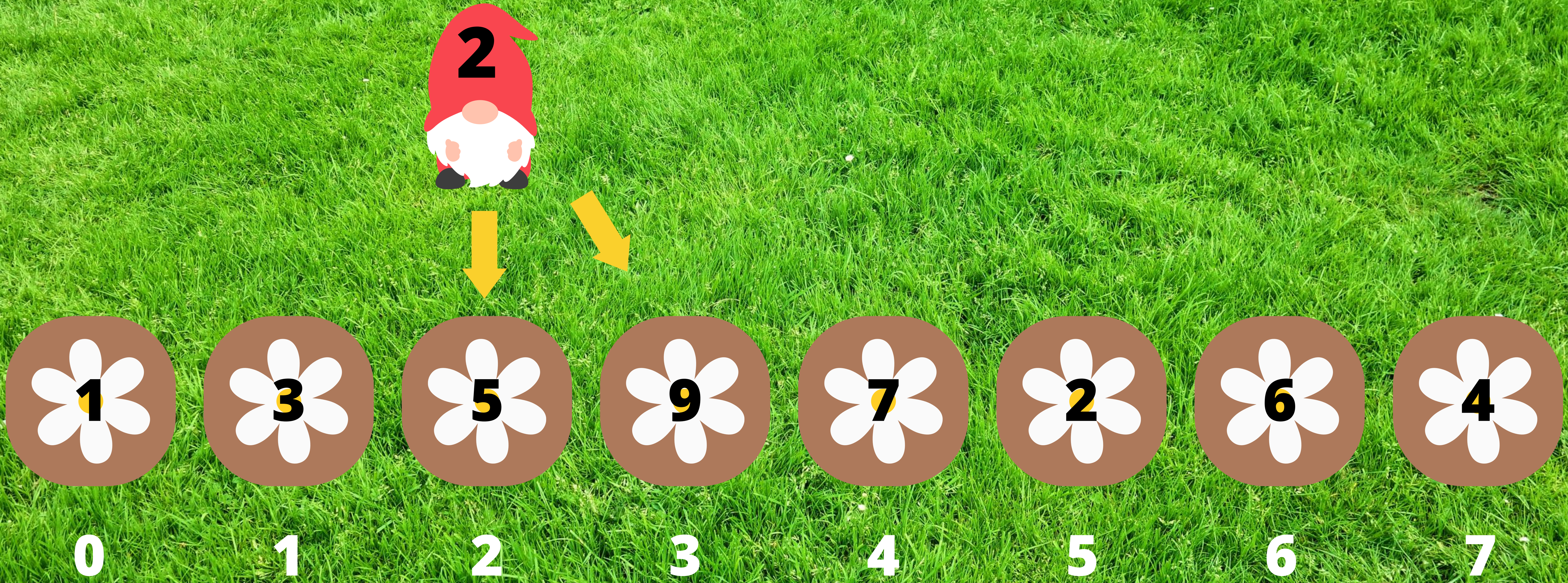
SIMULATION



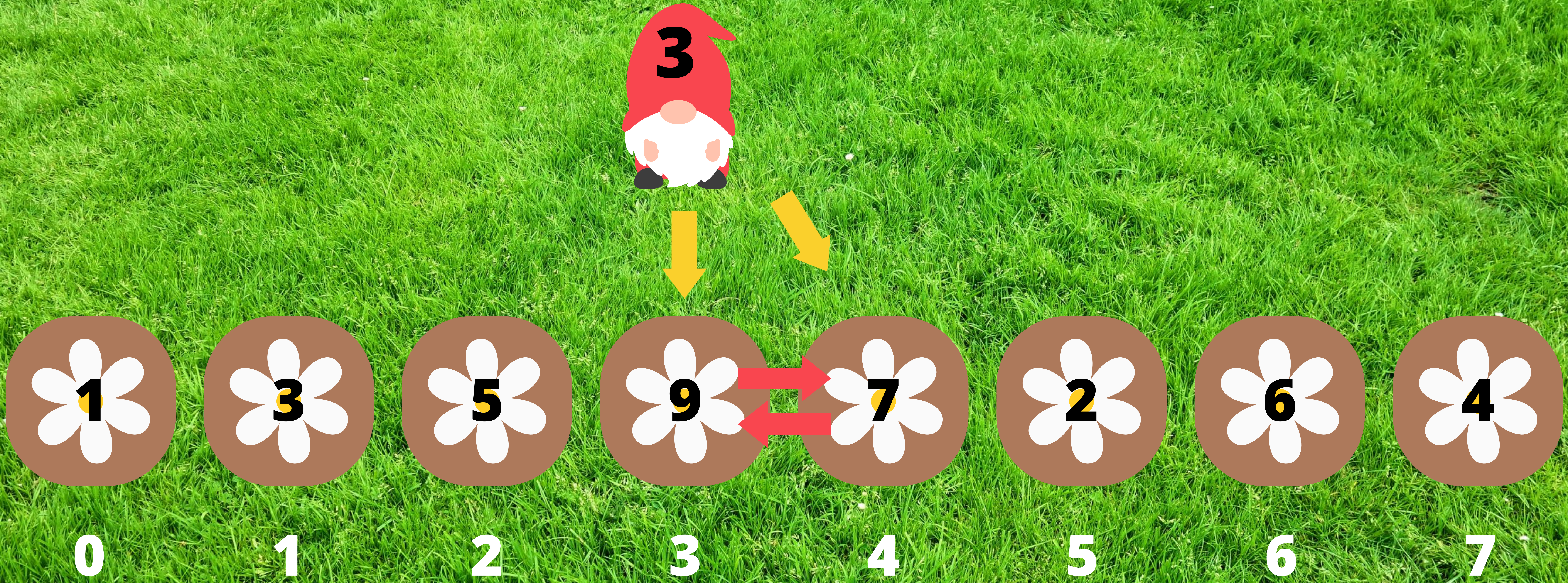
SIMULATION



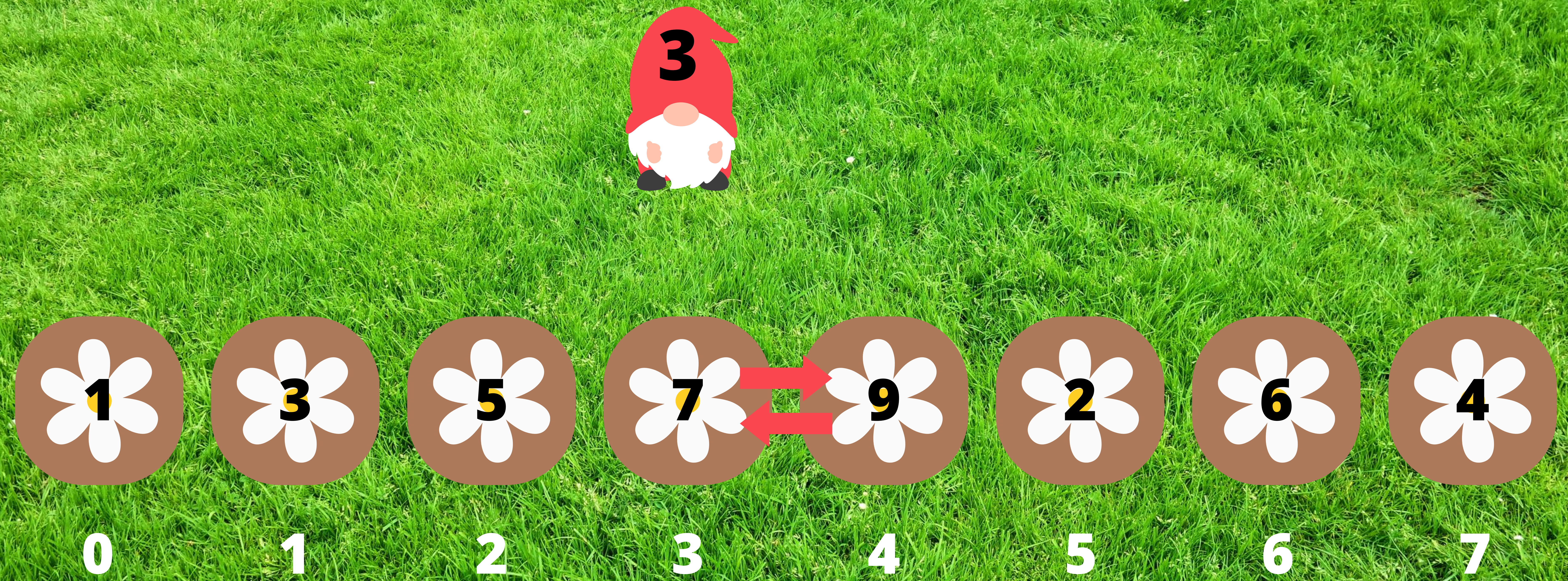
SIMULATION



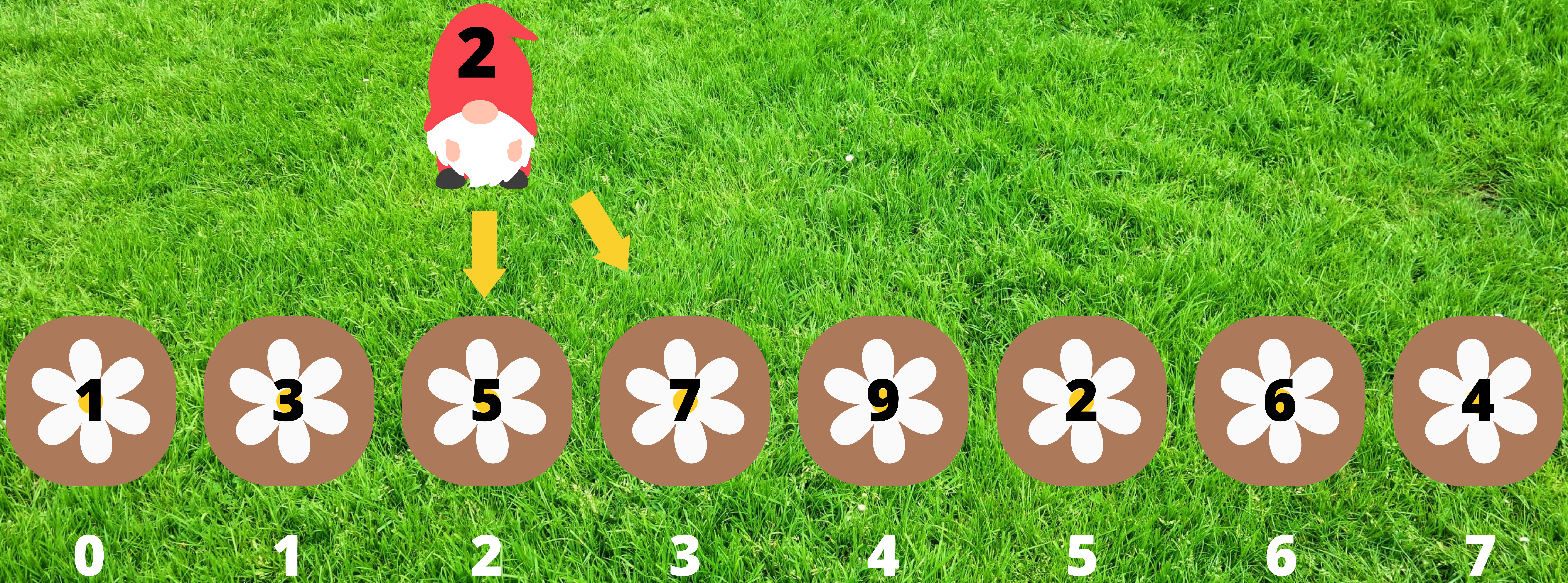
SIMULATION



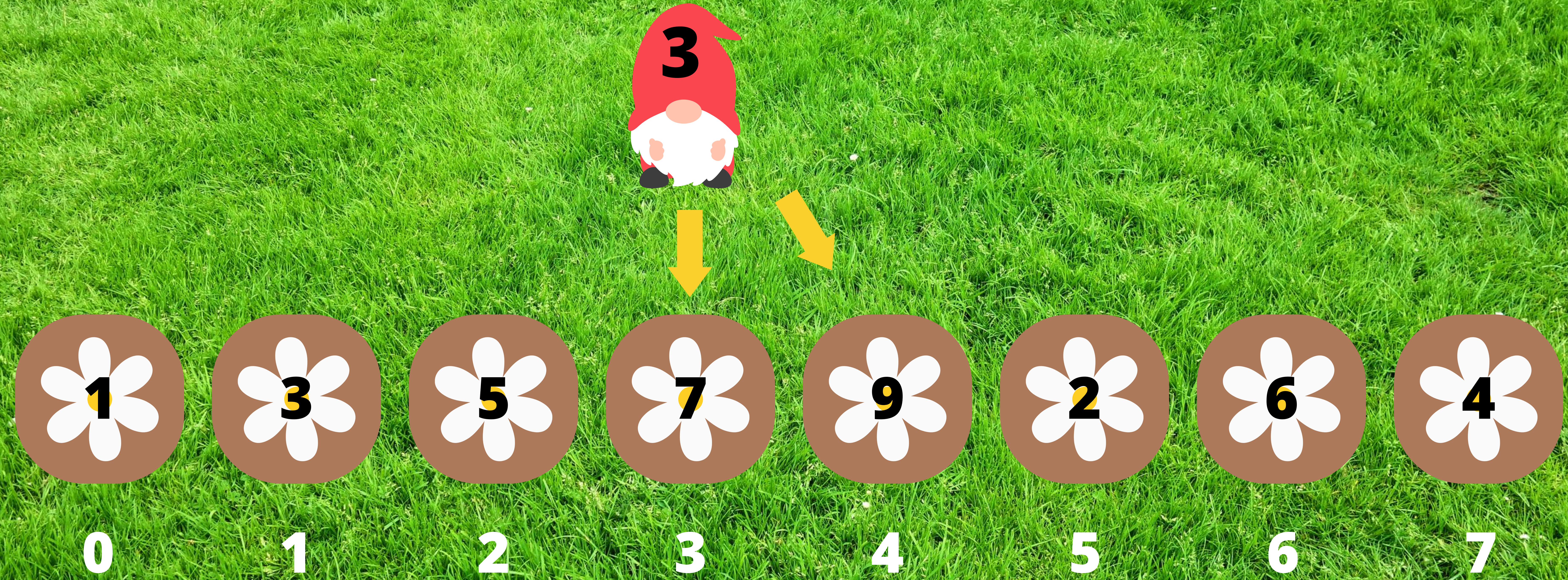
SIMULATION



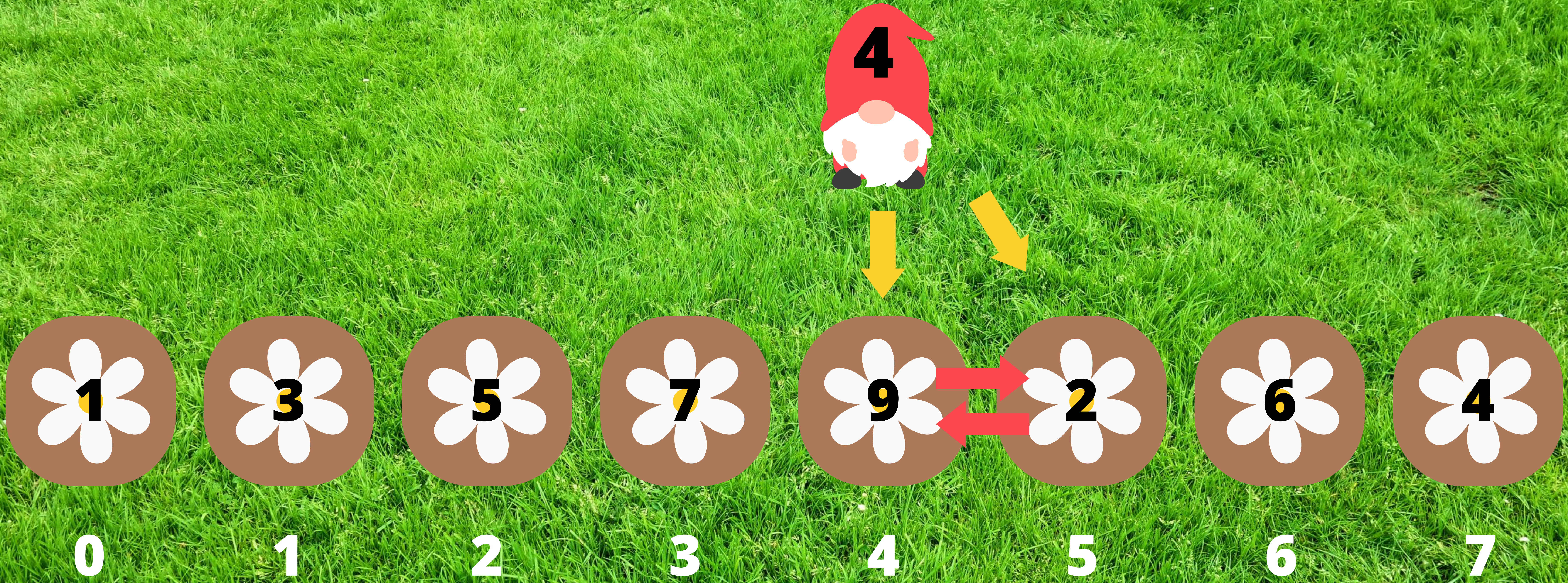
SIMULATION



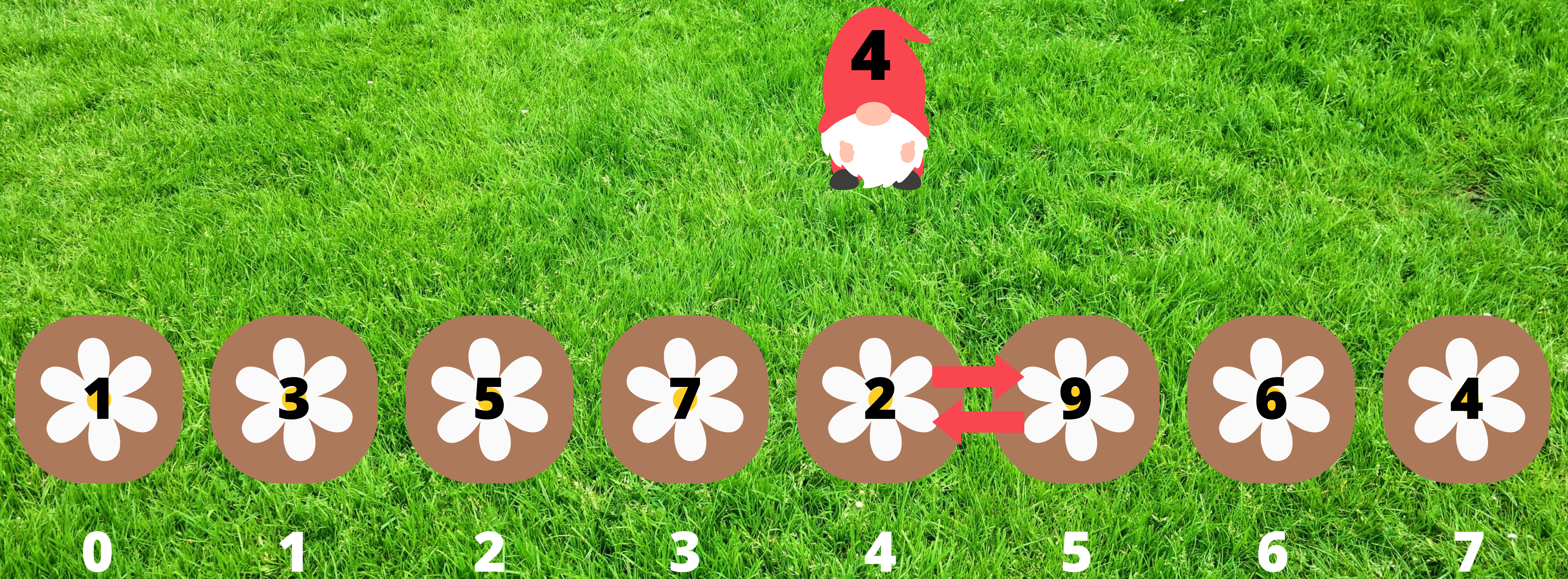
SIMULATION



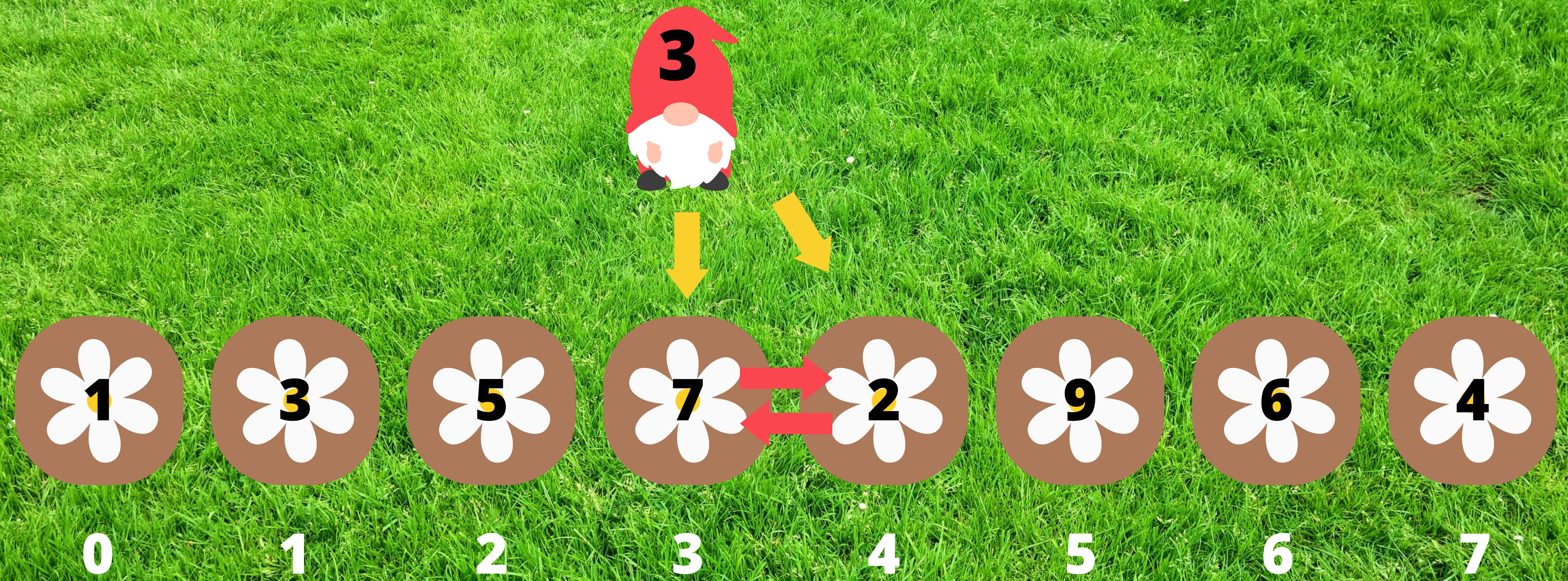
SIMULATION



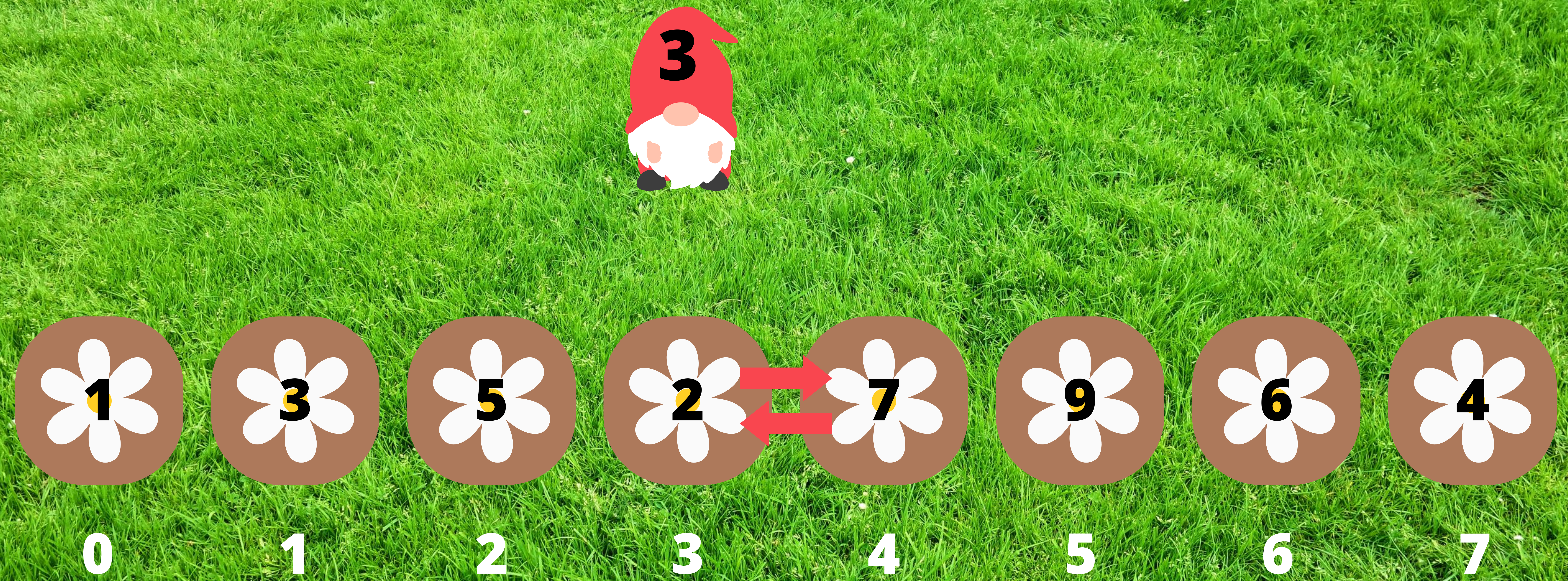
SIMULATION



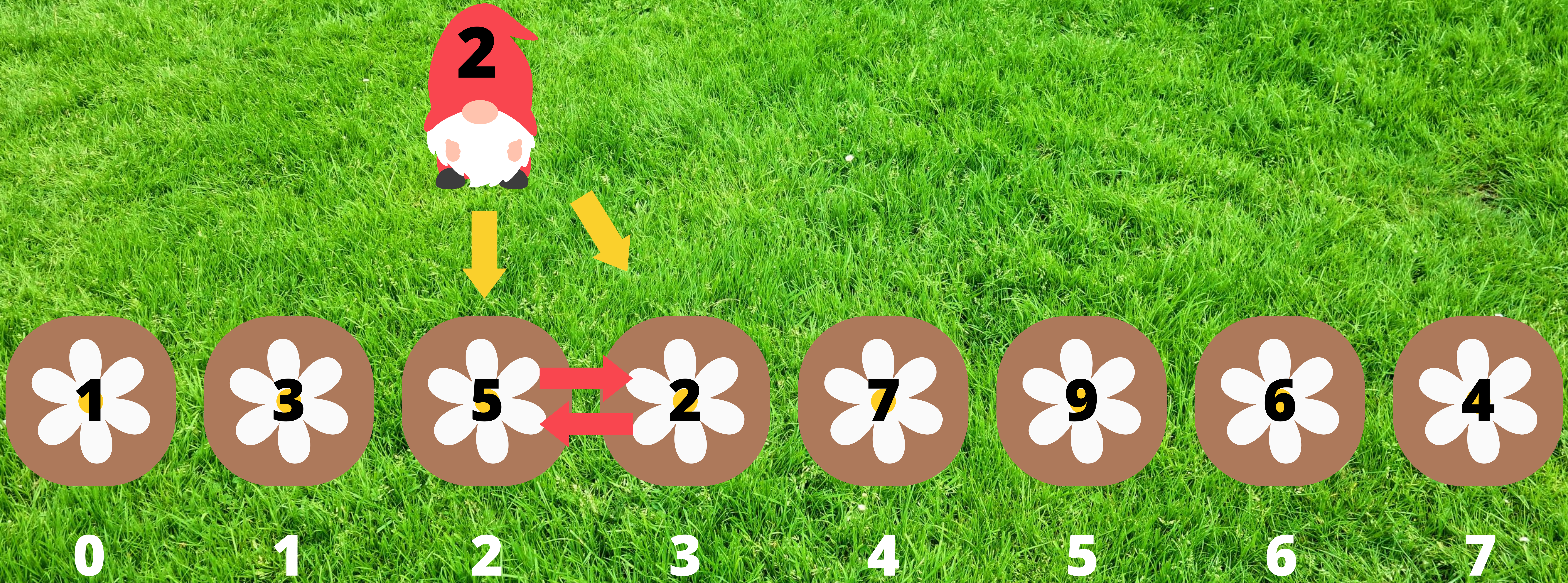
SIMULATION



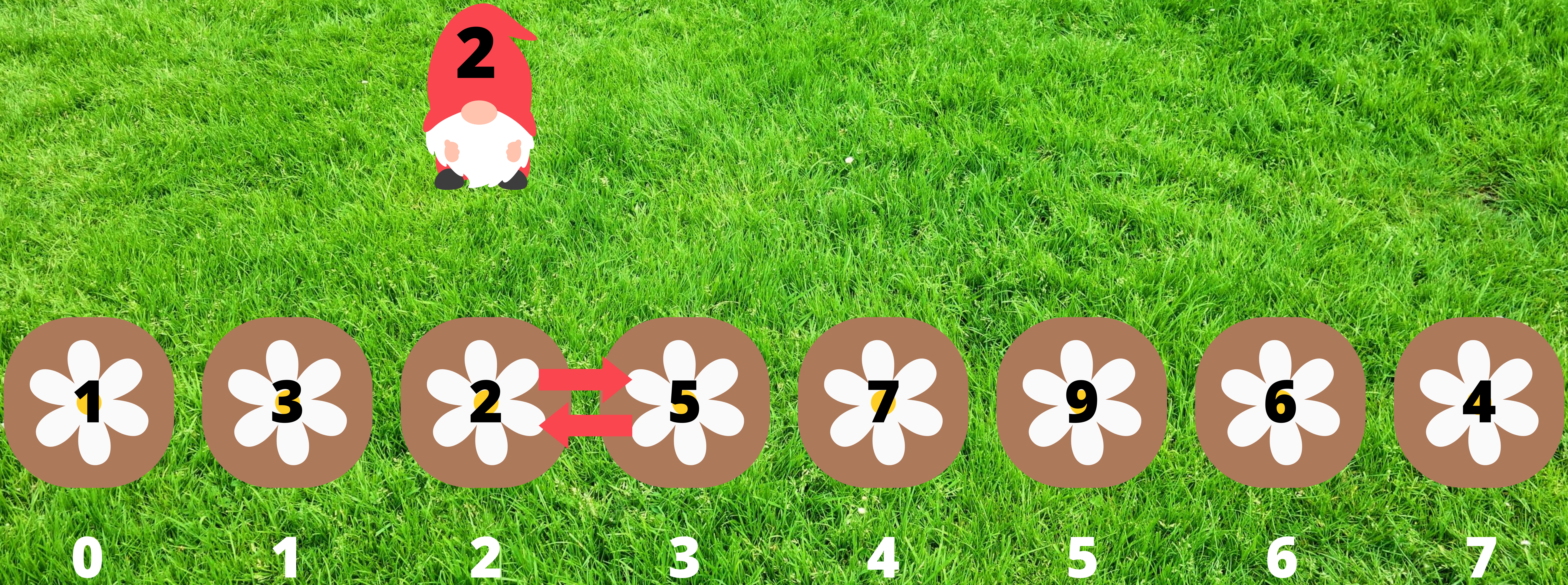
SIMULATION



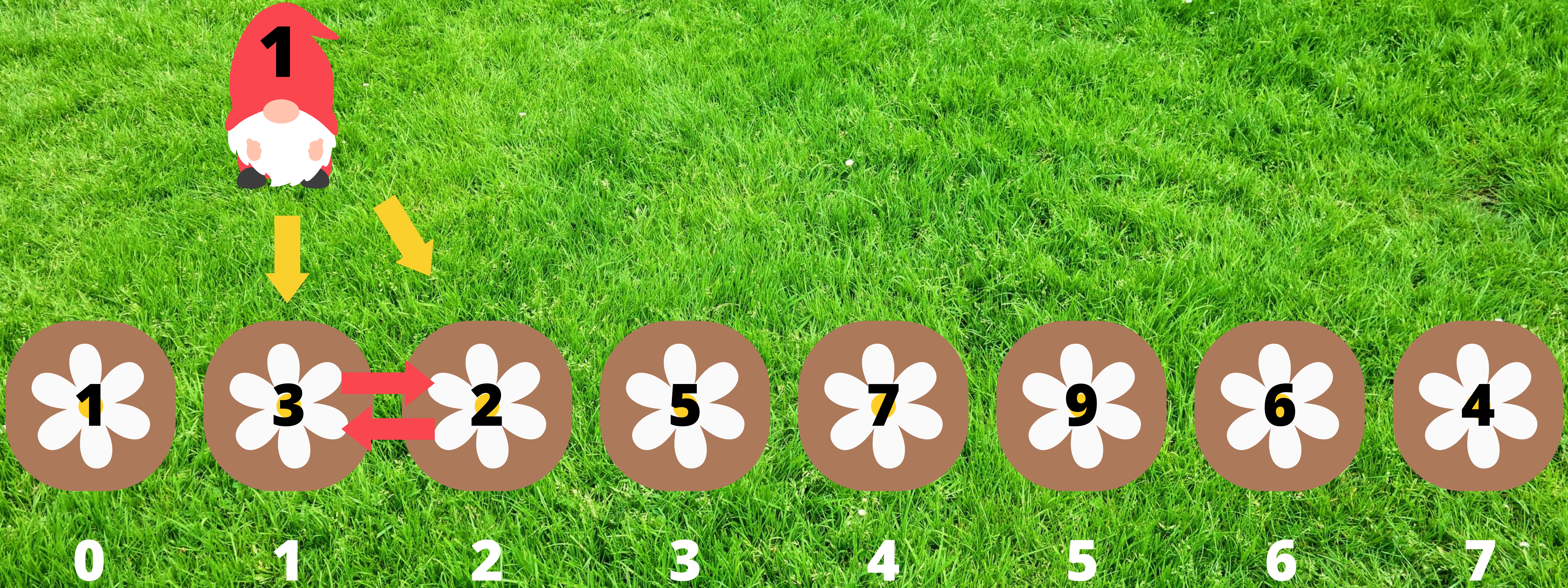
SIMULATION



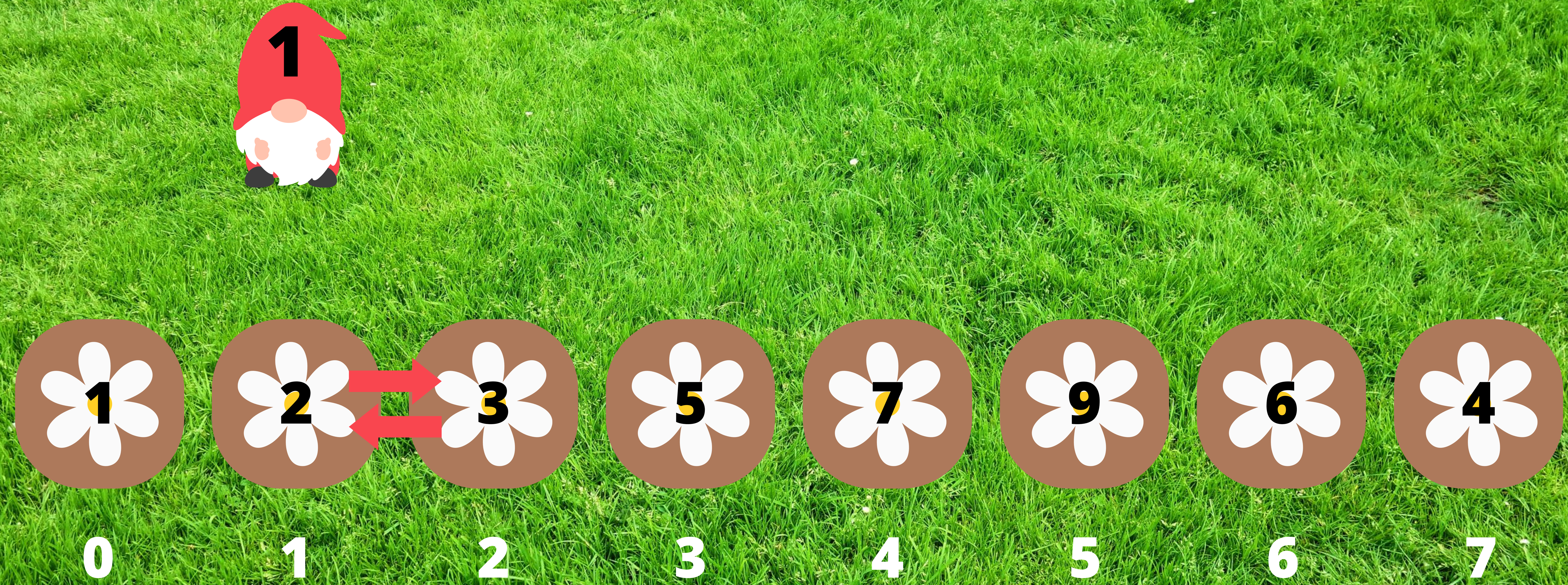
SIMULATION



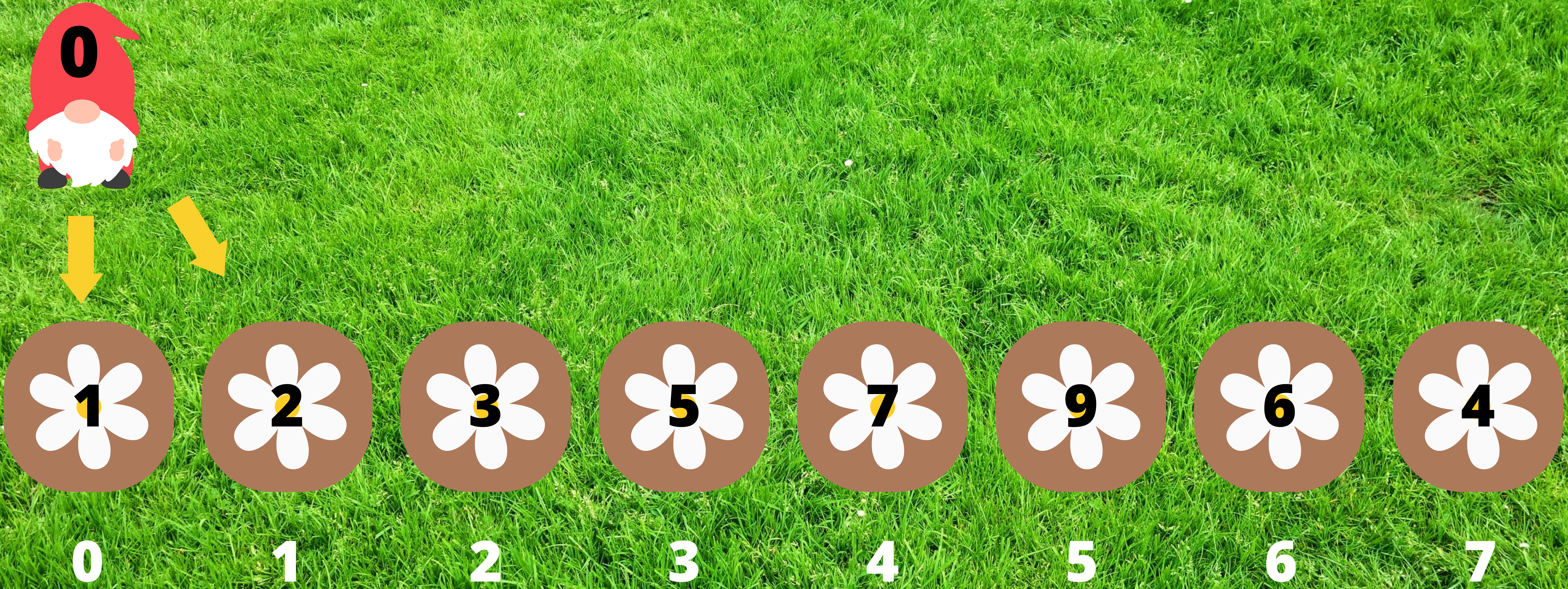
SIMULATION



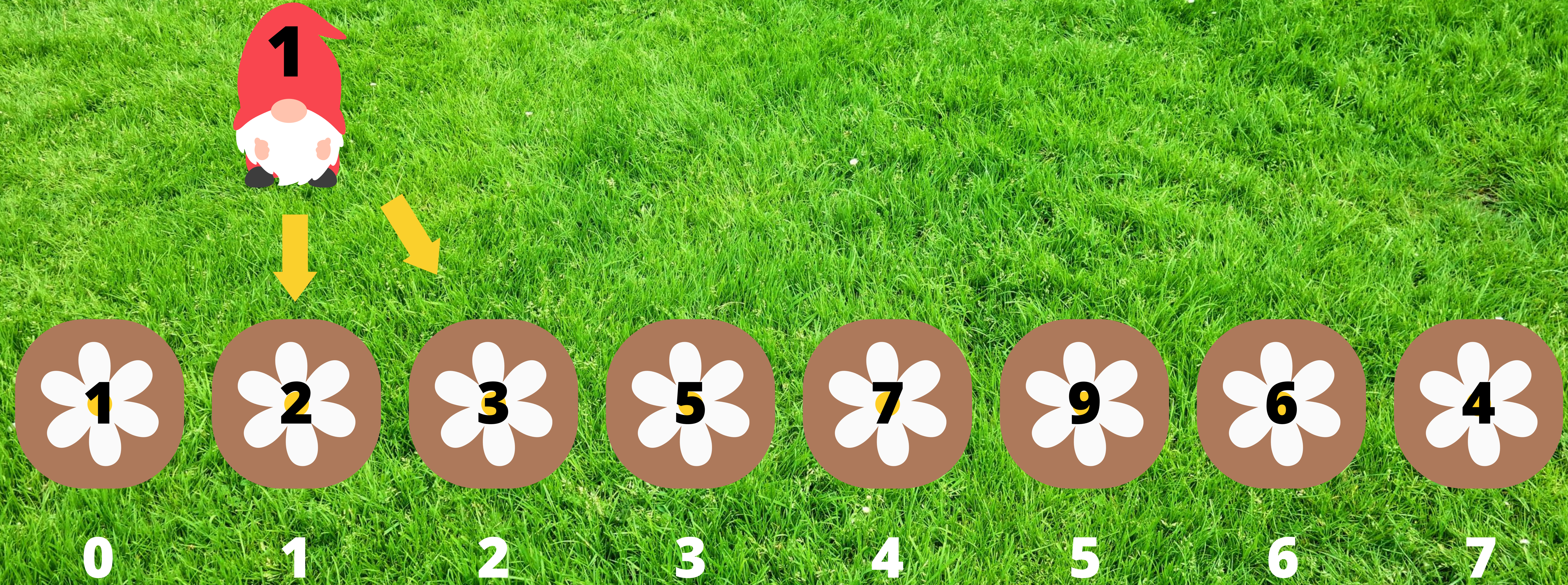
SIMULATION



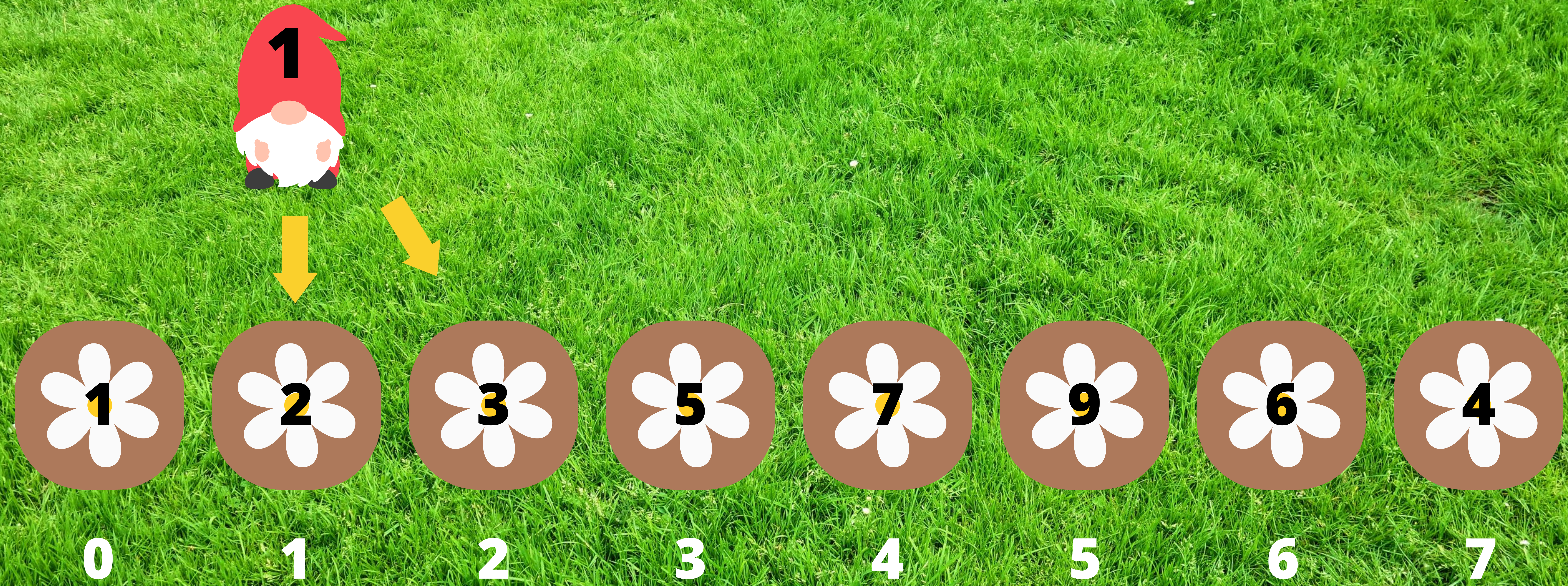
SIMULATION



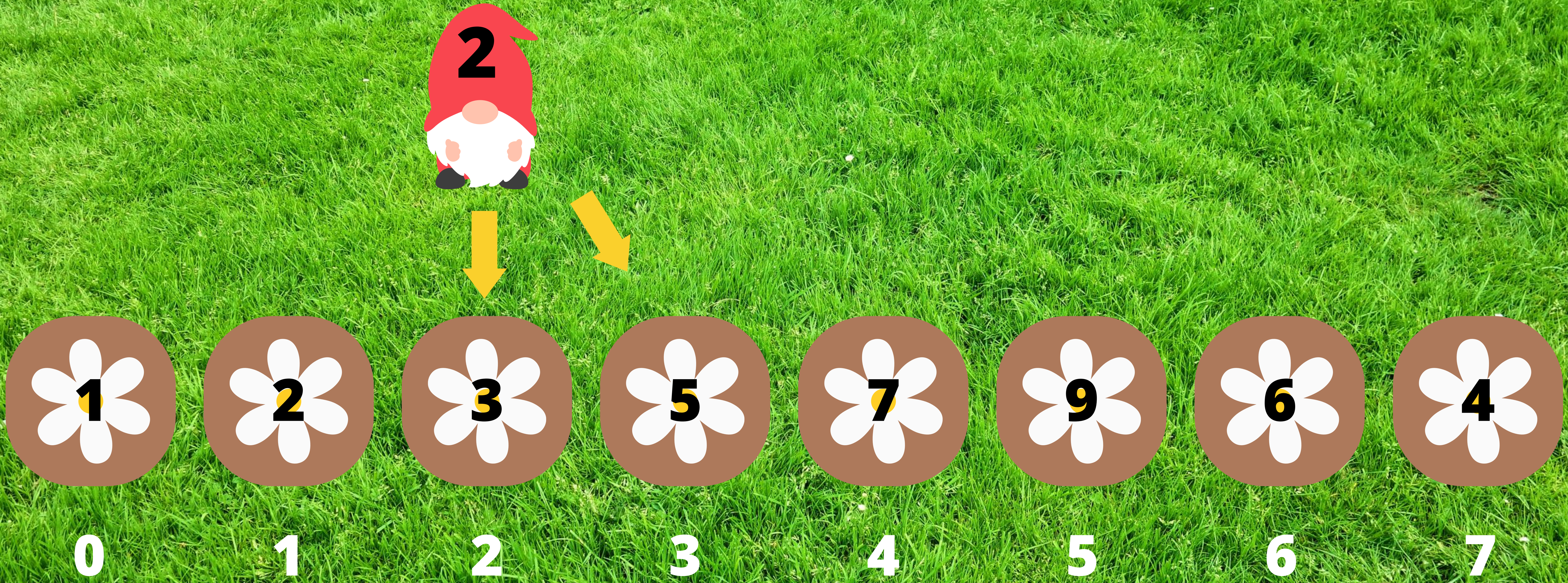
SIMULATION



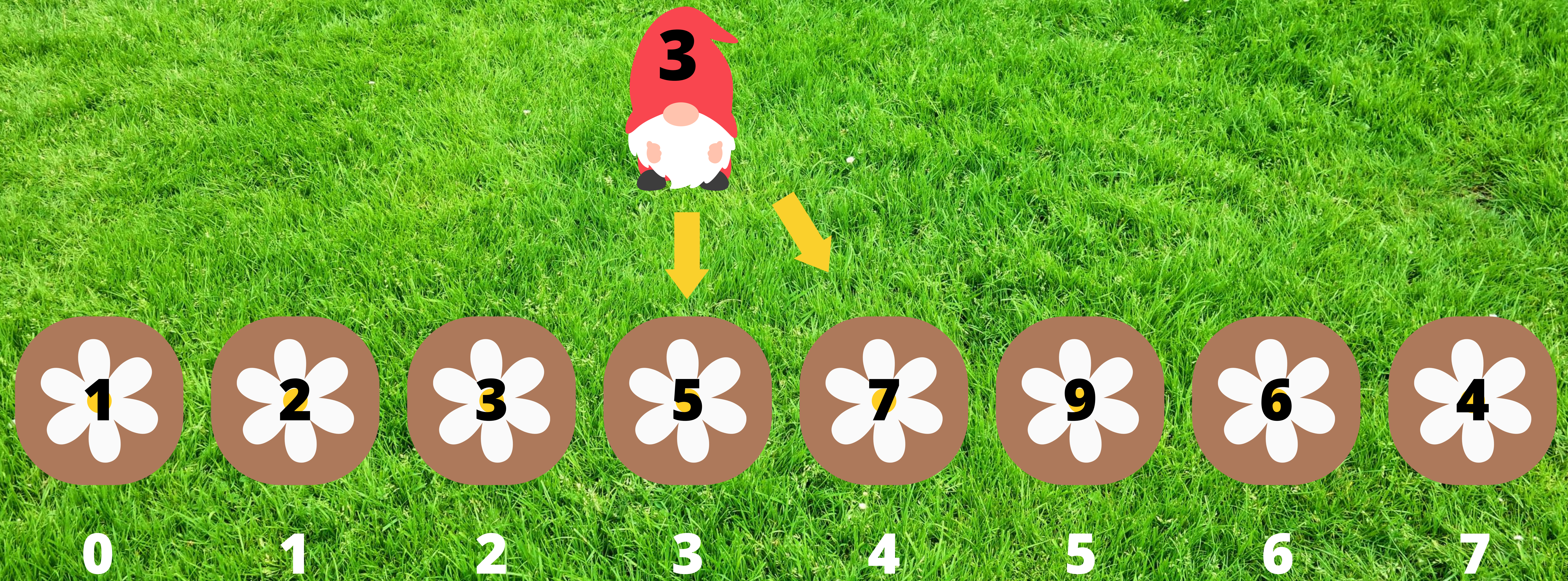
SIMULATION



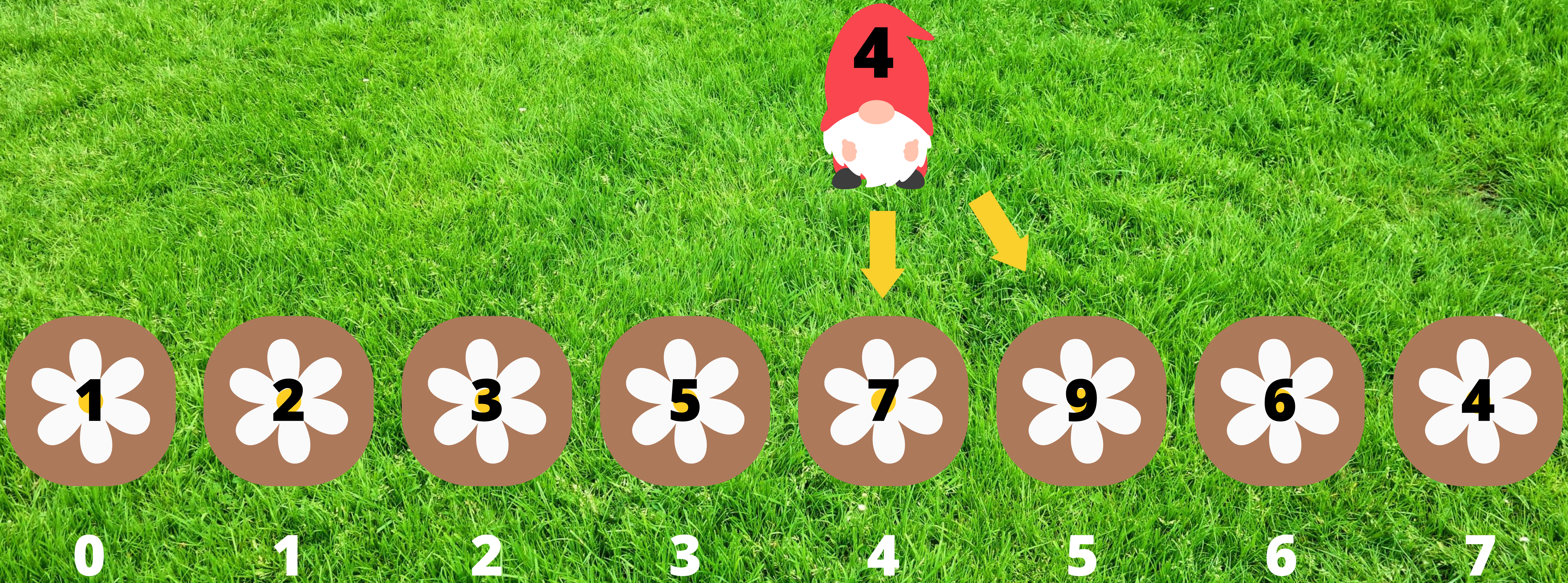
SIMULATION



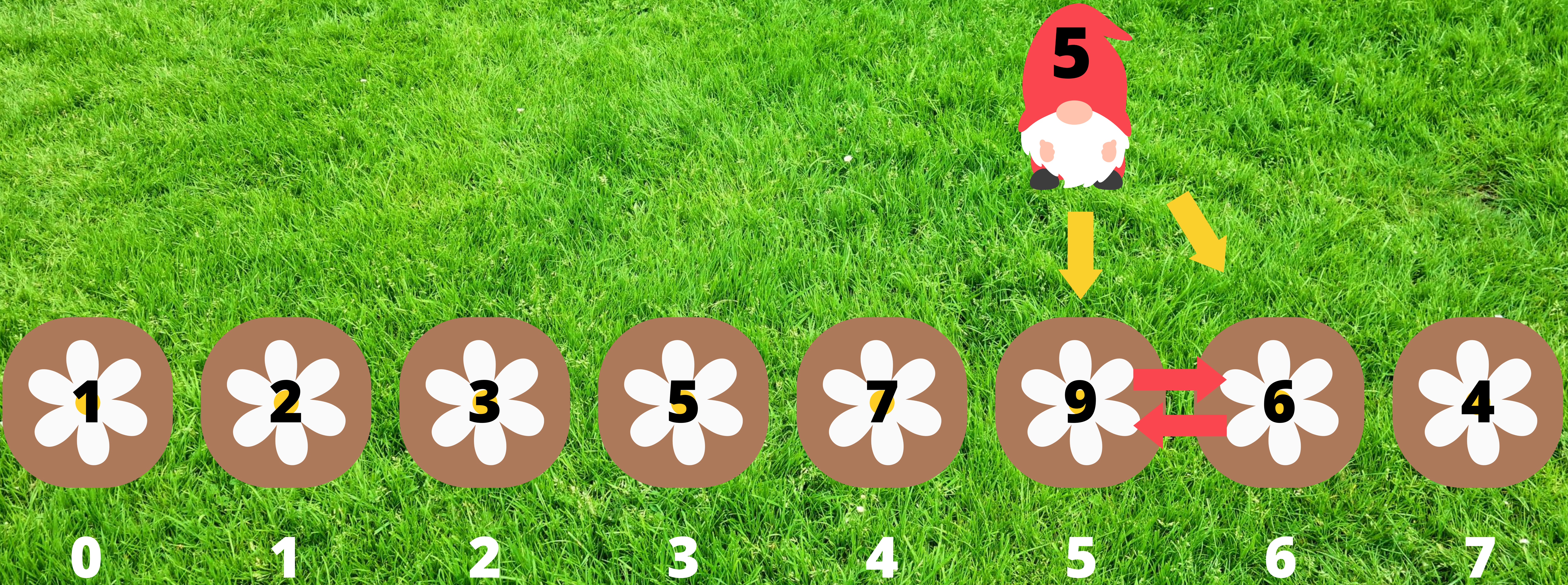
SIMULATION



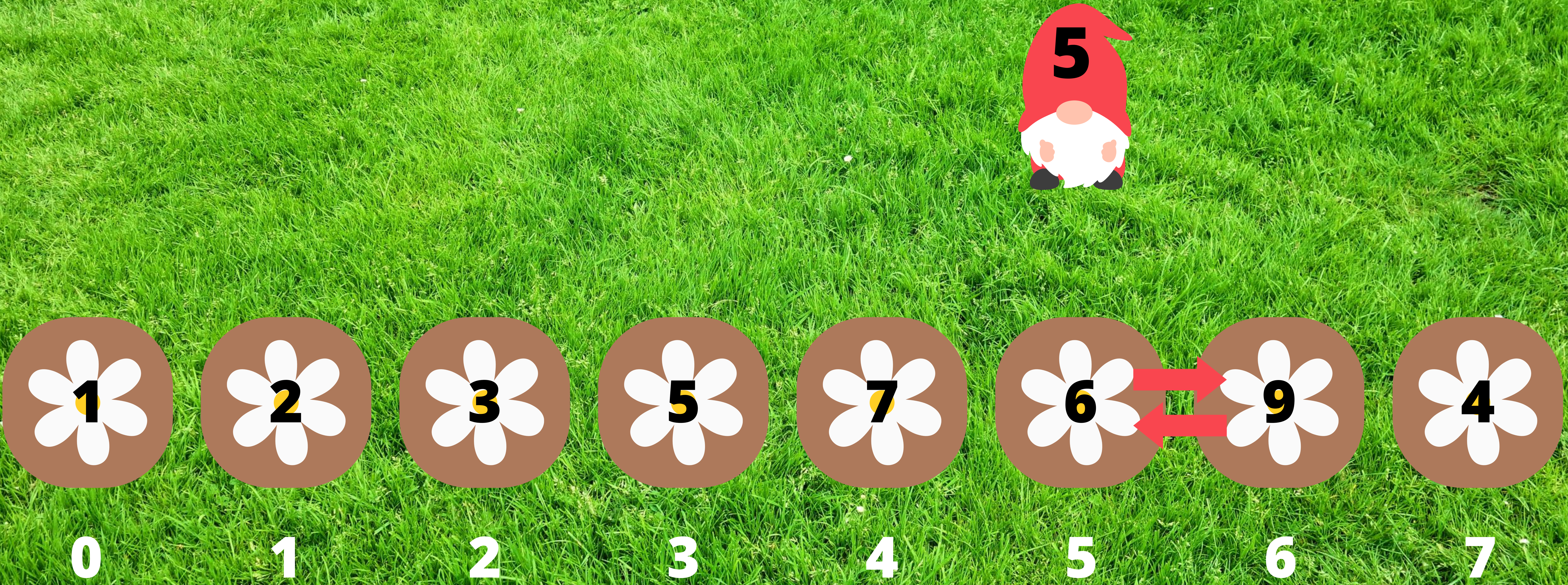
SIMULATION



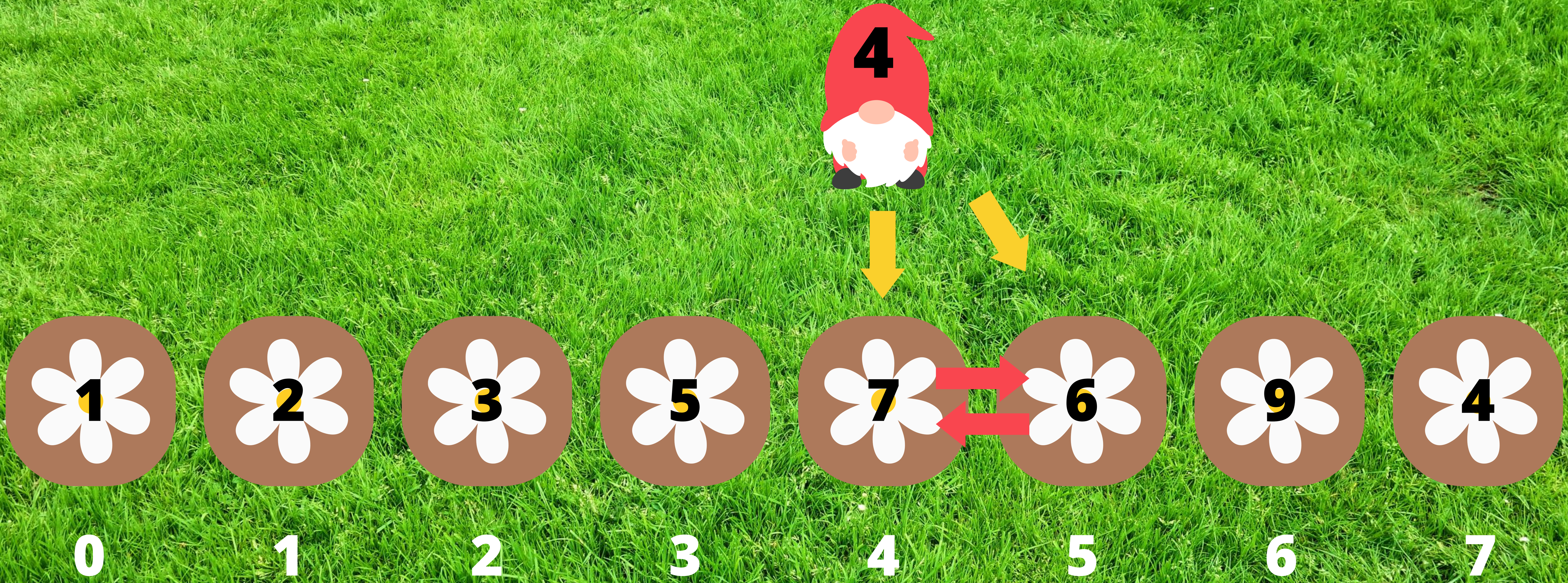
SIMULATION



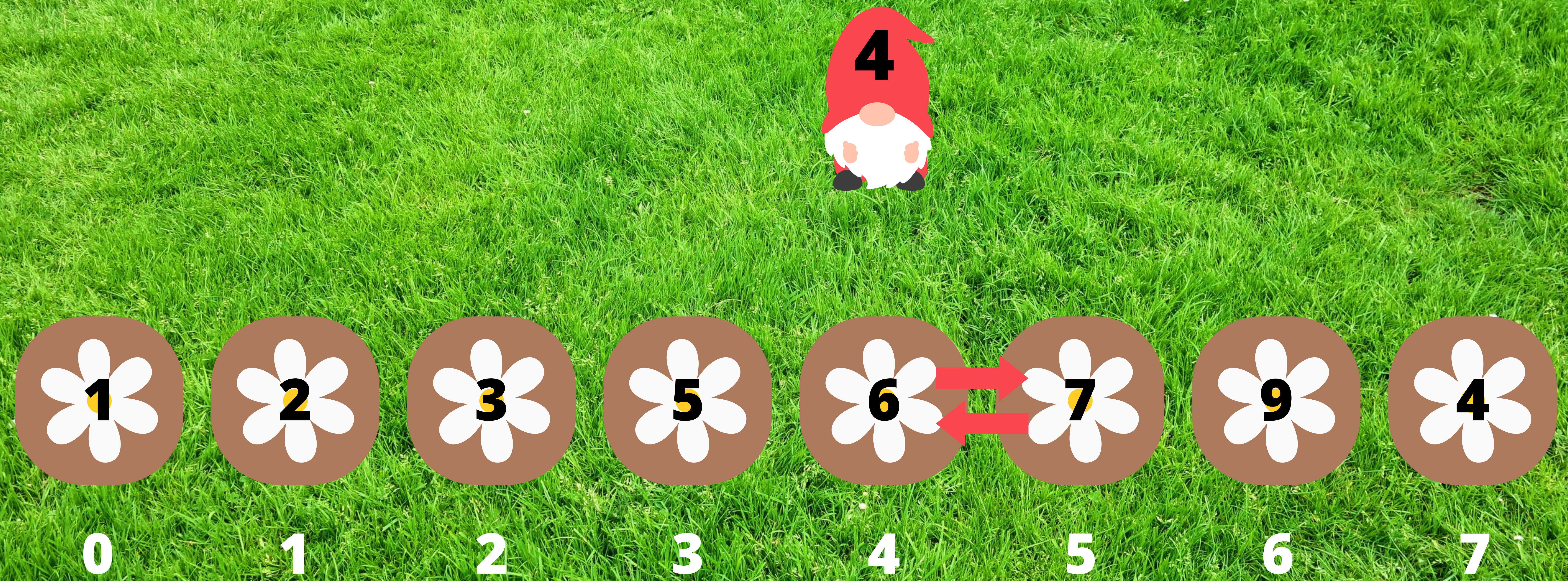
SIMULATION



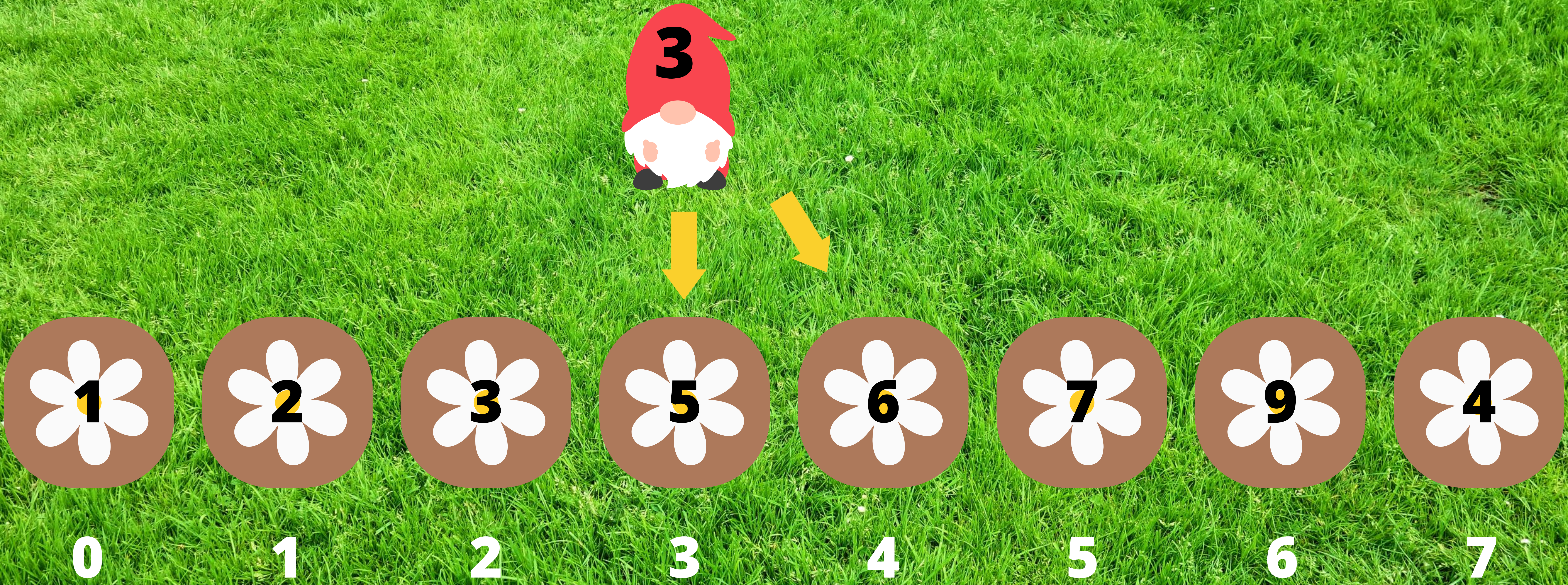
SIMULATION



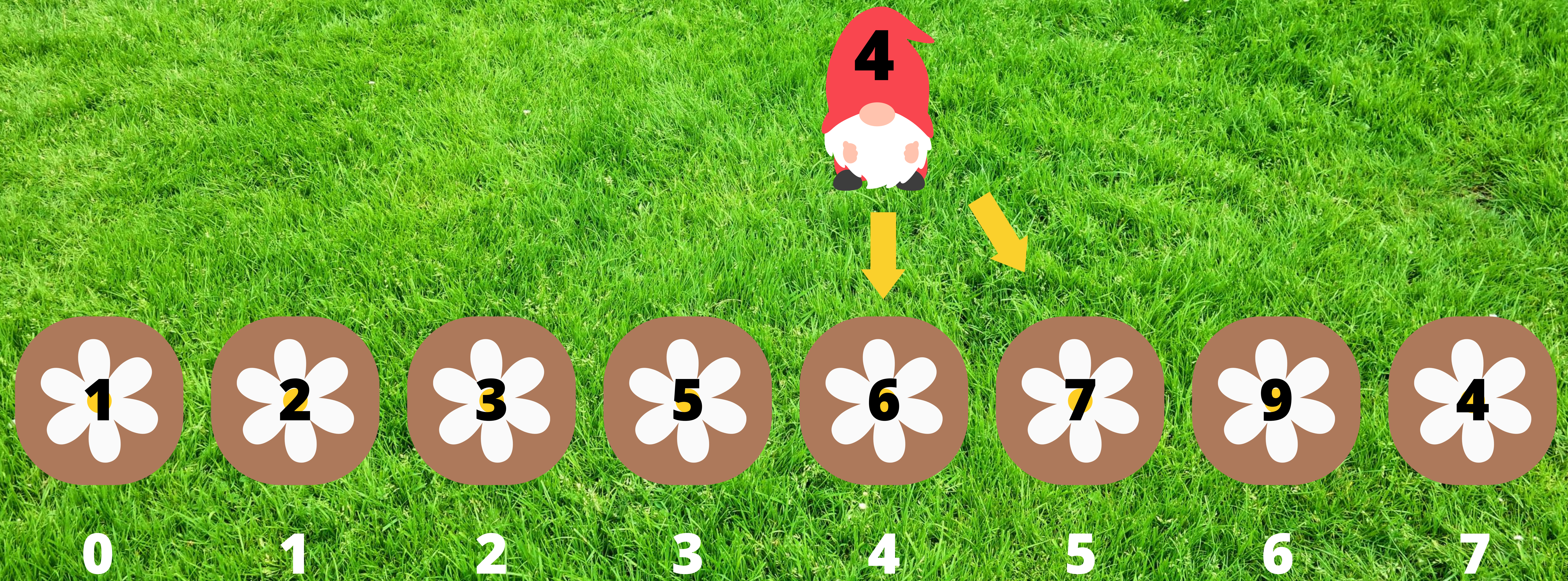
SIMULATION



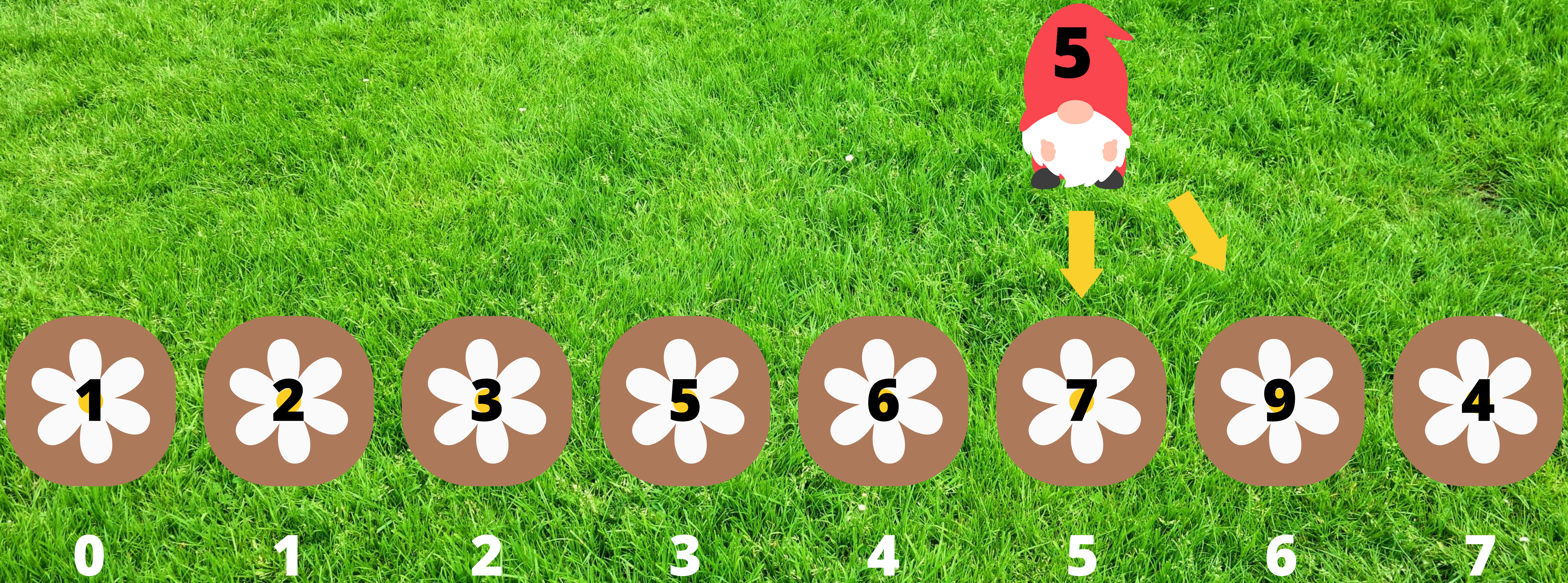
SIMULATION



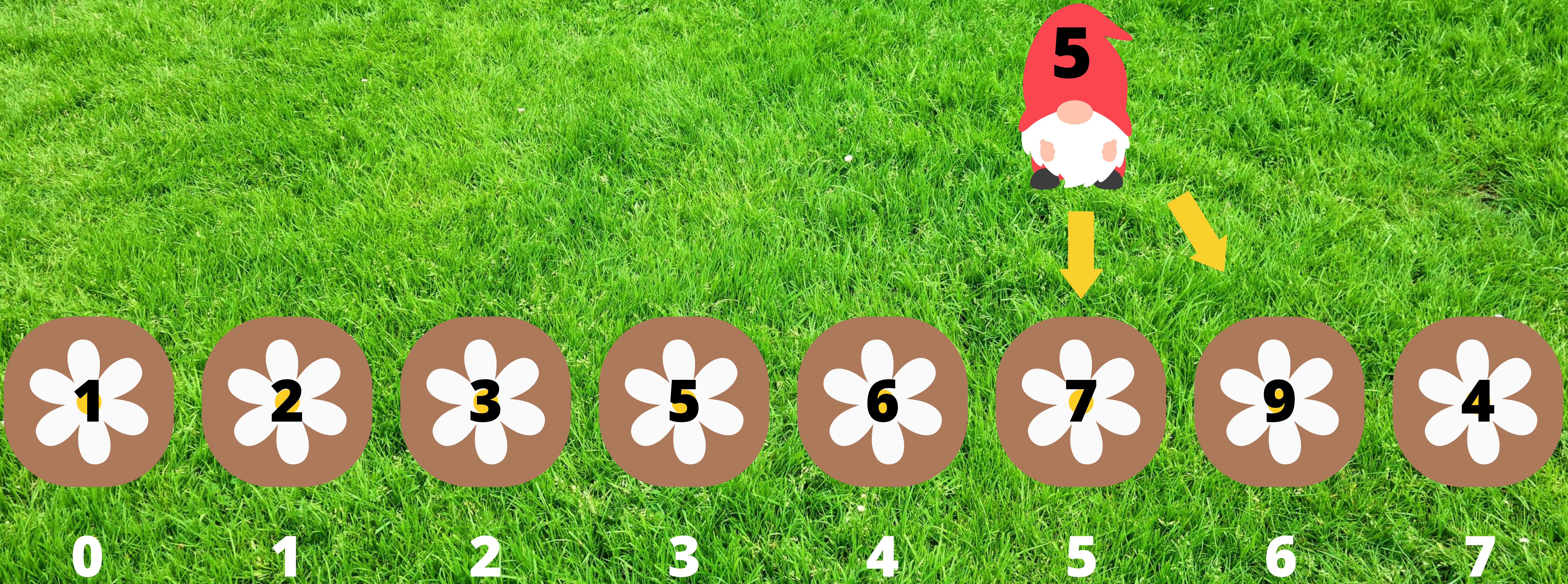
SIMULATION



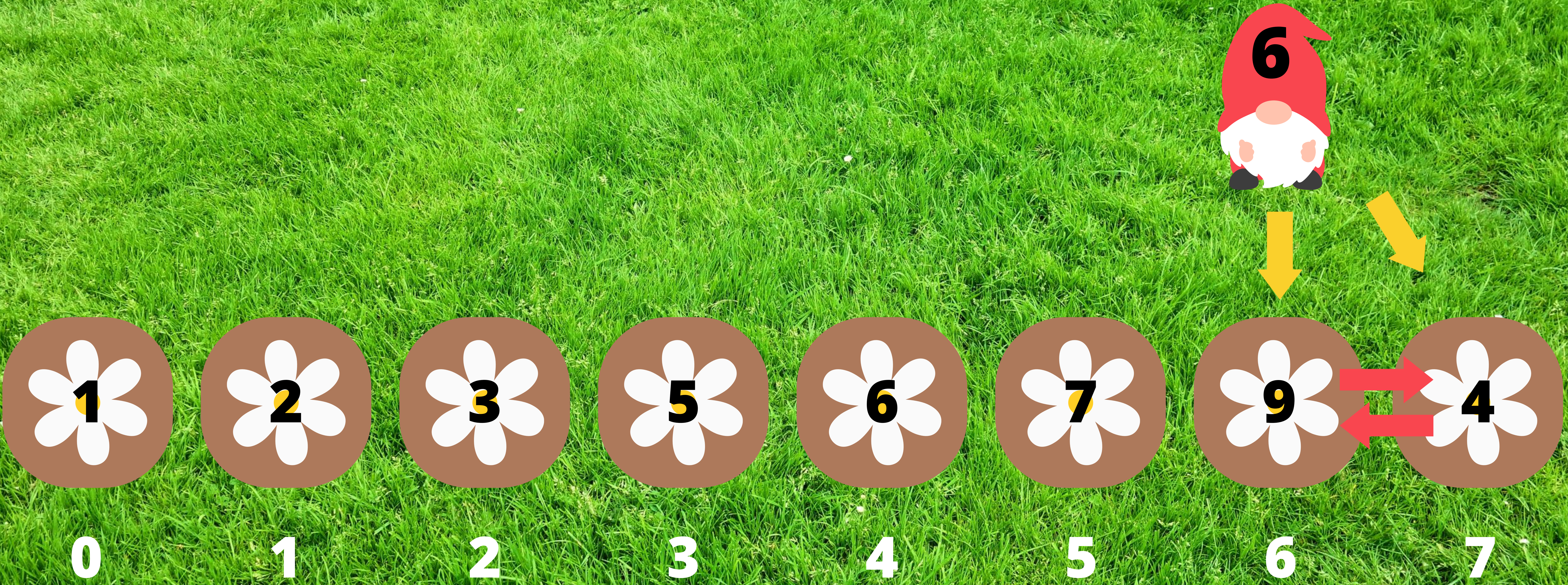
SIMULATION



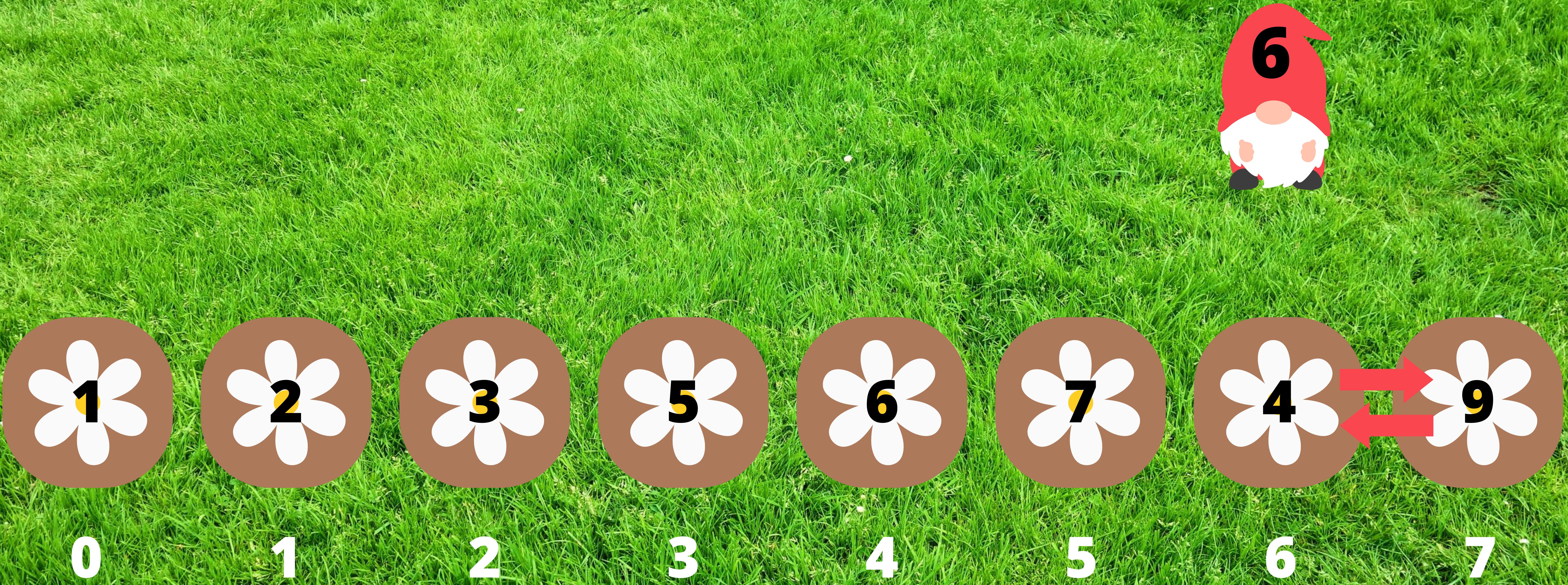
SIMULATION



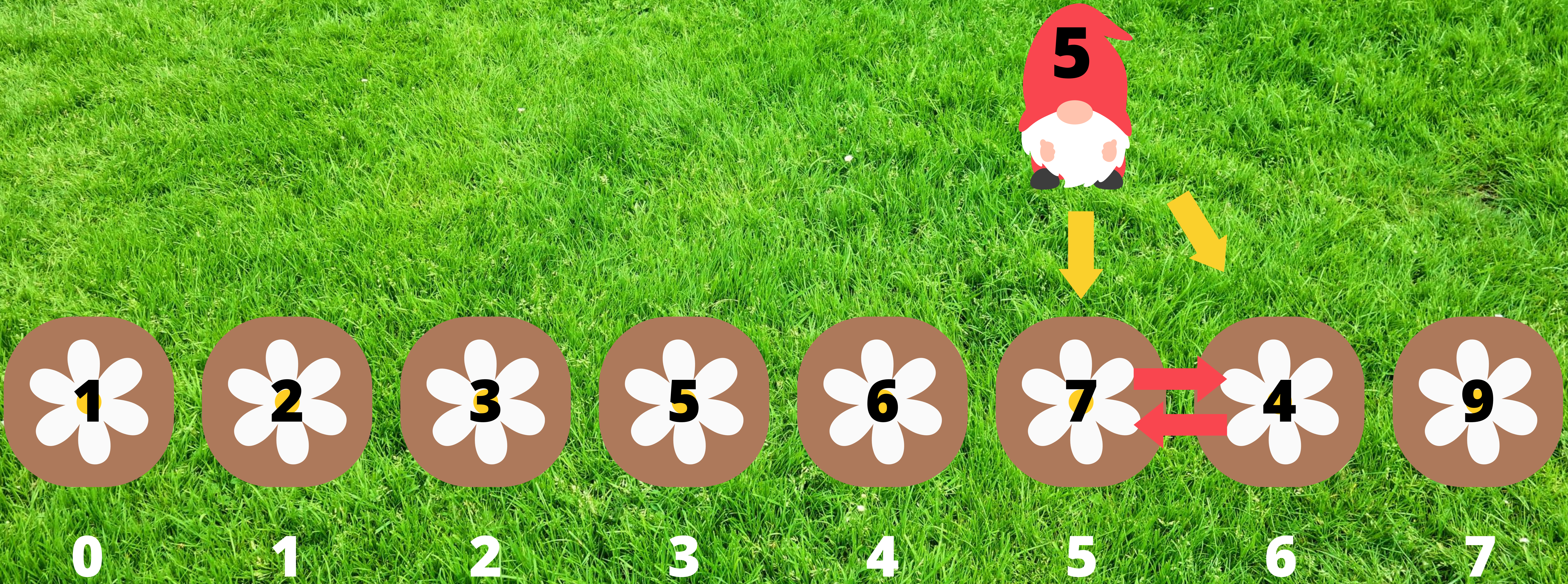
SIMULATION



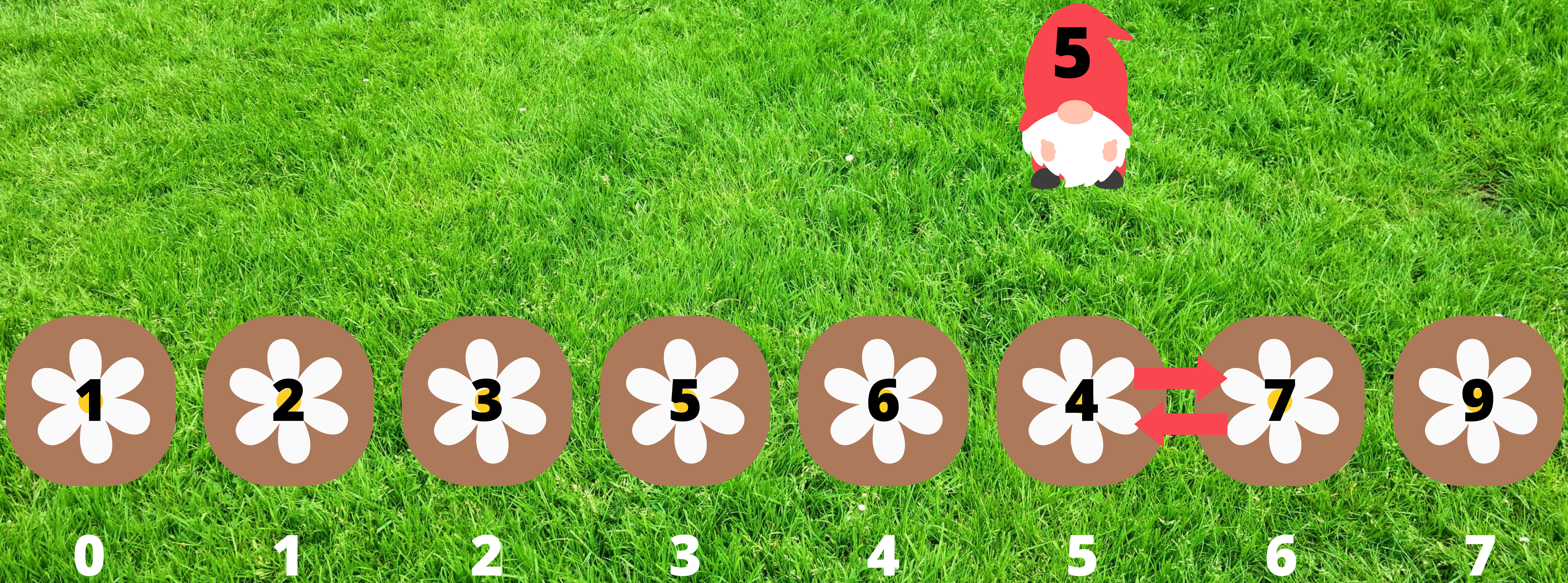
SIMULATION



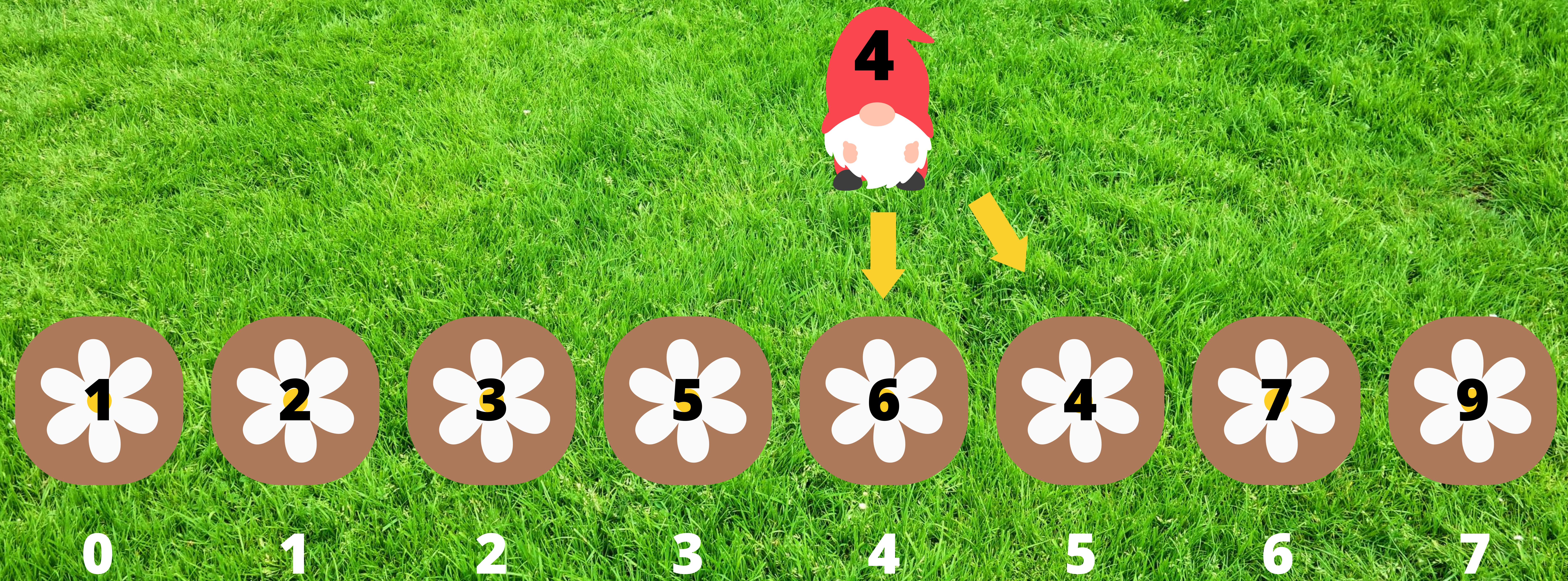
SIMULATION



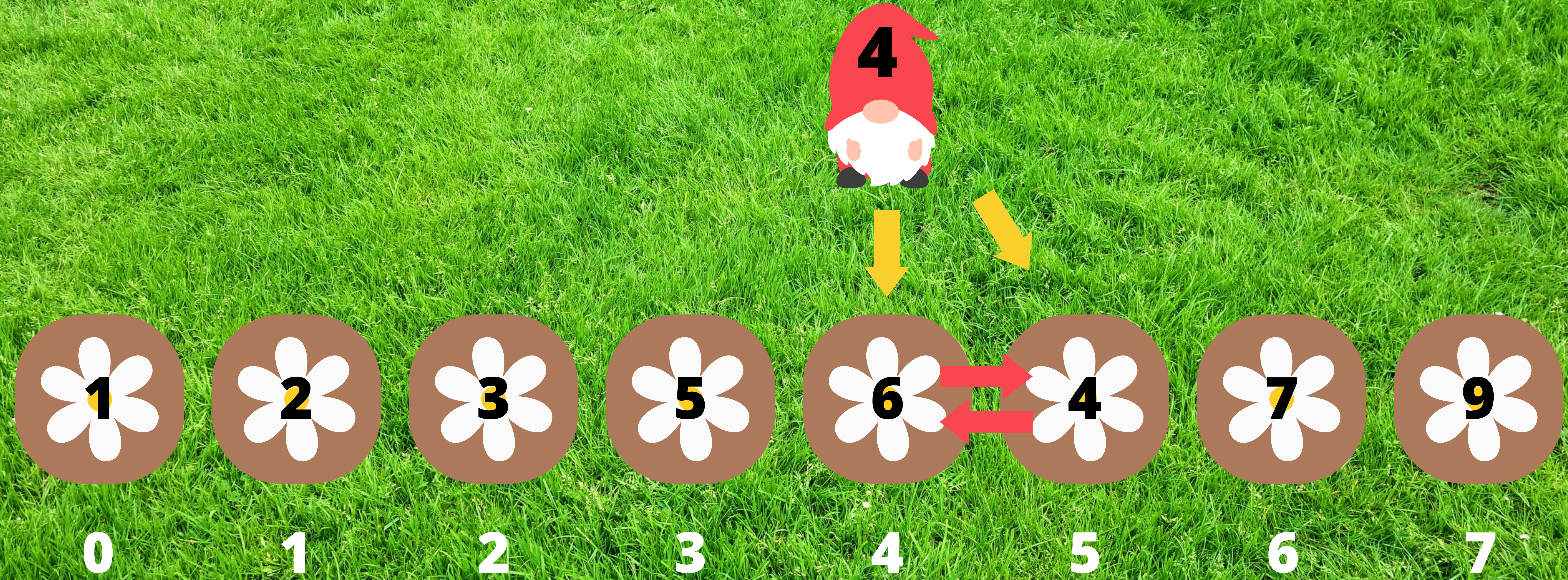
SIMULATION



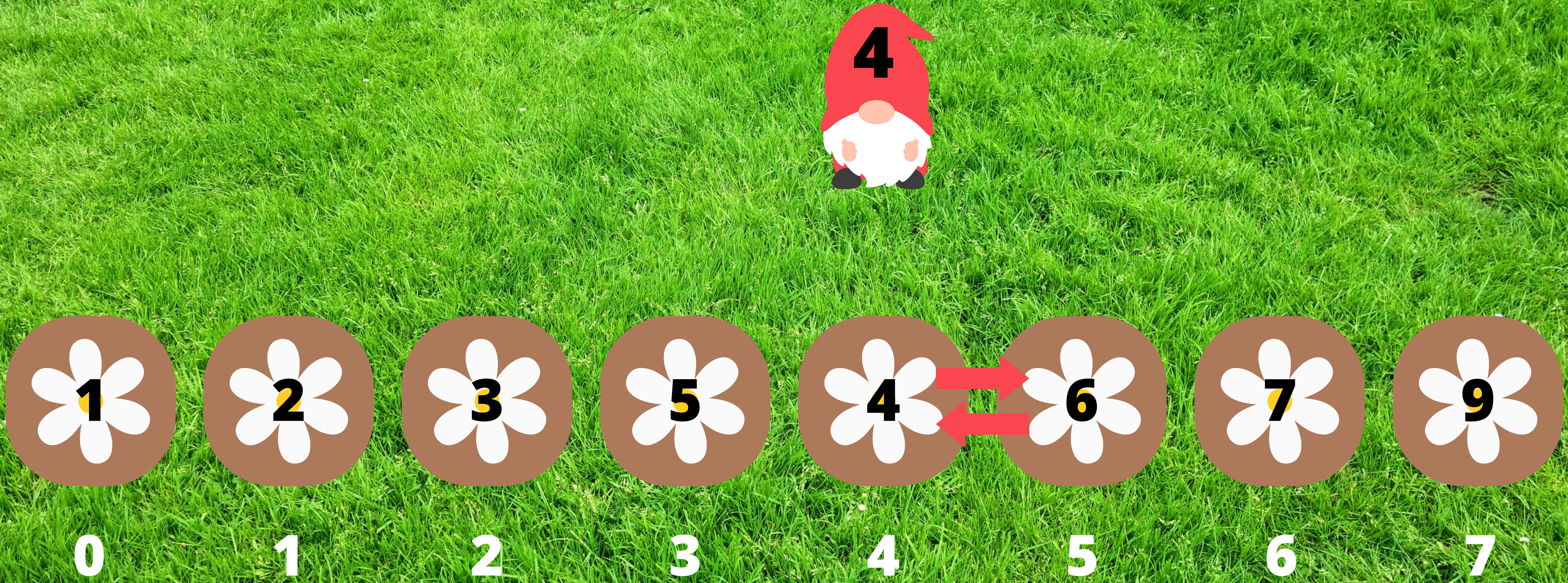
SIMULATION



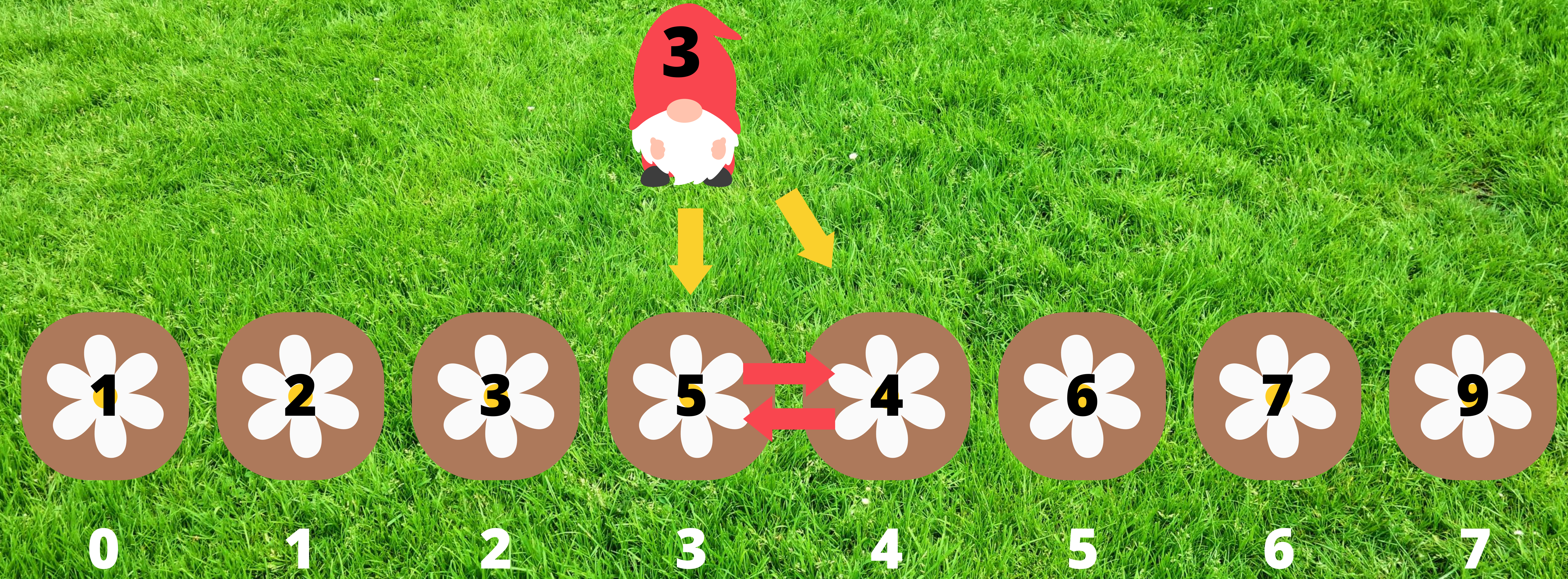
SIMULATION



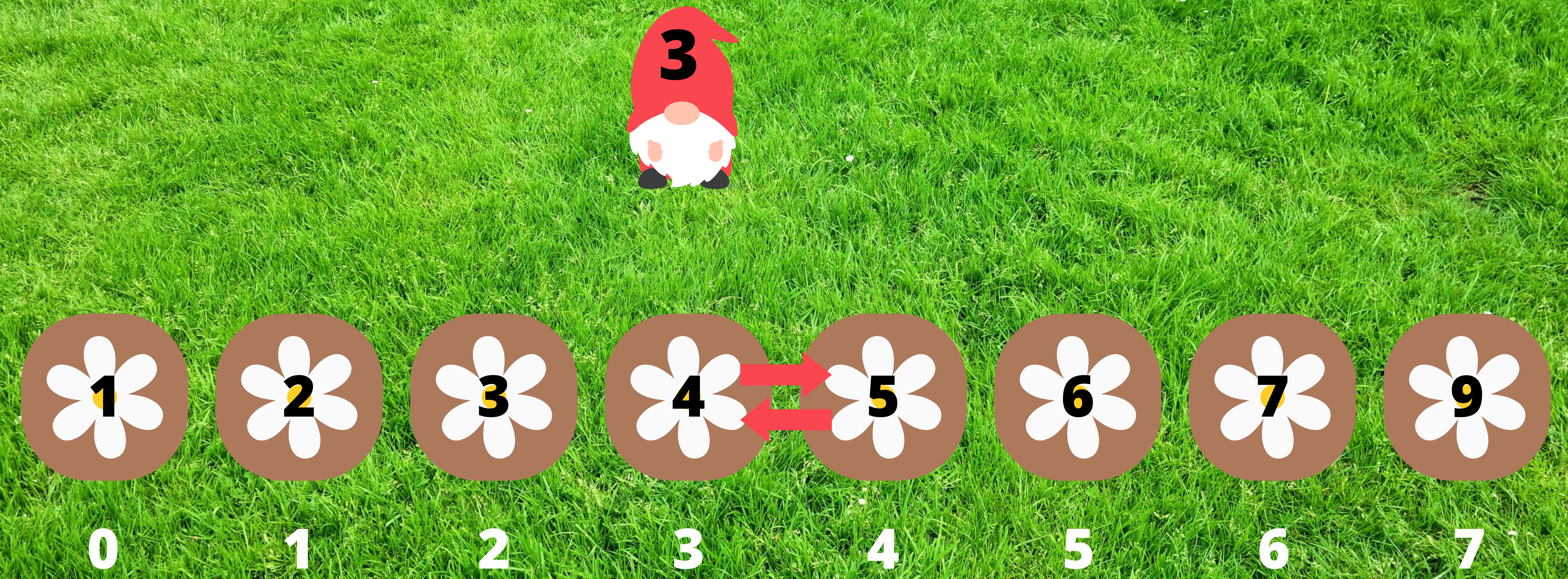
SIMULATION



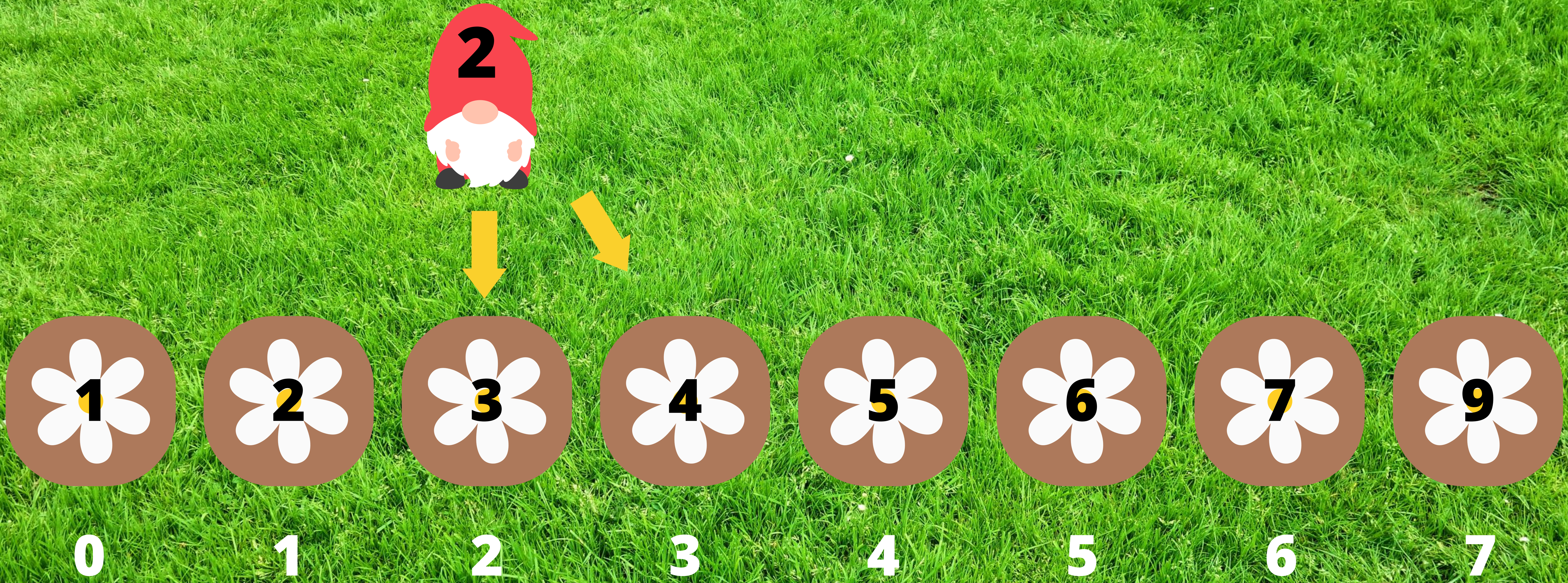
SIMULATION



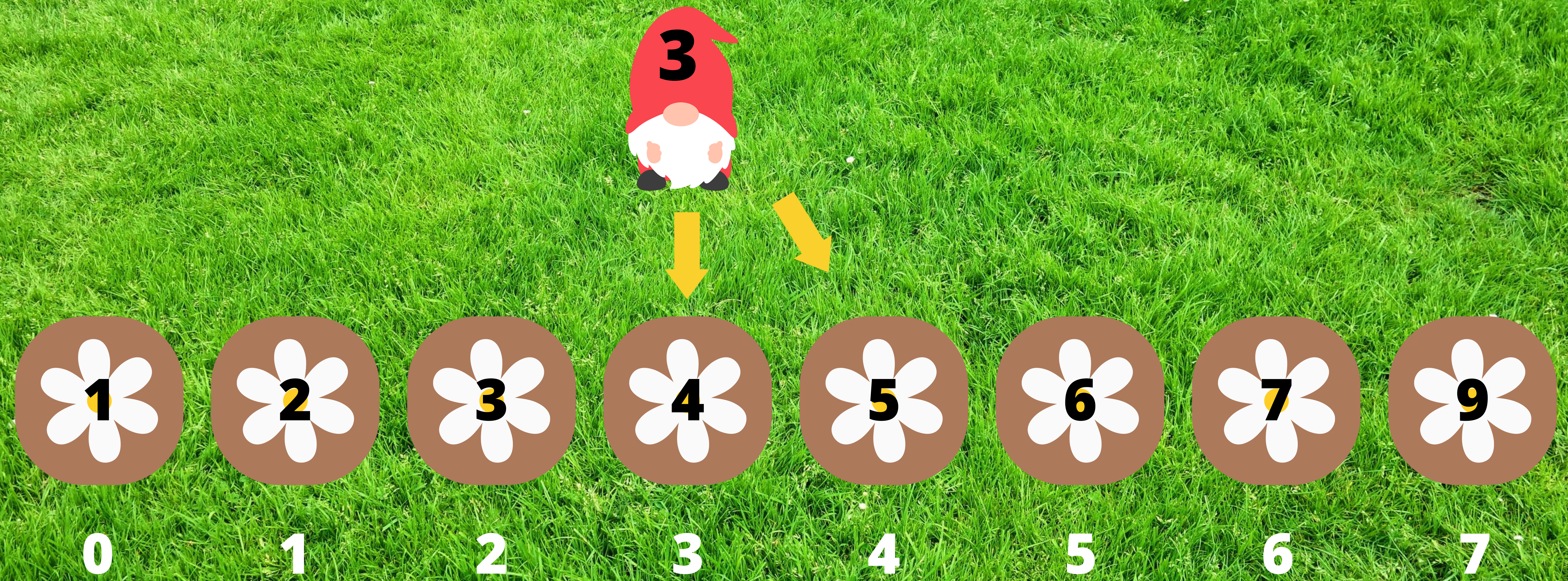
SIMULATION



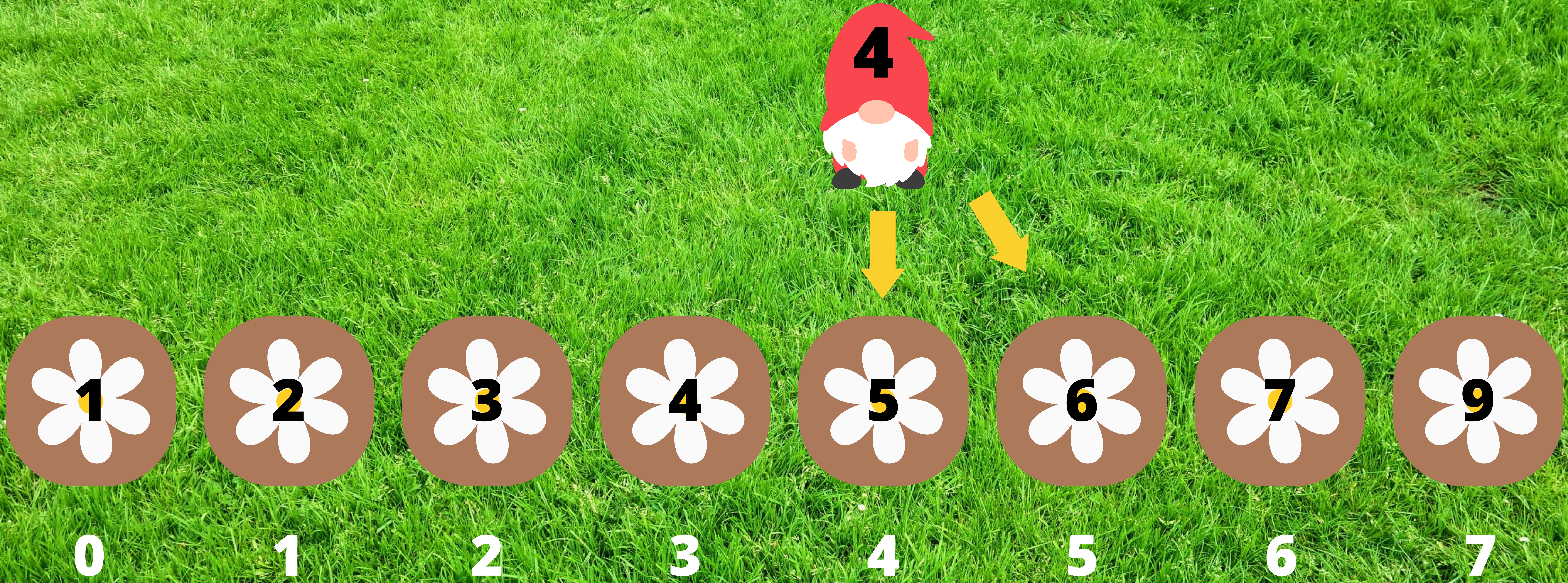
SIMULATION



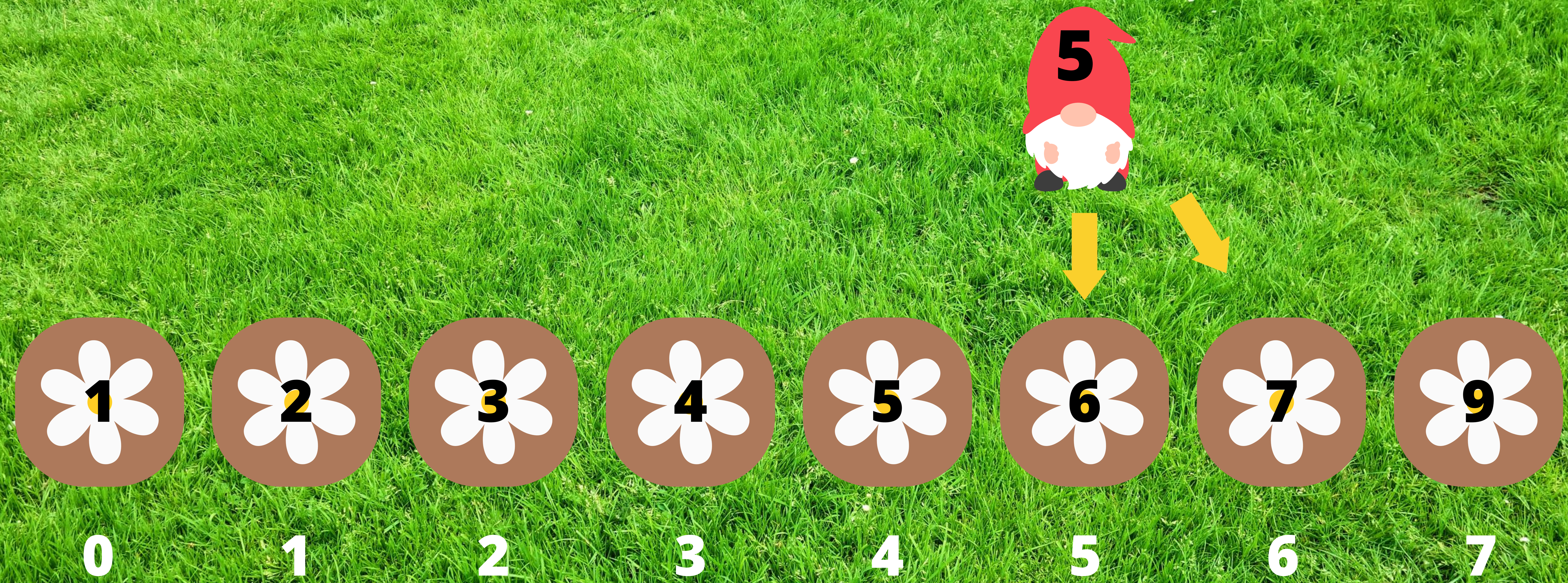
SIMULATION



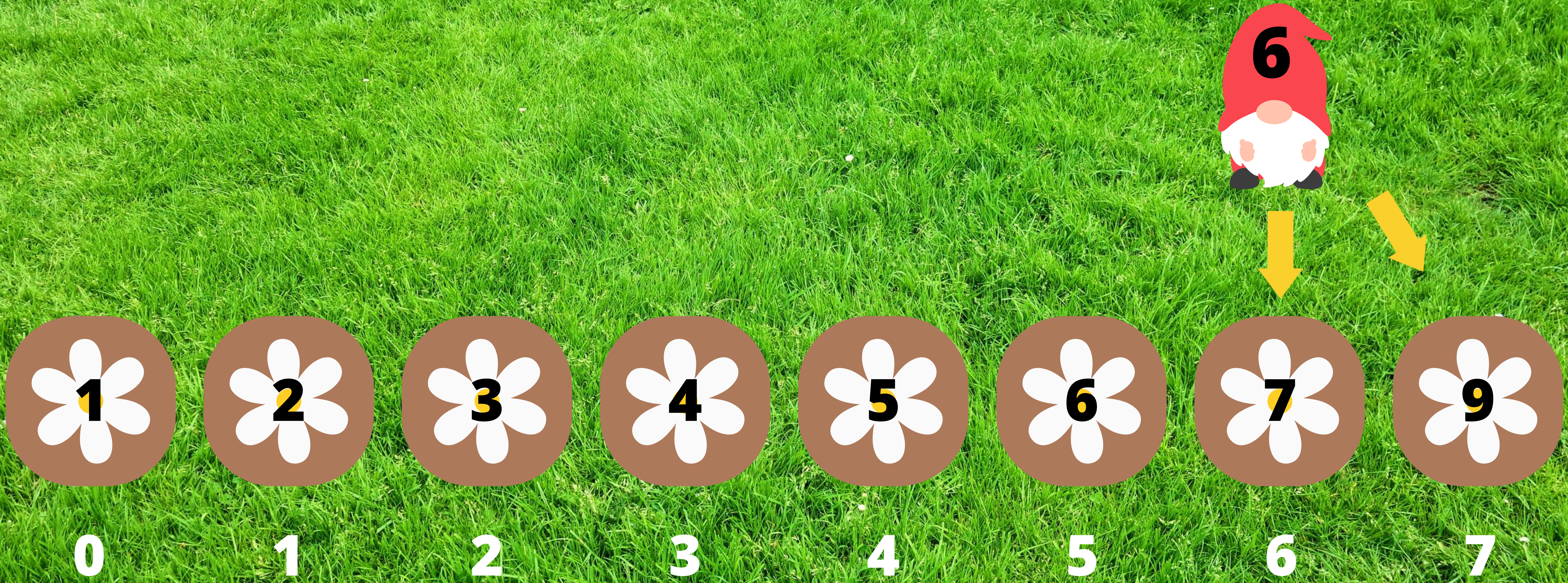
SIMULATION



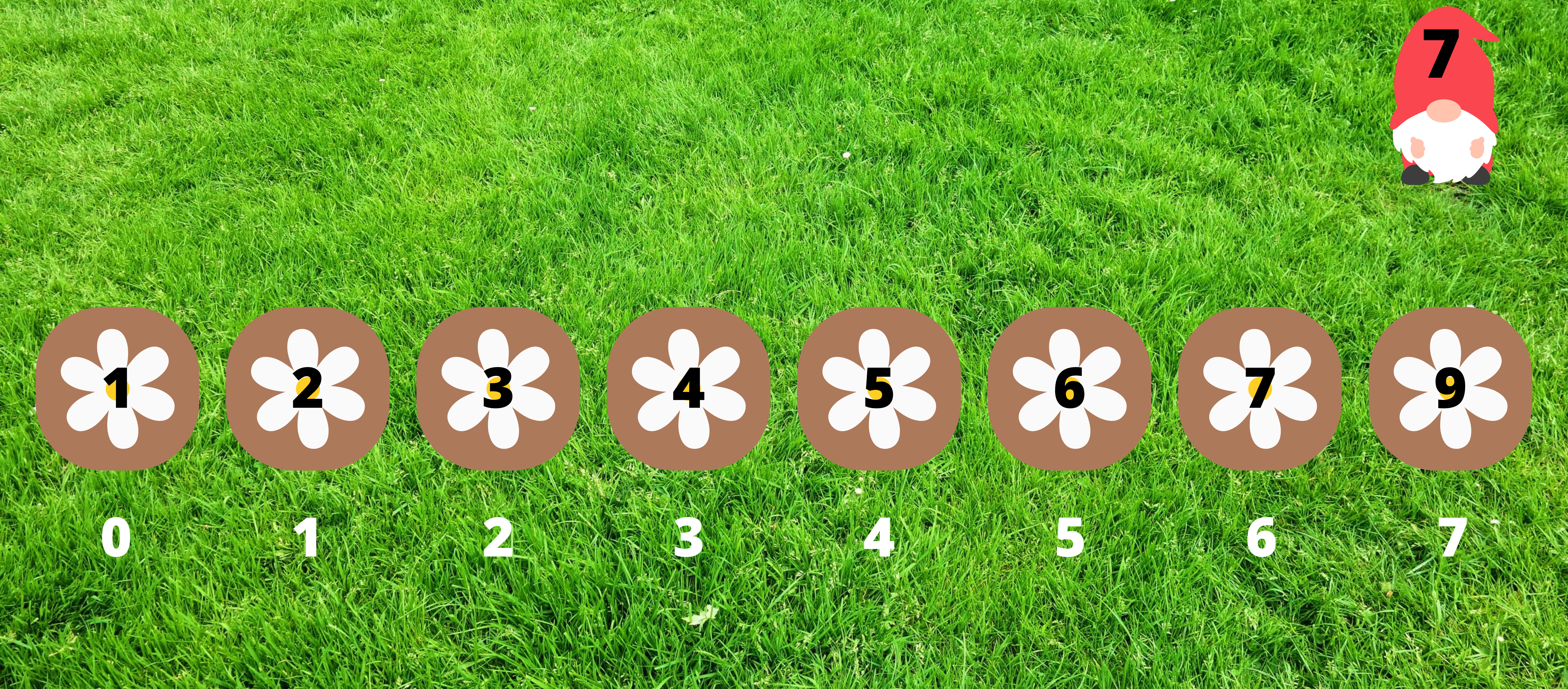
SIMULATION



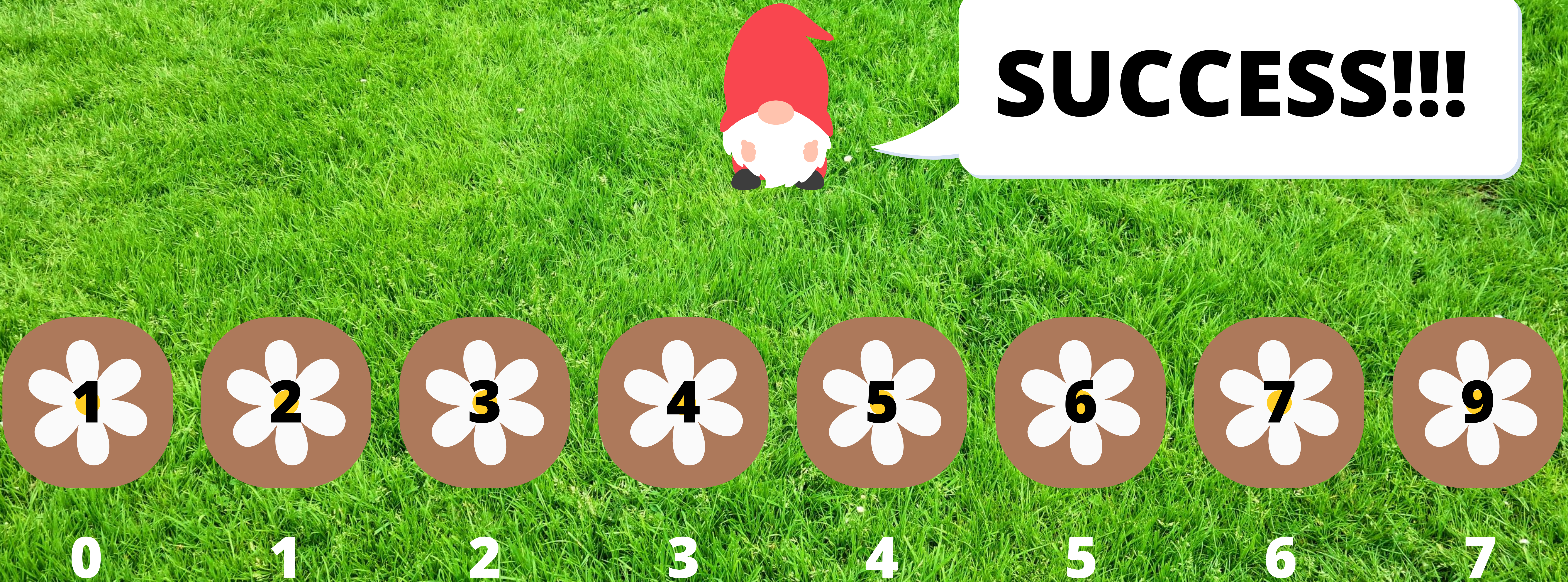
SIMULATION



SIMULATION



SIMULATION

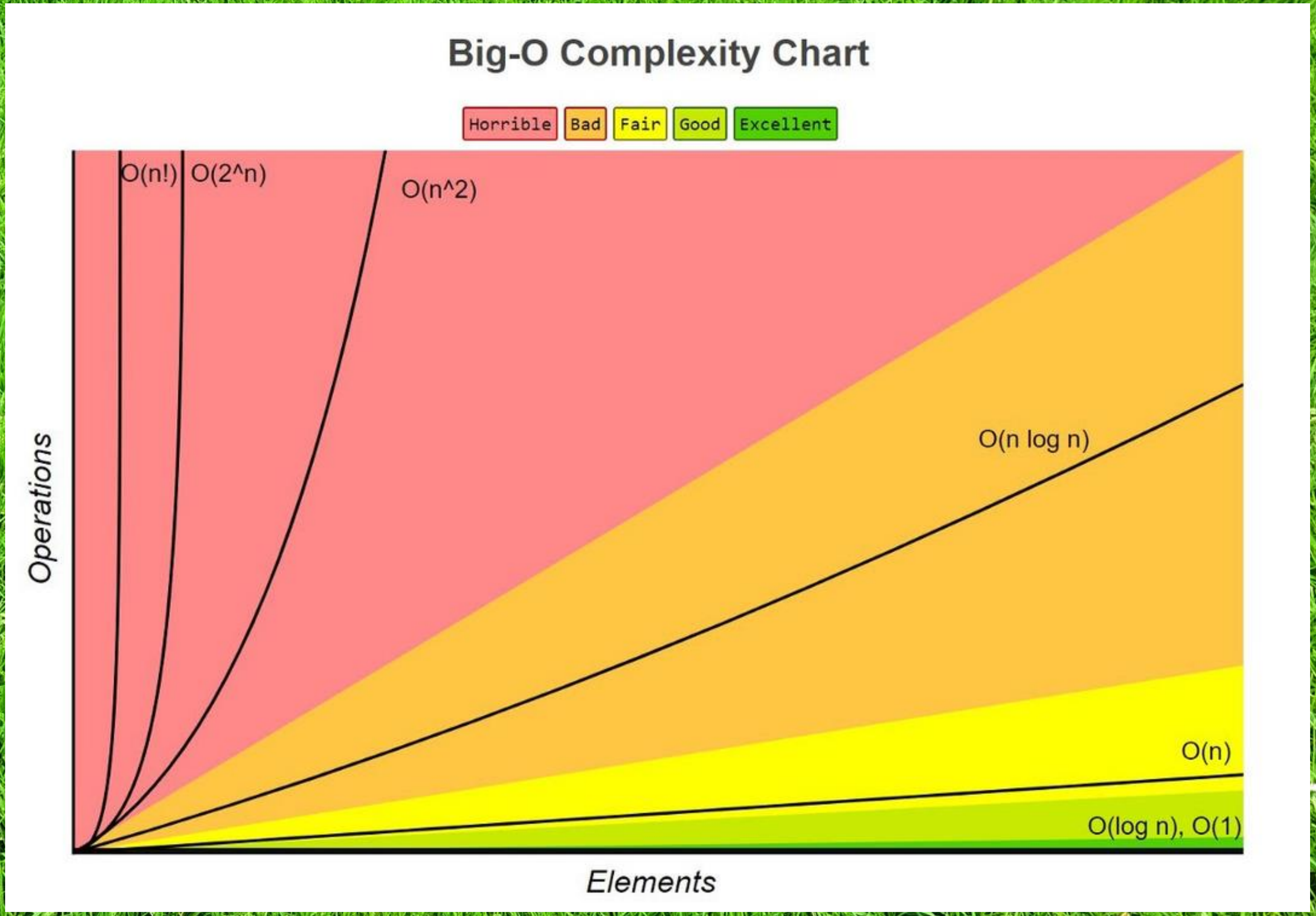


TIME AND SPACE COMPLEXITY

Time Complexity

- **Best Case: 0(N)**
- **Average Case: 0(N2)**
- **Worst Case: 0(N2)**

Space Complexity: 0(1)



CODE COMPARISON

Internet Code

```
int i, temp, ar[10], n;
printf("\nEnter the elements number:");
scanf("%d", &n);
printf("\nEnter elements:\n");
for (i = 0; i < n; i++){
  scanf("%d", &ar[i]);
}
i = 0;
while (i < n){
 if (i == 0 || ar[i - 1] <= ar[i]){
    i++;
 }else{
    temp = ar[i-1];
    ar[i - 1] = ar[i];
    ar[i] = temp;
    i = i - 1;
  }
}
```

Group's Code

```
int arr[] = {3,5,9,1,7,4,6,2}
int size = sizeof(arr) / sizeof(int);
int curPos = 0;
while(curPos < n){
   if(curPos==0 || ar[curPos-1] <= ar[curPos] ){
      curPos++;
   }else{
      swapElem(ar[curPos-1], ar[curPos]);
      curPos = curPos - 1;
   }
}

void swapElem(int *elem1, int *elem2){
      int temp;
      temp = *elem1;
      *elem1 = *elem2;
      *elem2 = temp;
}
```

Team Stupid

THANK YOU!