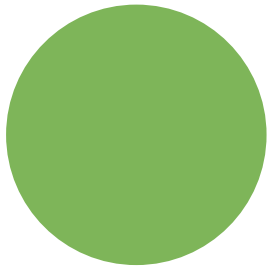
# Módulo 8: Recursión
# Parte 2
## Capítulo 11

https://jolson615.github.io/createasearchalgorithm/index.html

# ¿En cuántos clicks puedes resolver el siguiente problema?



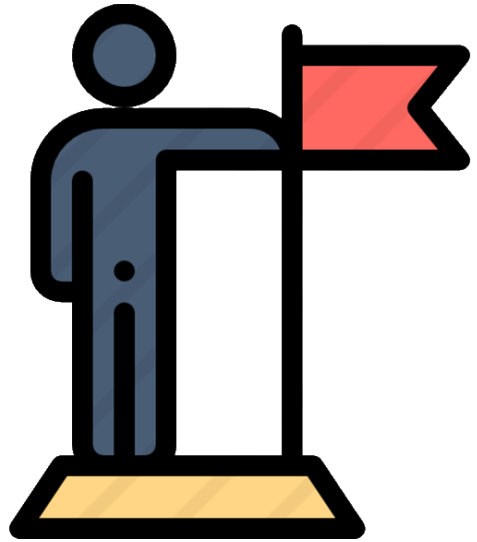
# Ejercicio

- ¿Qué patrón encontraron en los números generados para el ejercicio?
- ¿En cuántos clicks puedes resolver el problema?
- ¿En cuántos clicks puedes consistentemente resolver el problema?
- ¿En cuántos clicks resuelves el ejercicio utilizando búsqueda secuencial?

# Búsqueda Binaria

La búsqueda secuencial es un algoritmo que nos ayuda a encontrar elementos en una lista ordenada.

Es un algoritmo recursivo que utiliza una patrón “Divide and Conquer”.



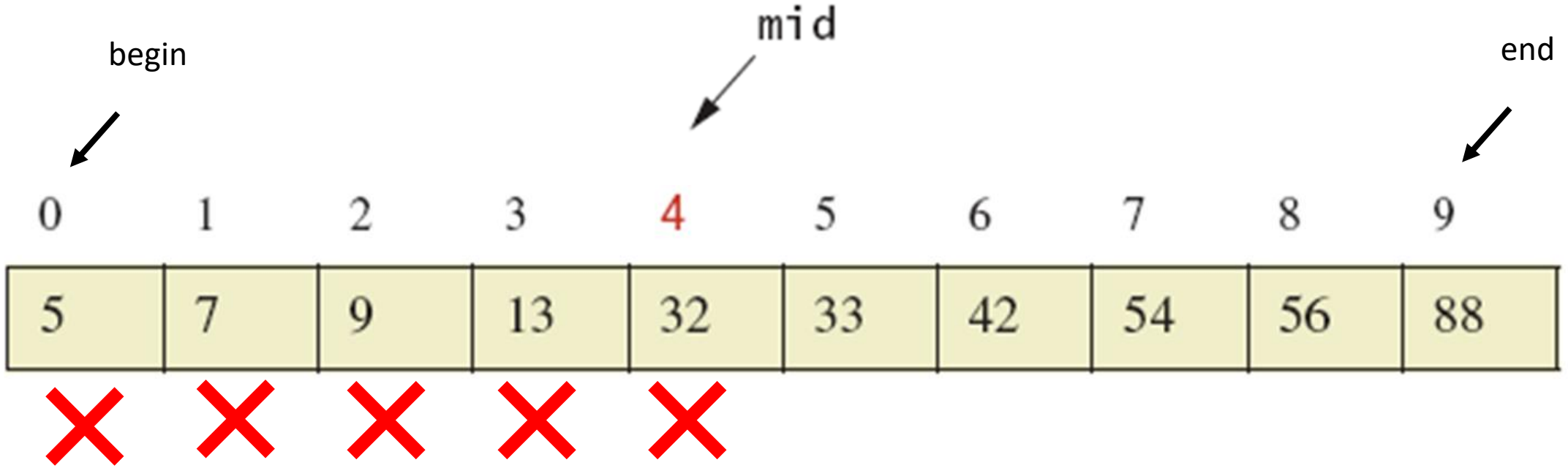
# Búsqueda Binaria

1. `mid` = midpoint between `first` y `last`.
2. `if (first > last)`
3. `  return -1`
4. `else if (target == a[mid])`
5. `  return mid`
6. `else if (target < a[mid])`
7. `  return` result of searching `a[first]` through `a[mid-1]`
8. `else if (target > a[mid])`
9. `  return` result of searching `a[mid+1]` through `a[last]`

# Búsqueda Binaria

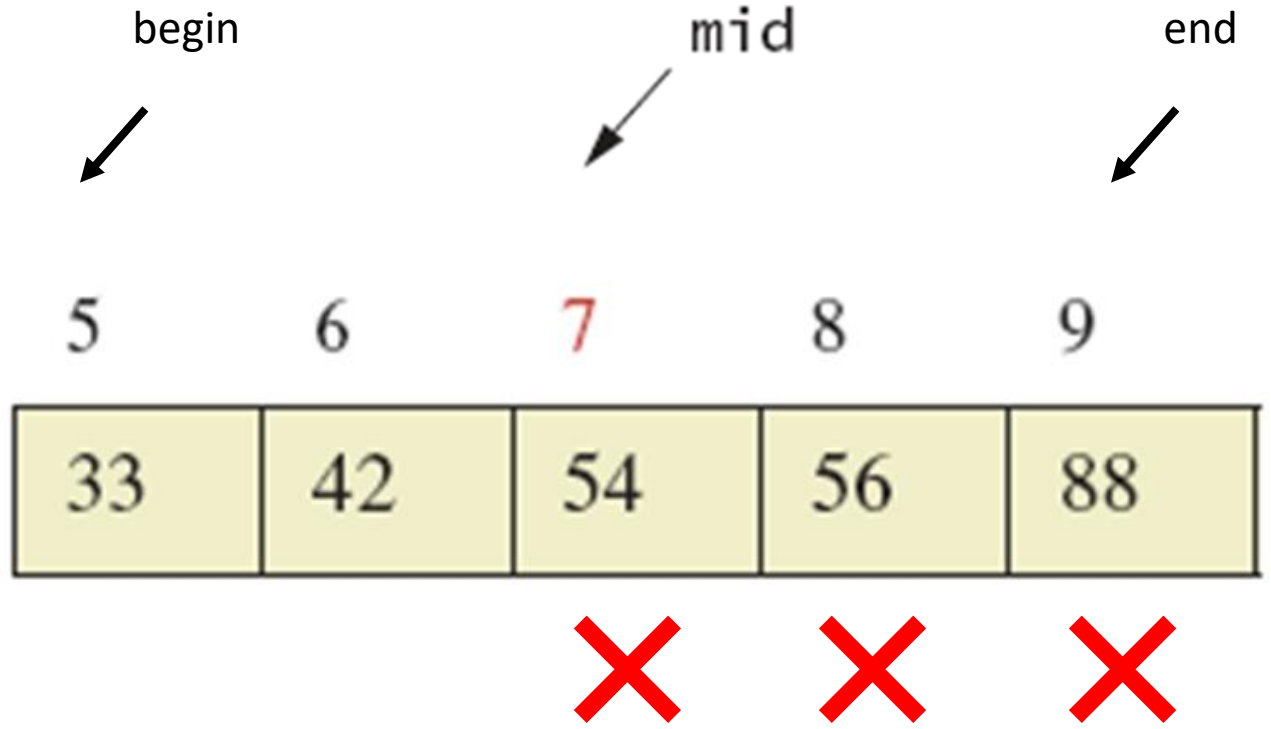
target = 33

1. Calulamos `mid` entre 0 y 9.
   `mid = (0 + 9 ) / 2`. (División entera)
2. `target` es mayor que `a[mid]`, por lo que podemos asegurar que el número sólo puede encontrarse desde `a[mid+1]` hasta `a[end]`.



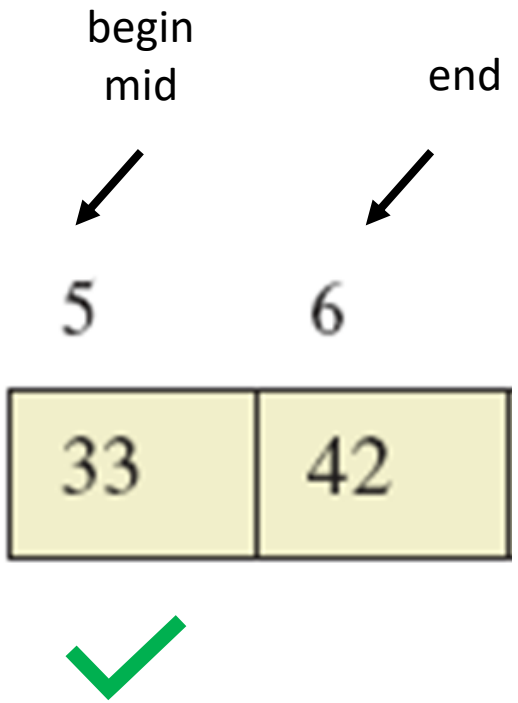
target = 33

1. Calulamos `mid` entre 5 y 9.
   `mid = (5 + 9 ) / 2`. (División entera)
2. `target` es menor que `a[mid]`, por lo que podemos asegurar que el número sólo puede encontrarse desde `a[begin]` hasta `a[mid-1]`.



target = 33

1. Calulamos `mid` entre 5 y 6.
   `mid = (5 + 6 ) / 2`. (División entera)
2. `target` es igual a `a[mid]`, por lo que el resultado es retornar el índice 5.



# Binary Search

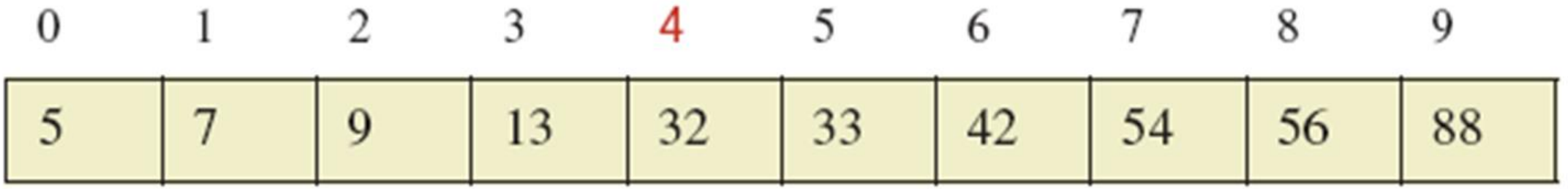
```
private static int binarySearch(int[] array, int target, int begin, int end) {
  int mid = (begin + end) / 2;

 //Base case #1: target not found
  if (begin > end)
    return -1;
 //Base case #2: target found
  if (array[mid] == target)
    return mid;
 //Recursive case #1: target is on left side of array
  if (array[mid] > target) {
    return binarySearch(array, target, begin, mid-1);
  } else {
 //Recursive case #2: target is on right side of array
    return binarySearch(array, target, mid+1, end);
  }
}
```

# Binary Search

```
public static int binarySearch(int[] array, int target) {
  return binarySearch(array, target, 0, array.length-1);
}
```

Sobrecargamos el método `binarySearch` para simplificar la firma al utilizar el método desde fuera.

# Ejercicio




1. Realiza una búsqueda binaria sobre el número 5
2. Realiza una búsqueda binaria sobre el número 100.