Рекурсивна версія бінарного пошуку

Двійковий (бінарний) пошук

Двійко́вий по́шук — алгоритм знаходження заданого значення у впорядкованому масиві, який полягає у порівнянні серединного елемента масиву з шуканим значенням, і повторенням алгоритму для тієї або іншої половини (див. двійкове дерево пошуку), залежно від результату порівняння.

https://uk.wikipedia.org/wiki/%D0%94%D0%B2%D1%96%D0%B9%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BF%D0%BE%D1%88%D1%83%D0%BA

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число searchElement = **27***

Спочатку діапазон пошуку – весь масив

- *Початковий номер діапазону пошуку **start = 0***
- *Кінцевий номер діапазону пошуку **end = a.length-1***

start

end

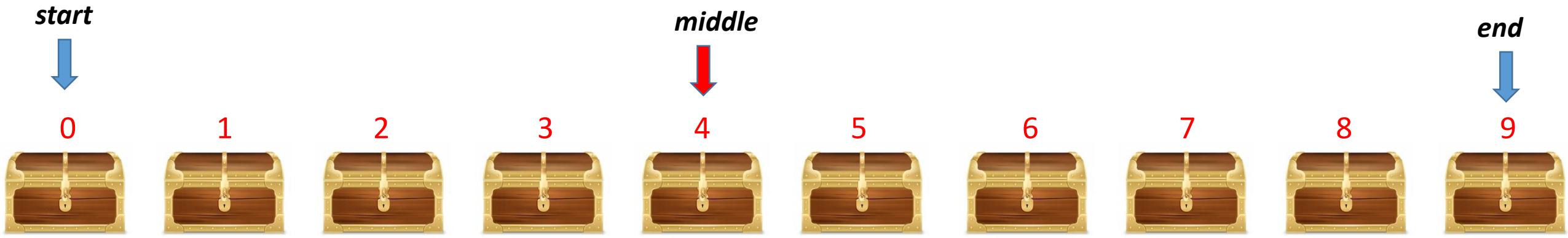

```
function binarySearch(arr, searchElement, start, end) {

}
```

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

***middle** = Math.floor((**start**+**end**)/2)*



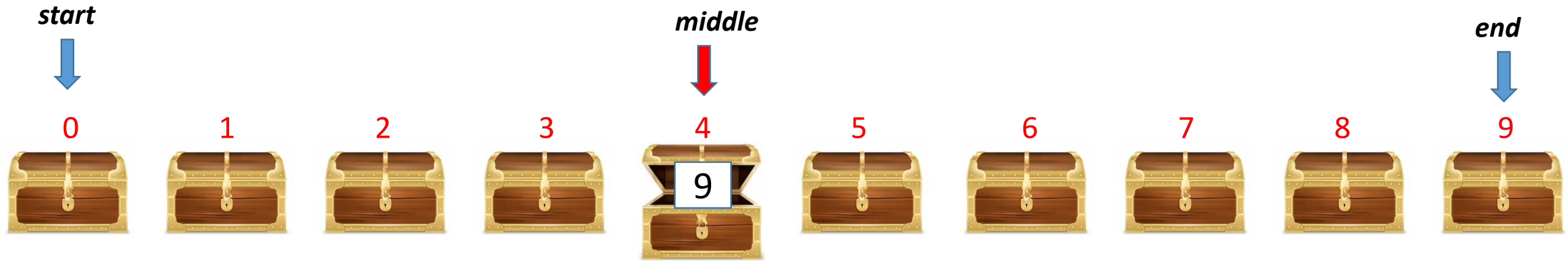
*Знаходимо елемент **middle**, що знаходиться між **start** та **end***

```
function binarySearch(arr, searchElement, start, end) {
  if (start <= end) {
    const middle = Math.floor((start + end) / 2)
    if (arr[middle] === searchElement) return middle
  }
}
```

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor((***start***+***end***)/2)

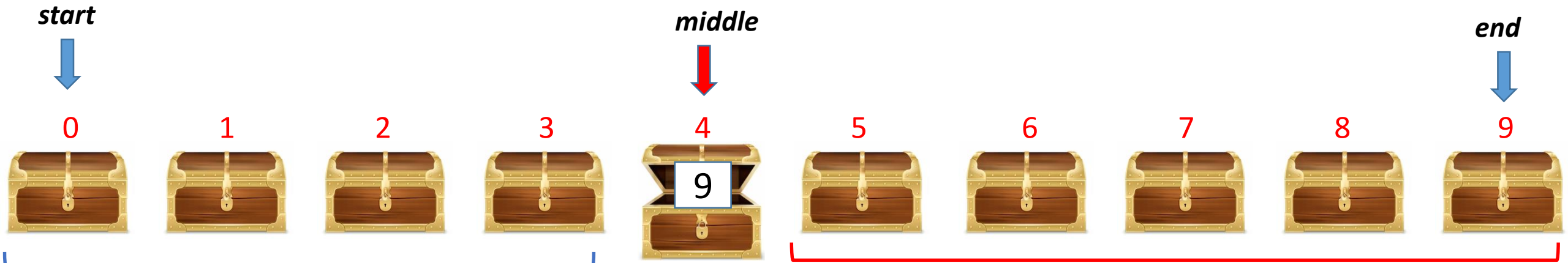


*Порівнюємо елемент, що знаходиться у позиції **middle** з шуканим елментом 27*

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor(***start***+***end***)/2)



Тут усі числа <= 9

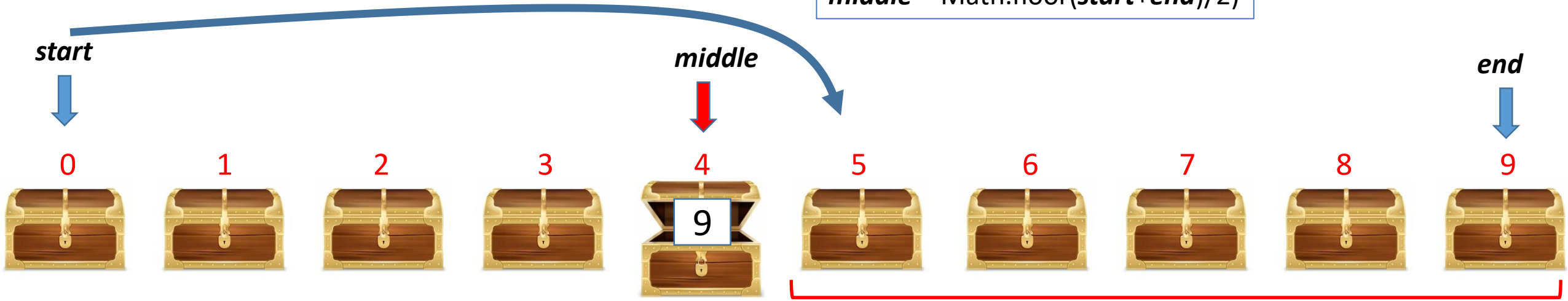
Тут усі числа >= 9

*Число **27 > 9**. Потрібно **<u>шукати справа</u>***

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число searchElemet = **27***

middle = Math.floor(***start***+***end***)/2)



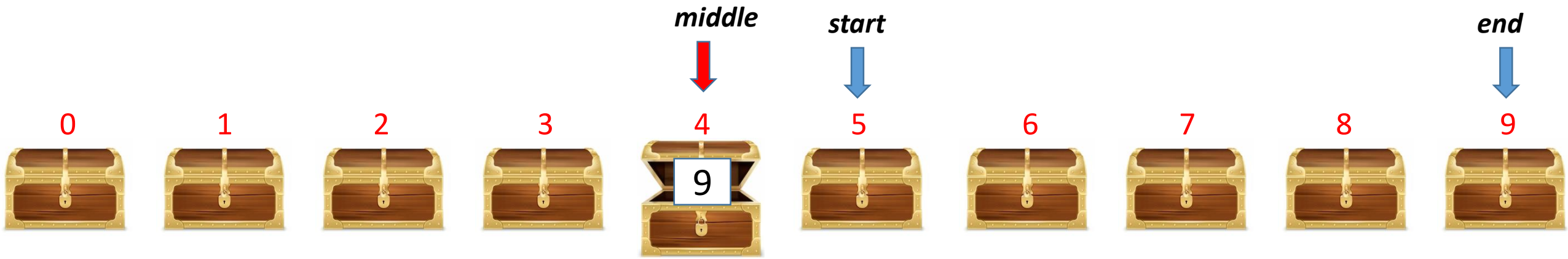
Тут усі числа >= 9

*Число **27 > 9**. Потрібно **<u>шукати справа</u>***
<u>start</u> – зміщуємо вправо від ***<u>middle</u>***
<u>(start = middle + 1)</u>

```
function binarySearch(arr, searchElement, start, end) {
  if (start <= end) {
    const middle = Math.floor((start + end) / 2)
    if (arr[middle] === searchElement) return middle
    if (arr[middle] < searchElement)
      return binarySearch(arr, searchElement, middle + 1, end)
  }
}
```

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

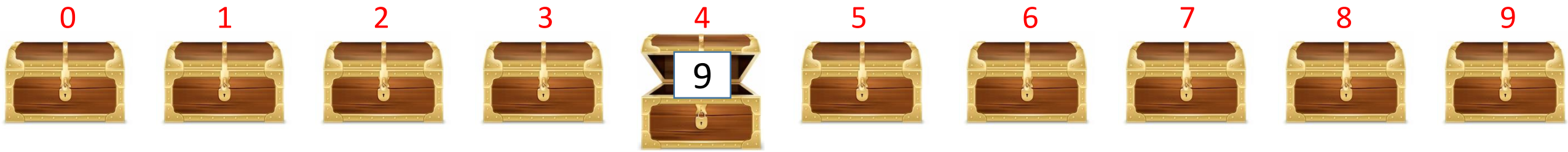


*Число **27 > 9.** Потрібно **<u>шукати справа</u>***
<u>start</u> – зміщуємо вправо від middle

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

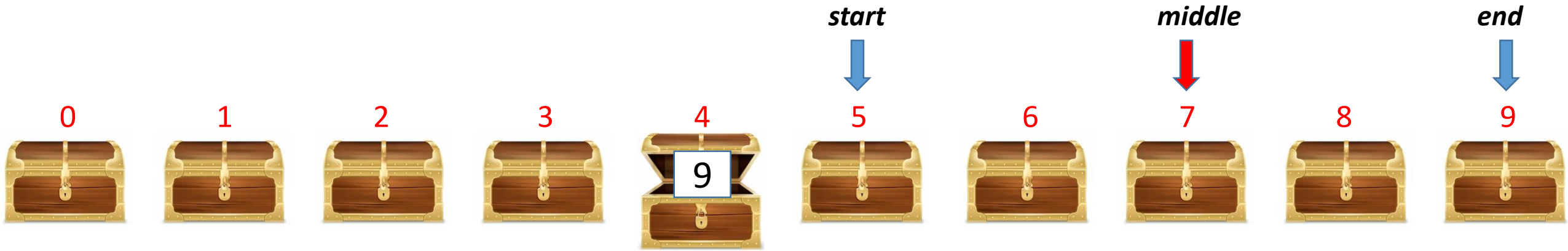
start ***end***



Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor((***start***+***end***)/2)

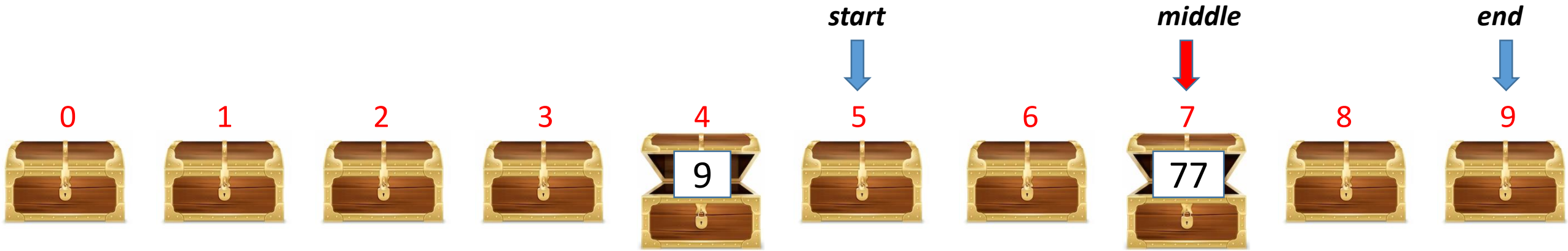


*Знаходимо елемент **middle**, що знаходиться між **start** та **end***

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor((***start***+***end***)/2)

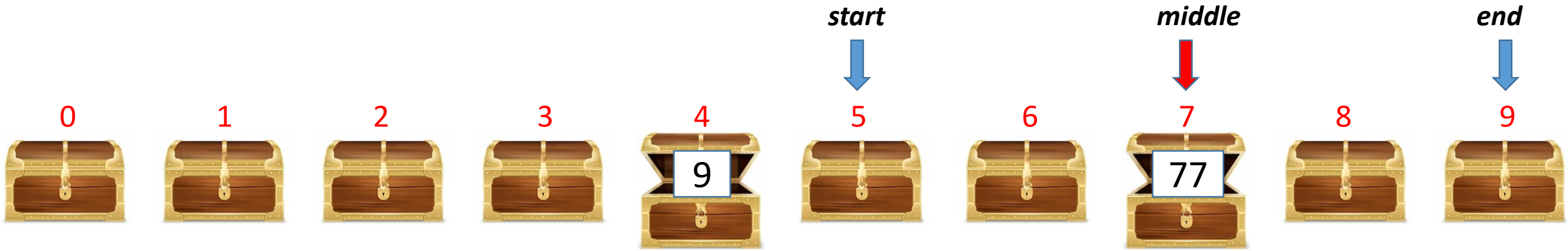


*Знаходимо елемент **middle**, що знаходиться між **start** та **end***

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor((***start***+***end***)/2)

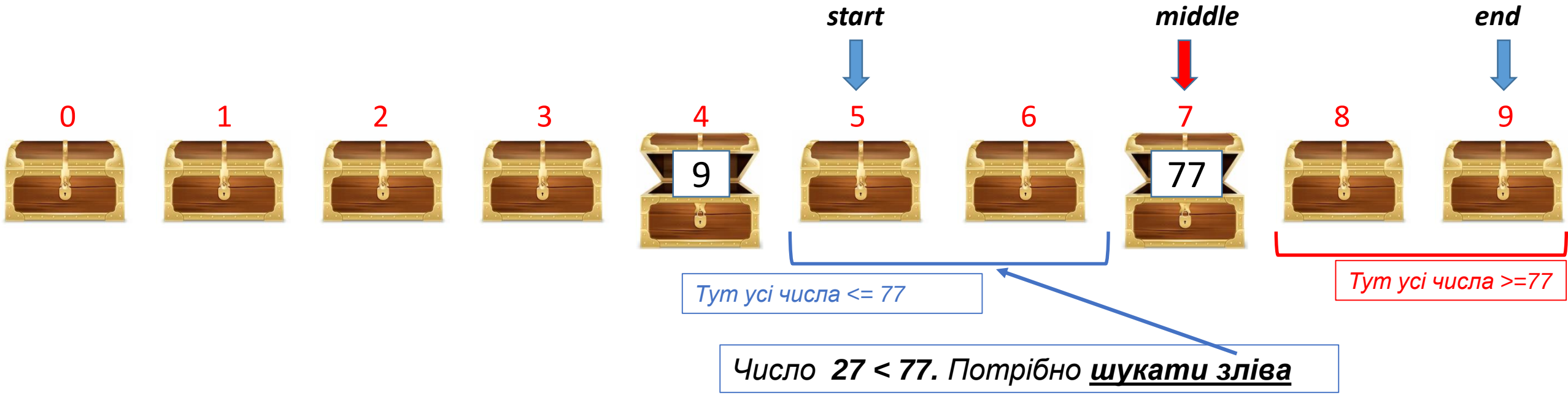


*Порівнюємо елемент, що знаходиться у позиції **middle** з шуканим елментом 27*

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

middle = Math.floor((***start***+***end***)/2)

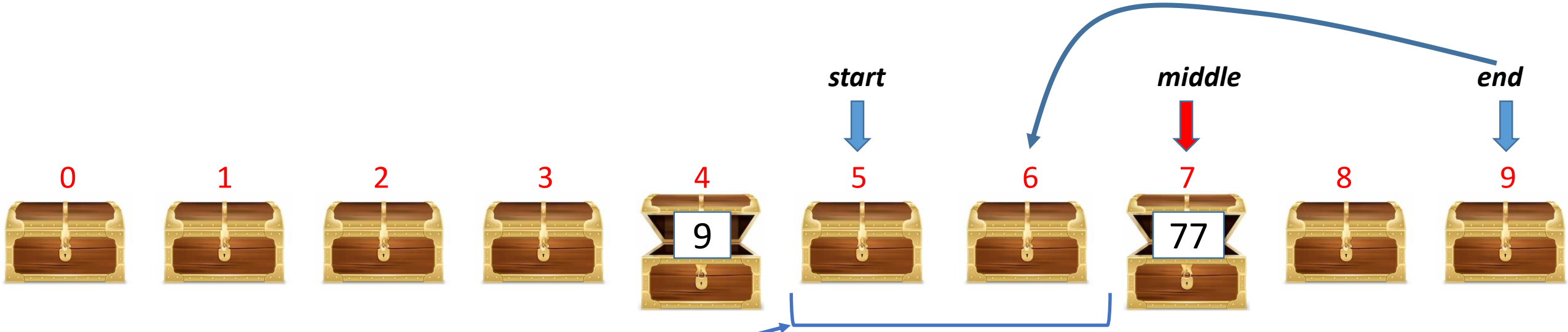


*Число **27 < 77**. Потрібно **<u>шукати зліва</u>***

Двійковий (бінарний) пошук

*Числа у скринях **упорядковані за зростанням**. Шукаємо число **27***

***middle** = Math.floor((**start**+**end**)/2)*



*Число **27 < 77**. Потрібно **<u>шукати зліва</u>***
<u>end</u>** – зміщуємо вліво від **<u>middle</u>
<u>(end = middle – 1)</u>

```
function binarySearch(arr, searchElement, start, end) {
  if (start <= end) {
    const middle = Math.floor((start + end) / 2)
    if (arr[middle] === searchElement) return middle
    if (arr[middle] < searchElement)
      return binarySearch(arr, searchElement, middle + 1, end)
    if (arr[middle] > searchElement)
      return binarySearch(arr, searchElement, start, middle - 1)
  } else return -1
}
```

```
function binarySearch(arr, searchElement, start, end) {
    if (start <= end) {
        const middle = Math.floor((start + end) / 2)
        if (arr[middle] === searchElement) return middle
        if (arr[middle] < searchElement)
            return binarySearch(arr, searchElement, middle + 1, end)
        if (arr[middle] > searchElement)
            return binarySearch(arr, searchElement, start, middle - 1)
    } else return -1
}
```

```
let arr = [0, 1, 3, 4, 6, 8, 9, 11, 23, 45]

document.write(binarySearch(arr, 231, 0, arr.length - 1))
```