Project 1 Sequence

CS 211 _ Fall 2020

COMPUTER SCIENCE
COLLEGE OF ENGINEERING

UIC

Step 1 – Read in data & store in array

Two types of data files:

Index comparison input

```
21  12
81  76  37  20
41  84  2   -999
1
```

Target sum input

```
21  12
81  76  37  20
41  84  2   -999
2
36  88
46  19  -999
```

Step 1 – Read in data & store in array

Index Comparison:

- Data file contains multiple integer values and one value of -999 followed by 1.

Index comparison input

```
21   12
81   76   37   20
41   84   2    -999
1
```

Step 1 – Read in data & store in array

Target sum:

- Data file contains multiple integers with two values of -999
- The first value of -999 is followed by a 2

Target sum input

```
21   12
81   76    37    20
41   84    2     -999
2
36   88
46   19    -999
```

Step 1 – Read in data & store in array

Target sum:

- Data file contains multiple integers with two values of -999
- The first value of -999 is followed by a 2

Data to be stored in the array

Target sum input

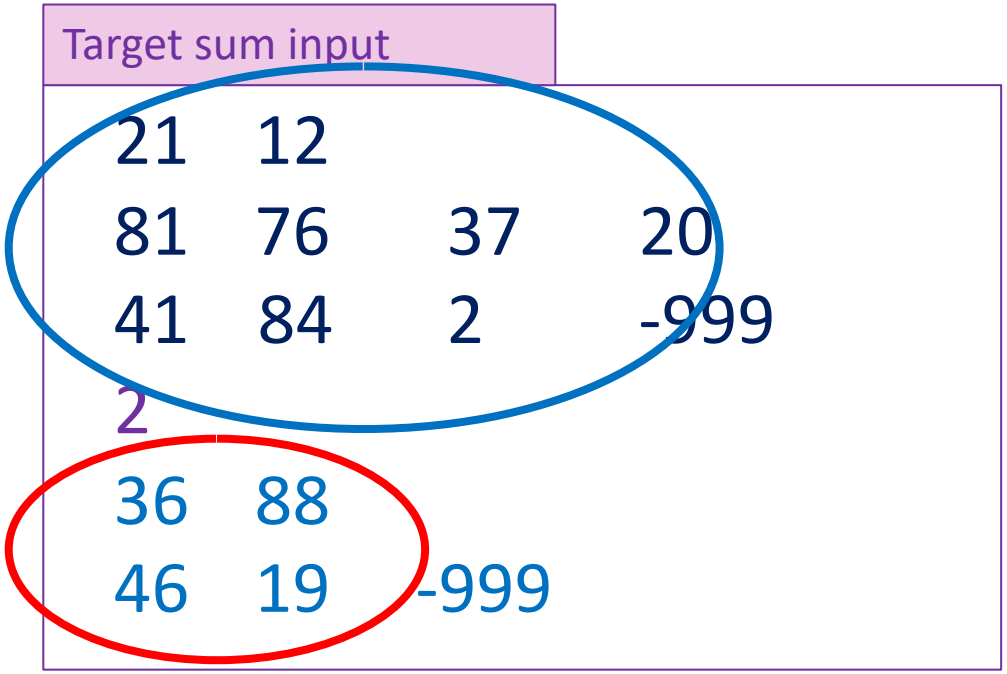
```
21   12
81   76    37    20
41   84    2     -999
2
36   88
46   19    -999
```

Step 1 – Read in data & store in array

Target sum:

- Data file contains multiple integers with two values of -999
- The first value of -999 is followed by a 2

Data to be stored in the array



Data to use for searching

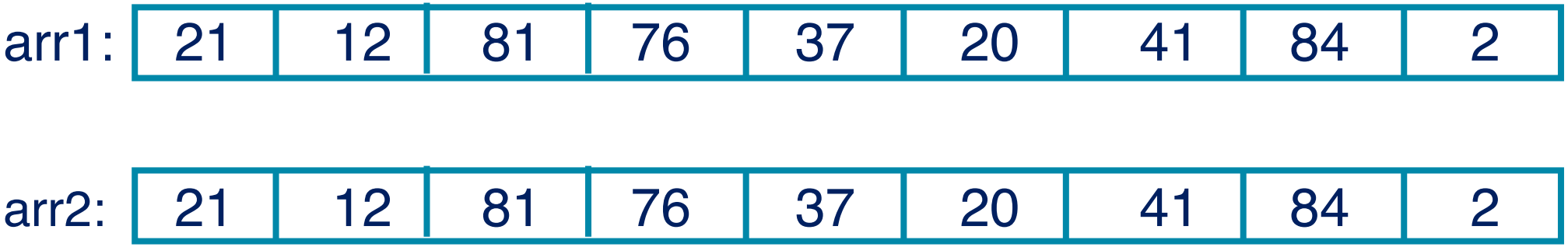
Step 1 – Read in data & store in array

- First read in values until first -999 is encountered
- Store these values into an array

- Dynamic Memory Allocation
- Grow the dynamic array when needed

Step 2 – Make of copy of the array

- Allocated enough space for the copy
- Call arrayCopy() to transfer values



Step 3 – Sort one of the arrays

- Call myFavoriteSort() and sort the second array
- “Write your own function….” i.e. you can’t call a library routine

arr1:	21	12	81	76	37	20	41	84	2

arr2:	2	12	20	21	37	41	76	81	84

Step 4 – Read in user input for task

- Prompt the user to enter a value to invoke a task
- User must enter either 1 or 2
- If you read a value that is not 1 or 2, display a warning message and read the user input again.

- User input = 1 → call indexComparison( )
- User input = 2 → call targetSum( )

Step 5.1 – Index Comparison

- How many elements are still in the same index after sort?
- A simple iteration

Must use the given prototype:

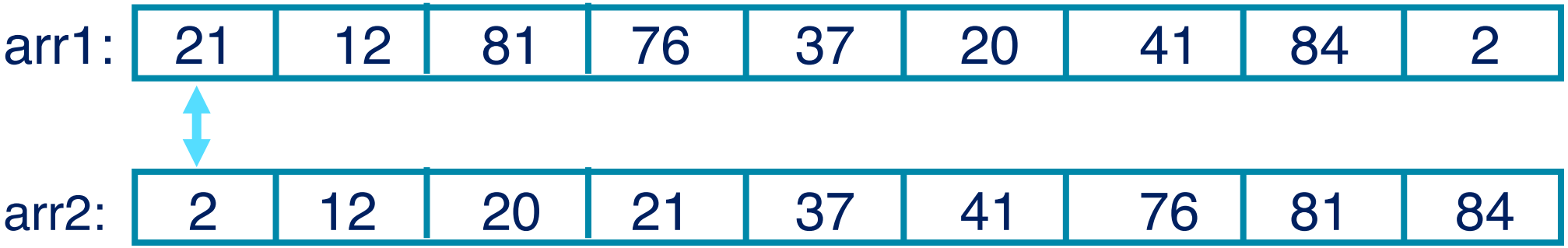
```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```

arr1:	21	12	81	76	37	20	41	84	2

arr2:	2	12	20	21	37	41	76	81	84

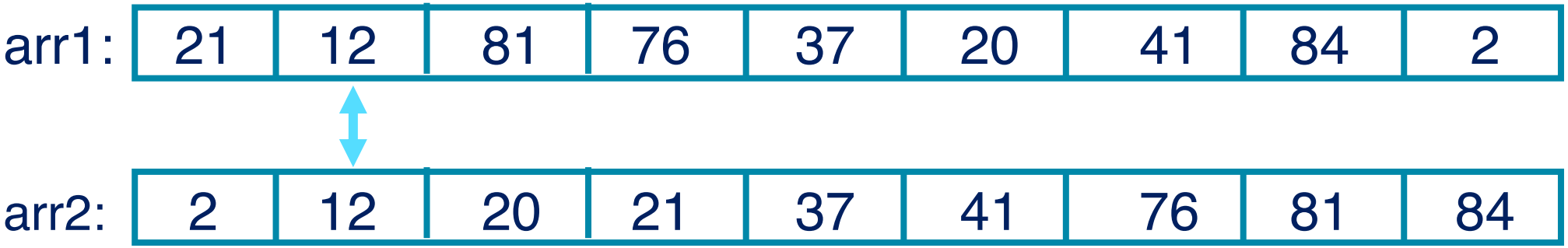
Step 5.1 – Index Comparison

```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



Step 5.1 – Index Comparison

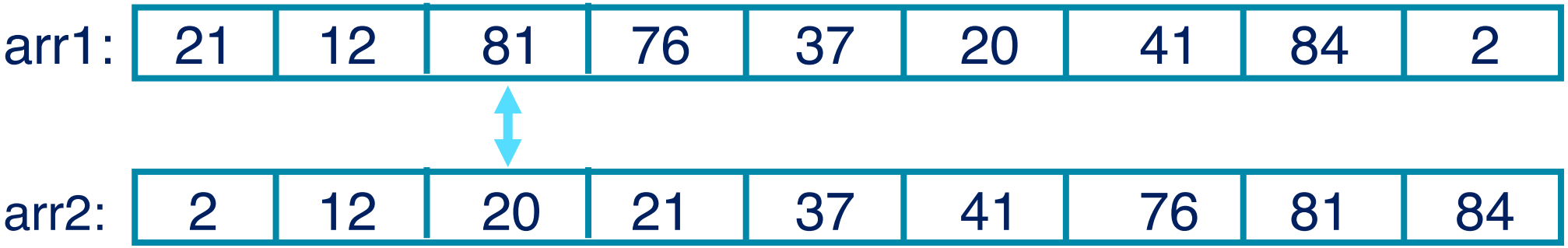
```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



Increase the counter

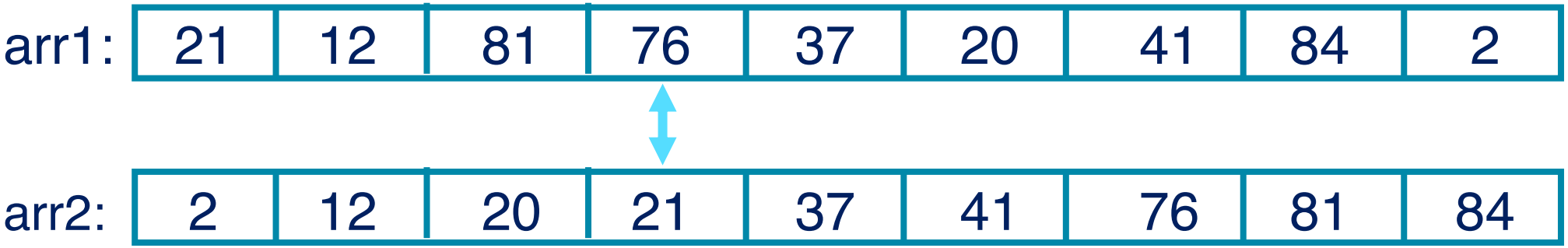
Step 5.1 – Index Comparison

```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



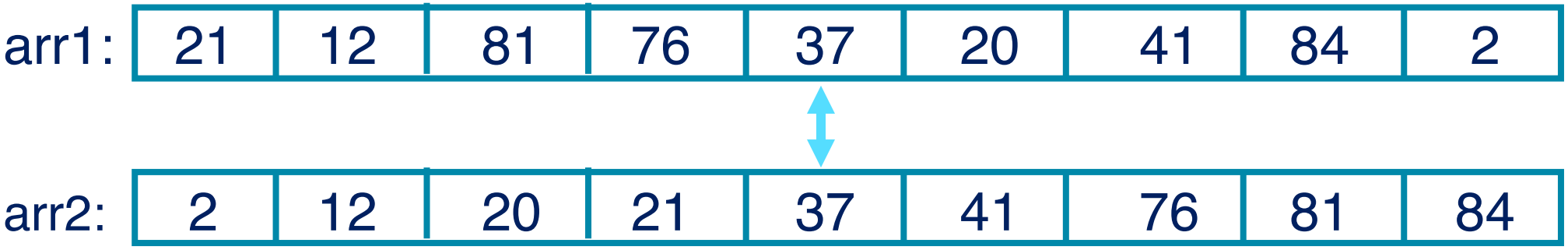
Step 5.1 – Index Comparison

```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



Step 5.1 – Index Comparison

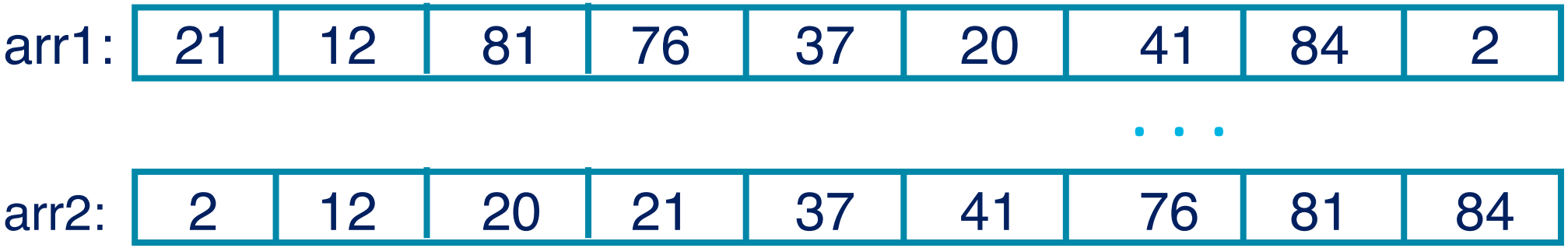
```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



Increase the counter

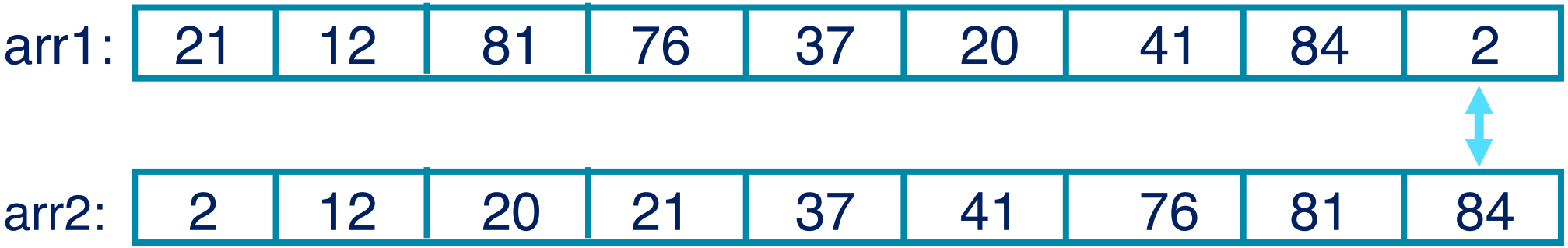
Step 5.1 – Index Comparison

```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



Step 5.1 – Index Comparison

```
void indexComparison (int unsortedArray[],int sortedArray[] ,
                      int size, int *counter);
```



2 elements are in the same location in both arrays

Step 5.2 – Target Sum: Read in the search data

- Loop until second value of -999 is encountered
- For each value, call targetSum ()
  - Print out results in main() or whichever function calls the targetSum

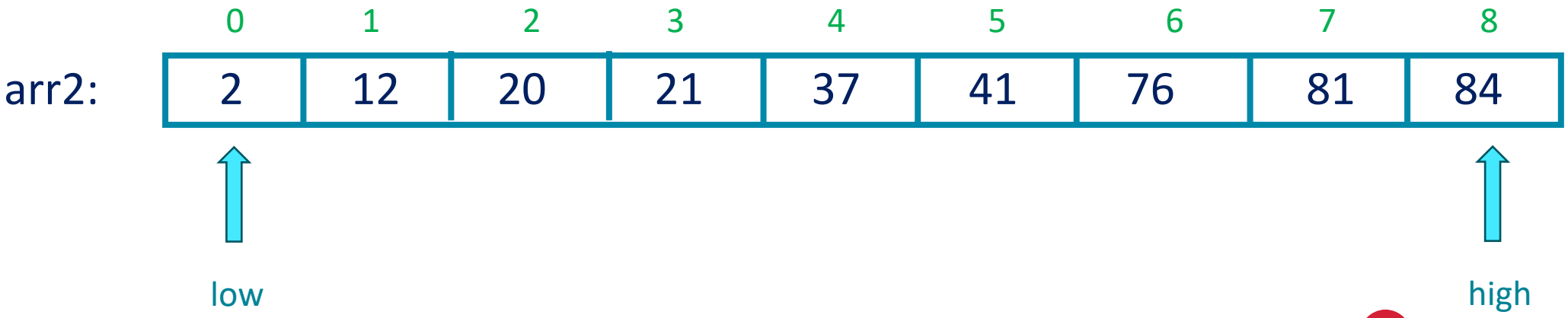
Target sum input

```
21   12
81   76    37    20
41   84    2     -999
2
36   88
46   19    -999
```

Data to use for searching

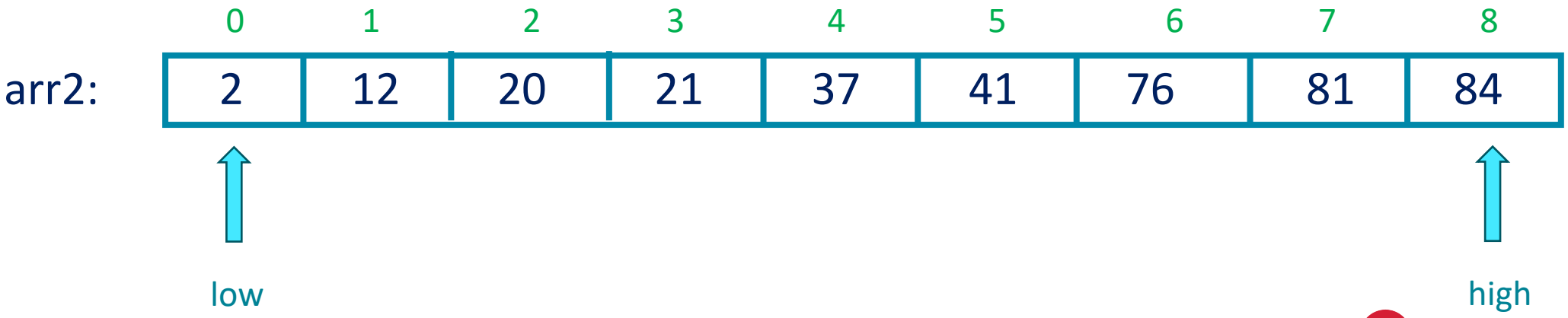
Step 5.2 – Target Sum function

- First value is 36
  - Set the iterators (index 0 and index size-1)



Step 5.2 – Target Sum function

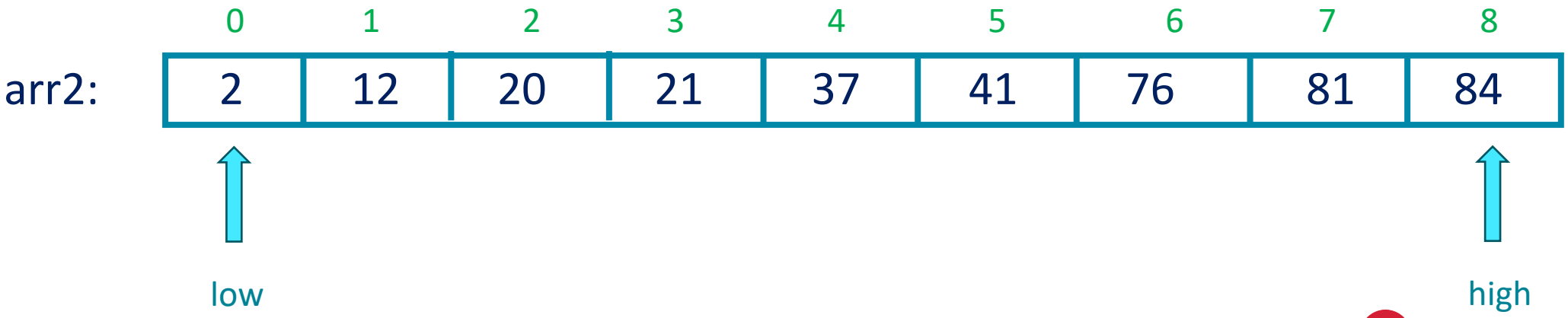
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

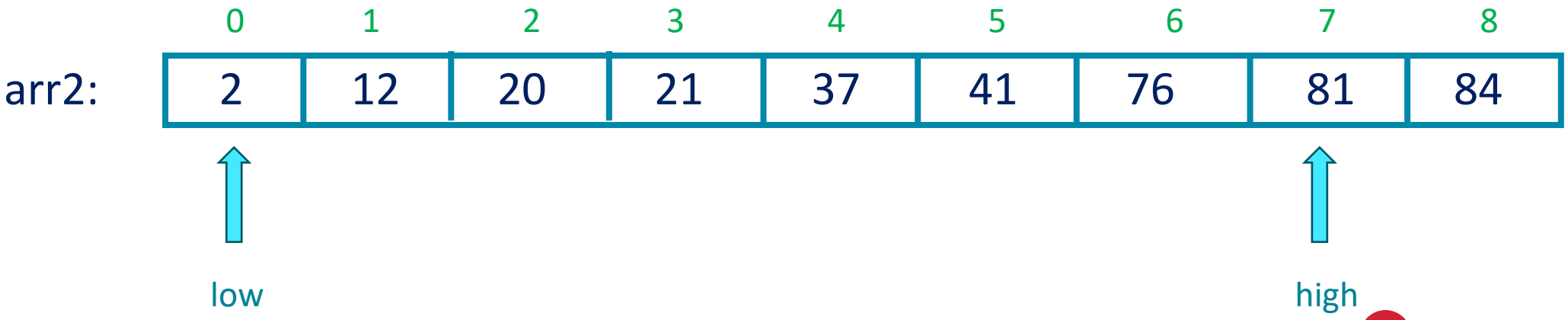
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 84 = 86 > 36 → Move “high” left



Step 5.2 – Target Sum function

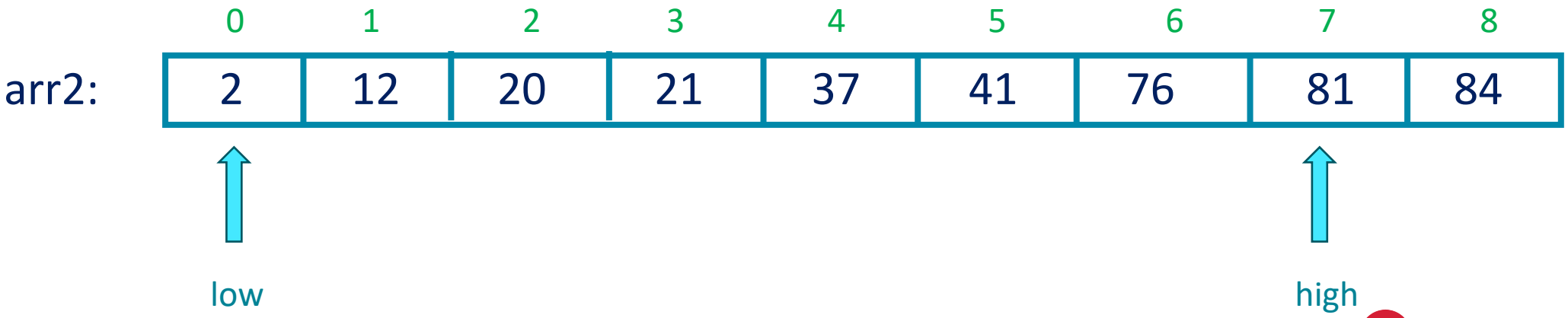
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

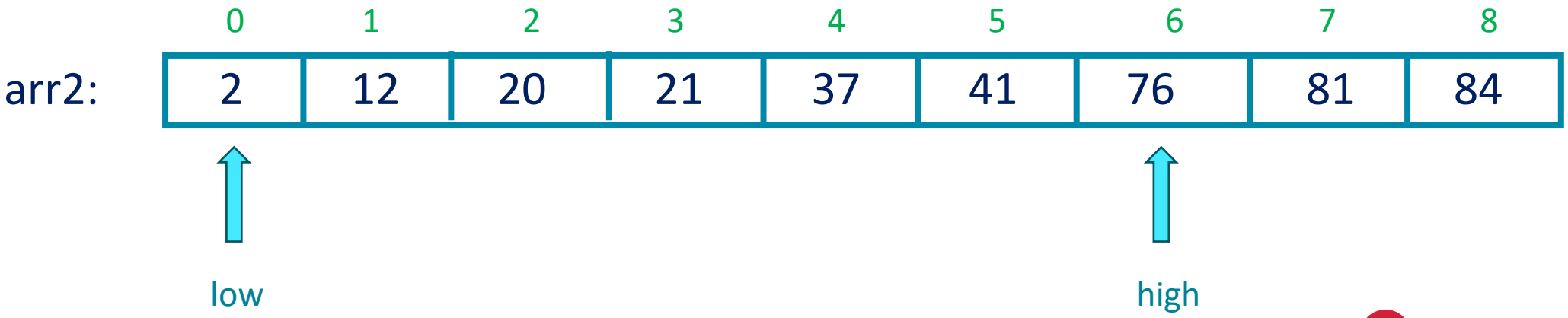
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 81 = 83 > 36 → Move “high” left



Step 5.2 – Target Sum function

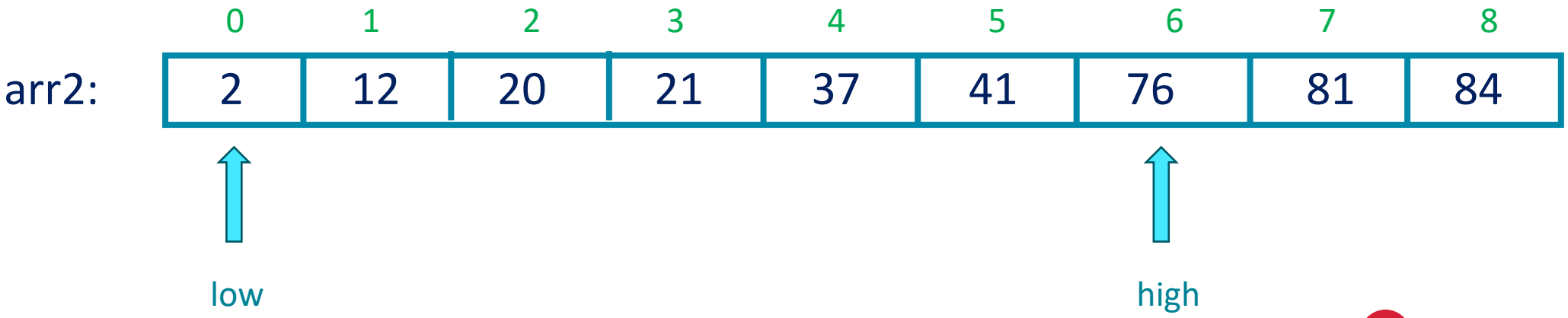
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

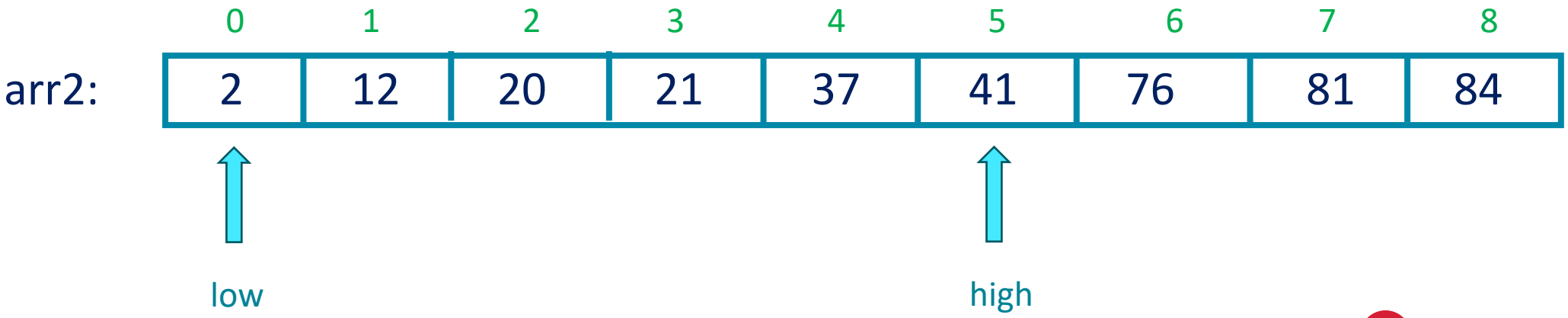
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 76 = 78 > 36 → Move “high” left



Step 5.2 – Target Sum function

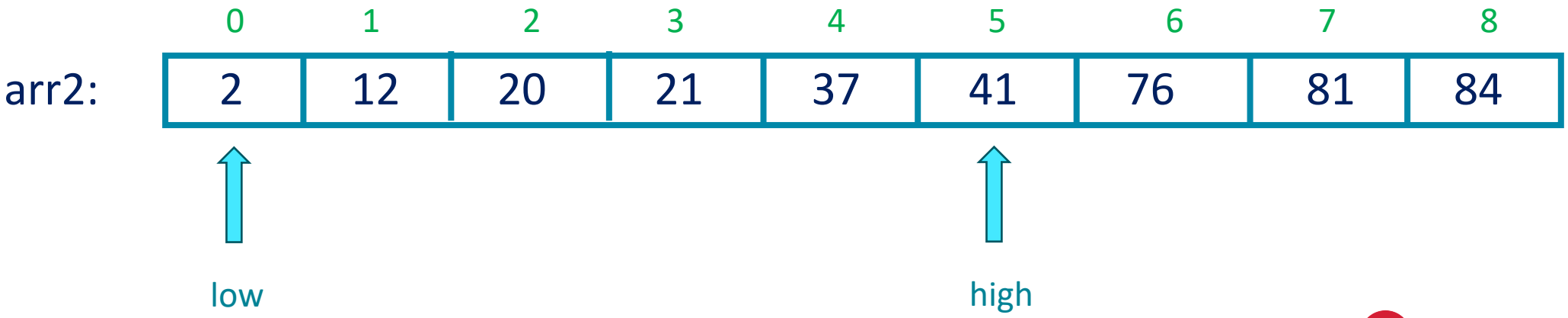
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

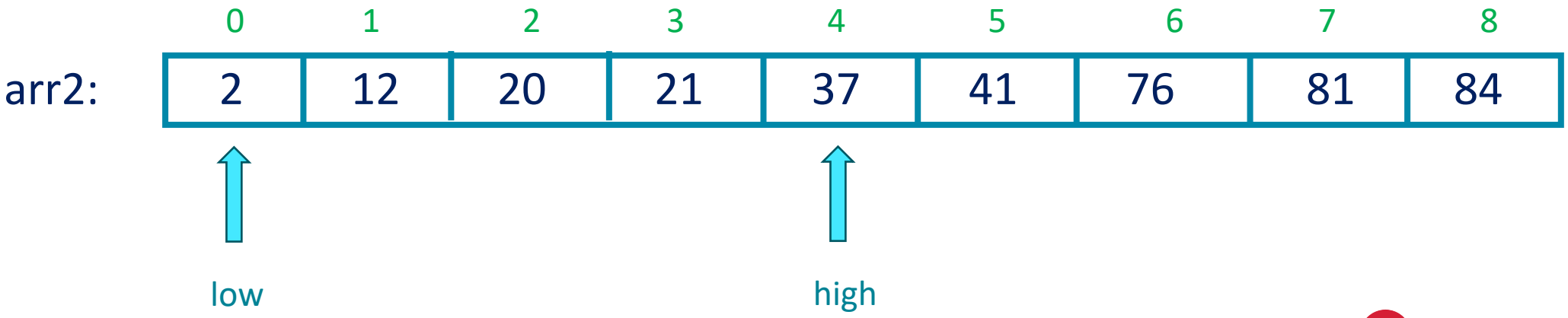
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 41 = 43 > 36 → Move “high” left



Step 5.2 – Target Sum function

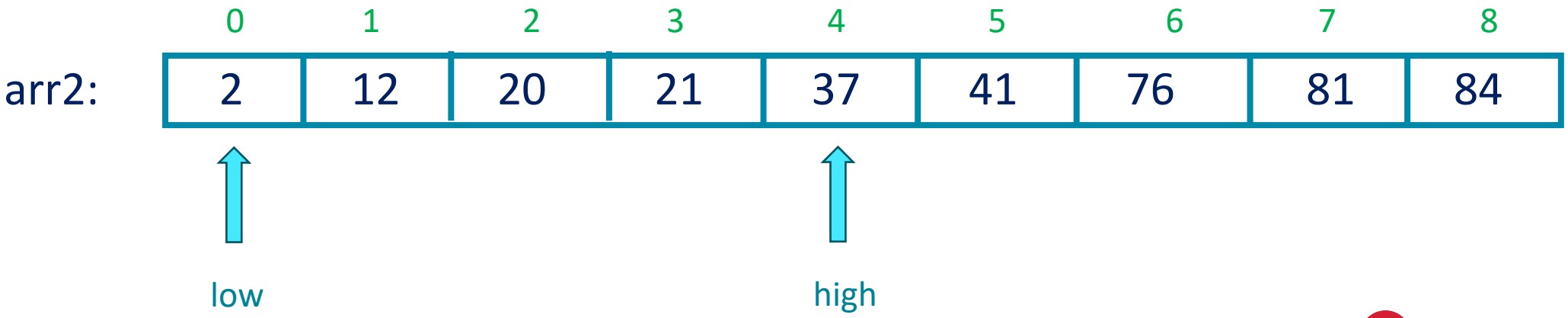
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

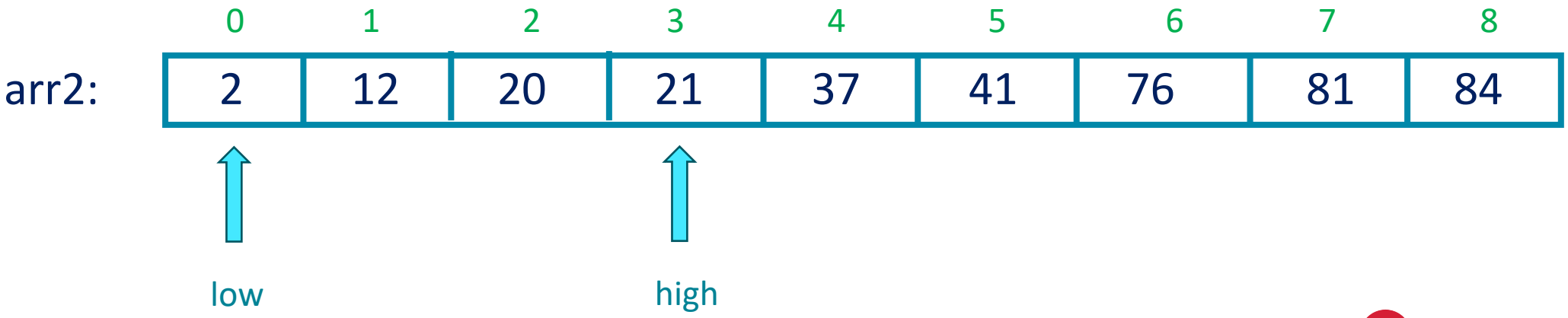
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 37 = 39 > 36 → Move “high” left



Step 5.2 – Target Sum function

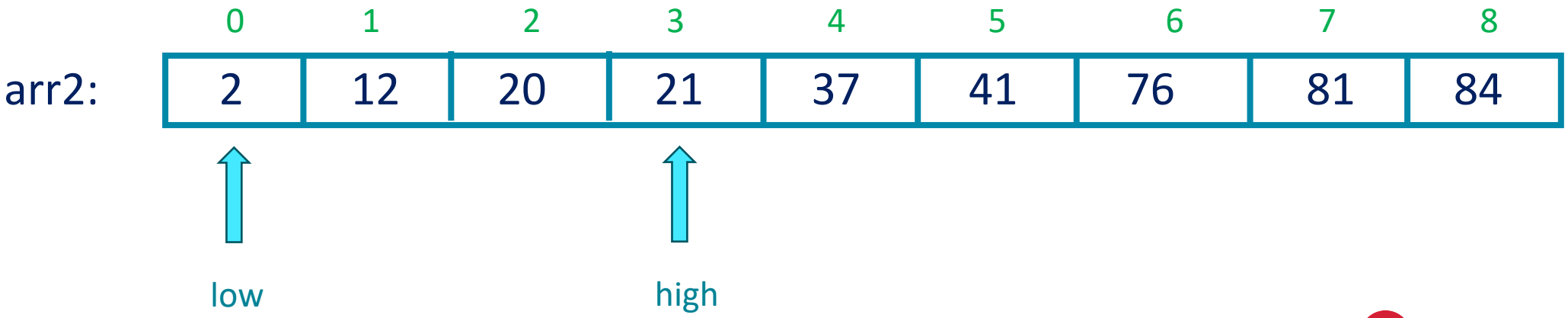
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

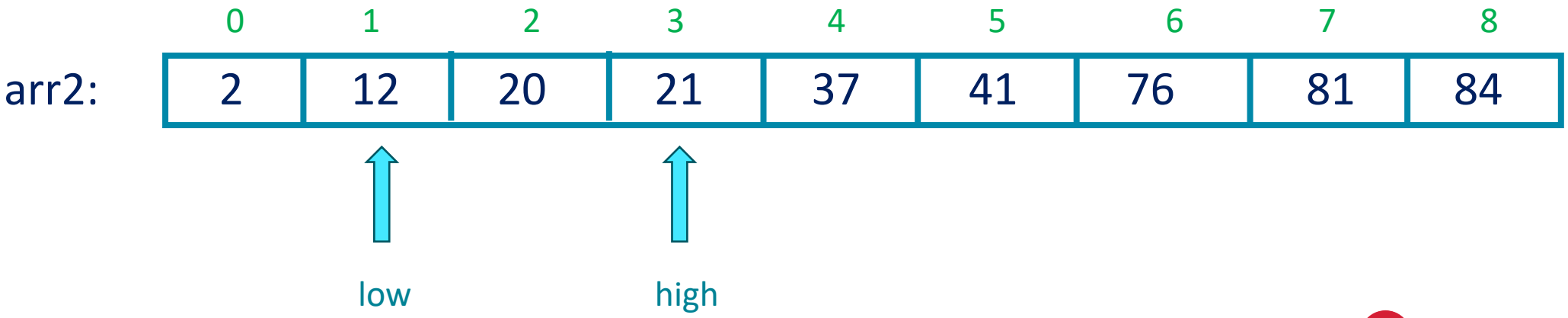
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 21 = 23 < 36 → Move “low” right



Step 5.2 – Target Sum function

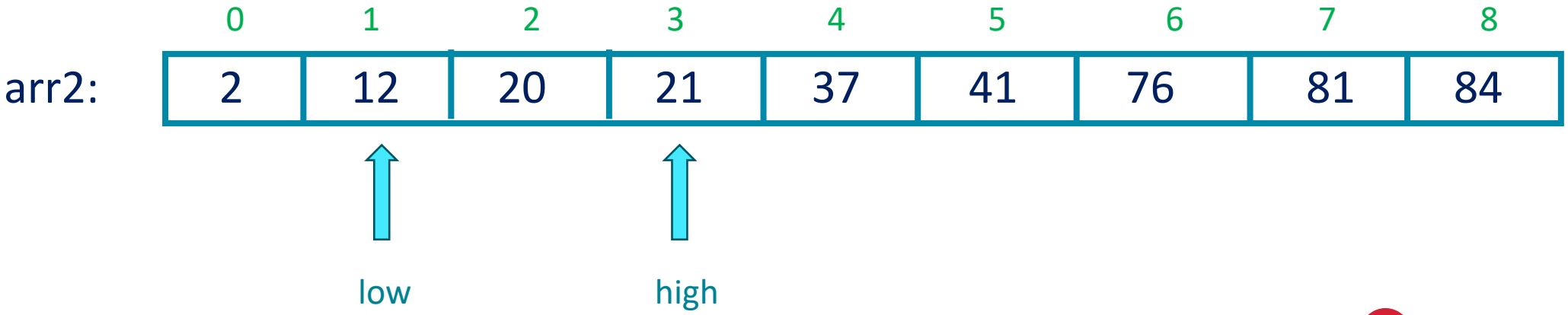
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

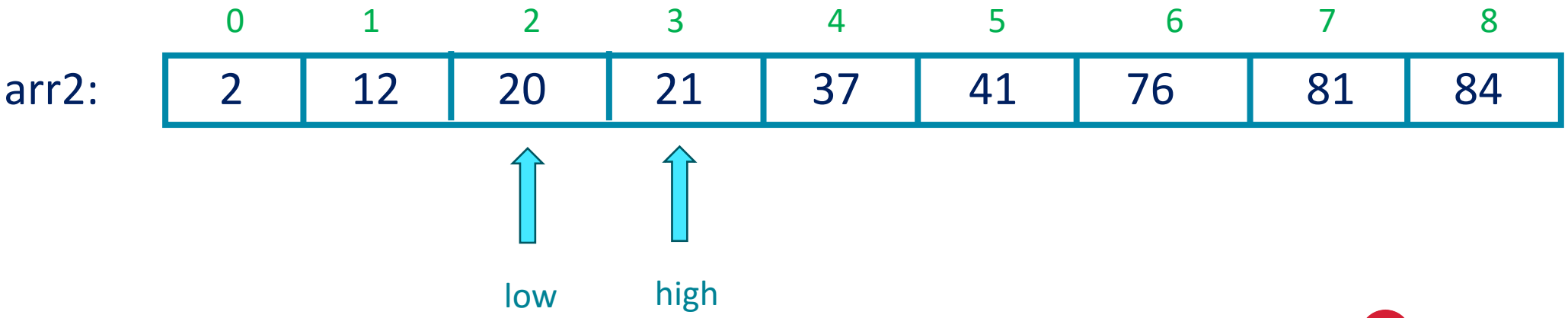
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

12 + 21 = 33 < 36 → Move “low” right



Step 5.2 – Target Sum function

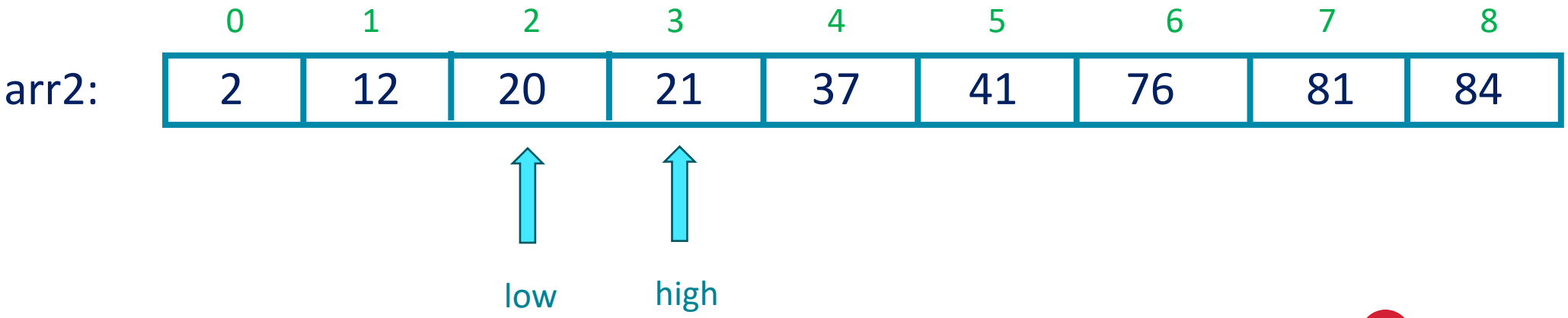
- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

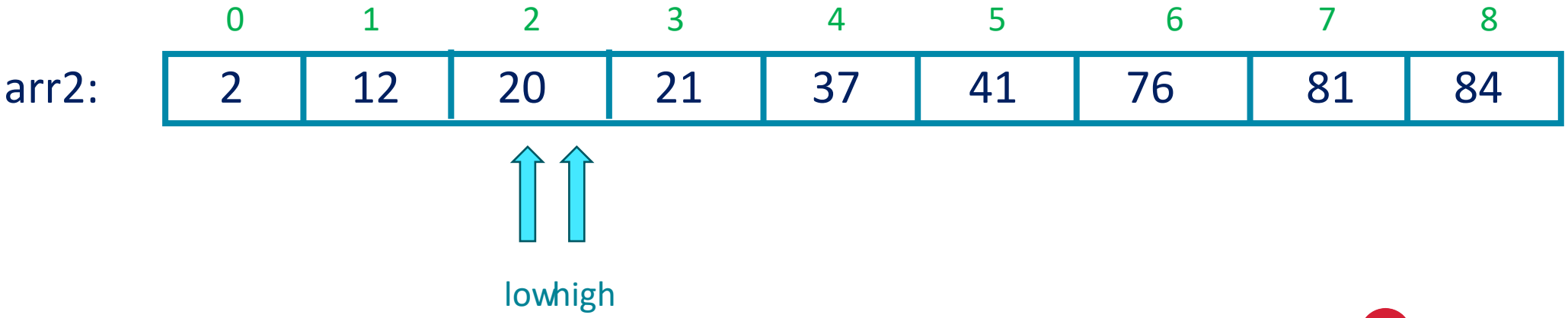
20 + 21 = 41 > 36 → Move “high” left



Step 5.2 – Target Sum function

- First value is 36
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

”low” and “high” align, target sum failed , return and print statement!



Step 5.2 – Target Sum: Read in the search data

- Loop until second value of -999 is encountered
- For each value, call targetSum ()
  - Print out results in main() or whichever function calls the targetSum

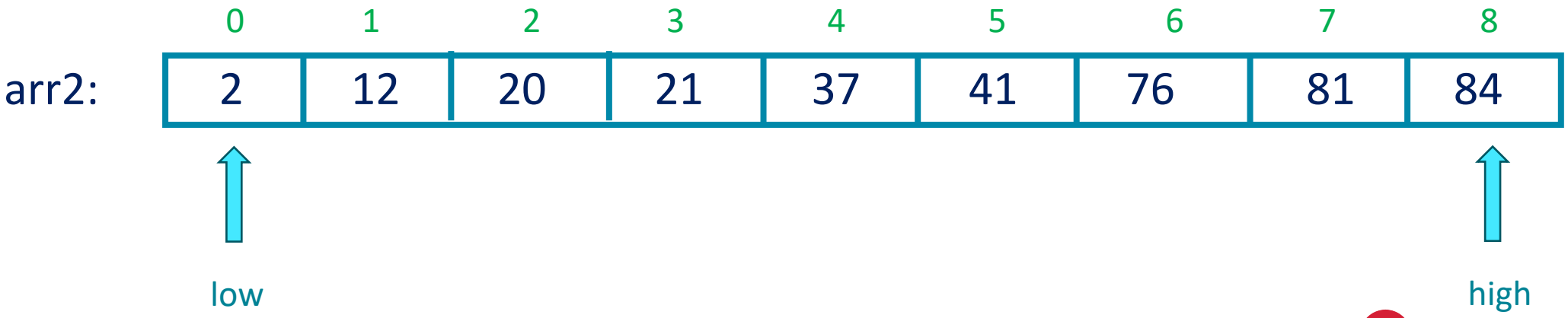
Target sum input

```
21   12
81   76    37    20
41   84    2     -999
2
36   88
46   19    -999
```

Data to use for searching

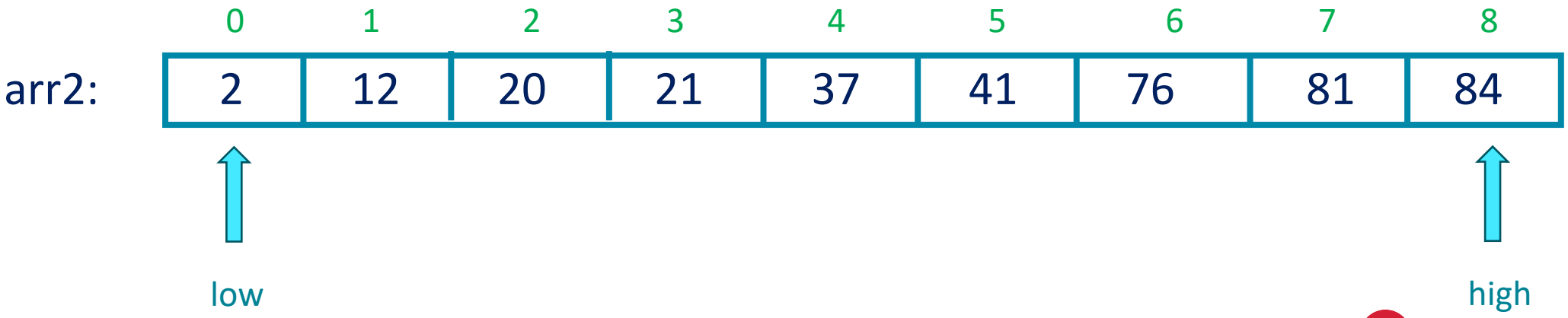
Step 5.2 – Target Sum function

- Second value is 88
  - Set the iterators (index 0 and index size-1)



Step 5.2 – Target Sum function

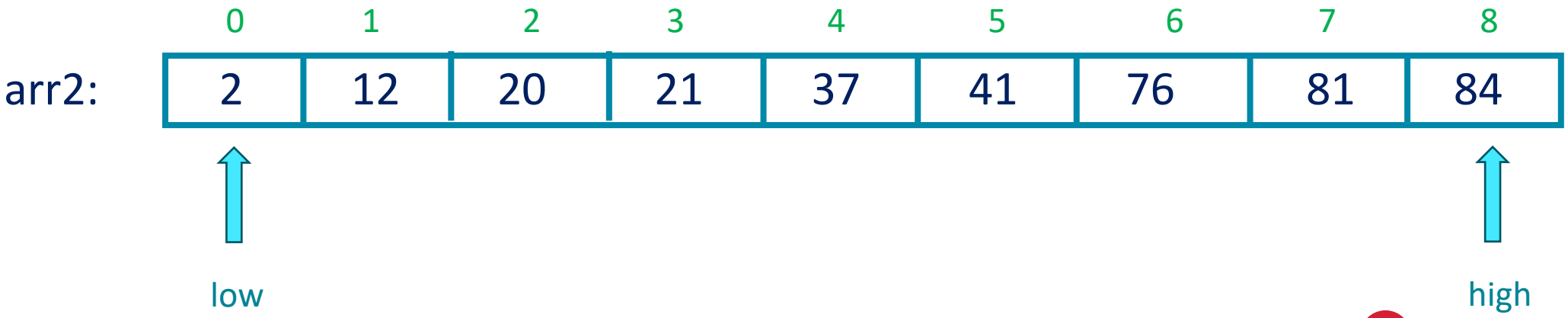
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

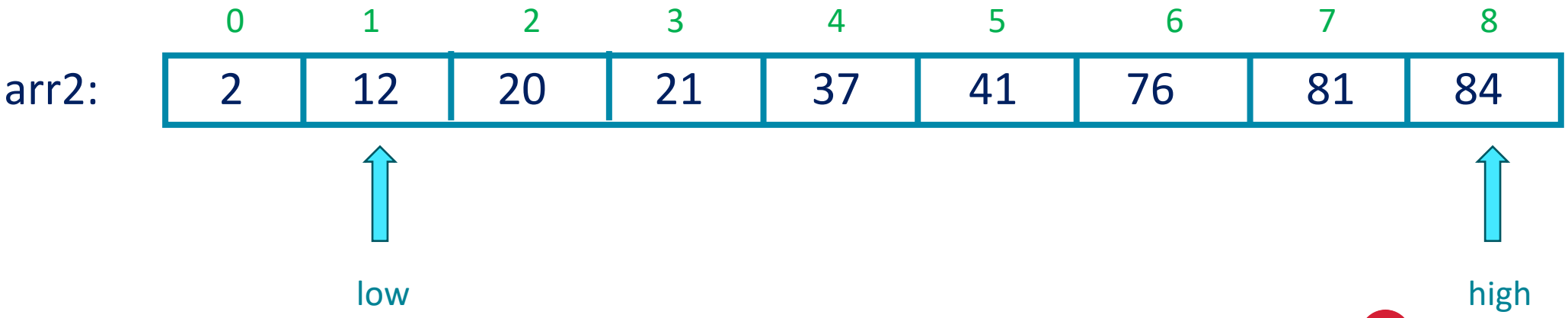
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

2 + 84 = 86 < 88 → Move “low” right



Step 5.2 – Target Sum function

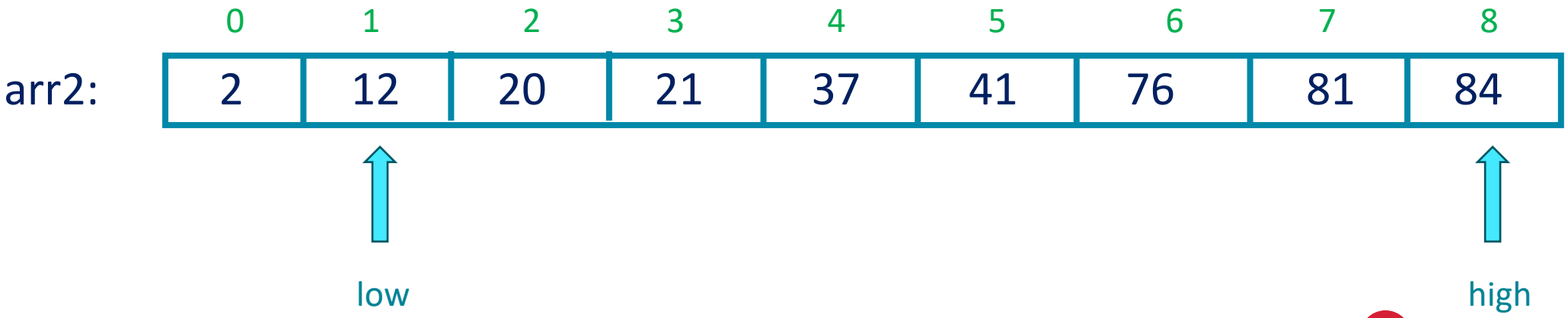
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

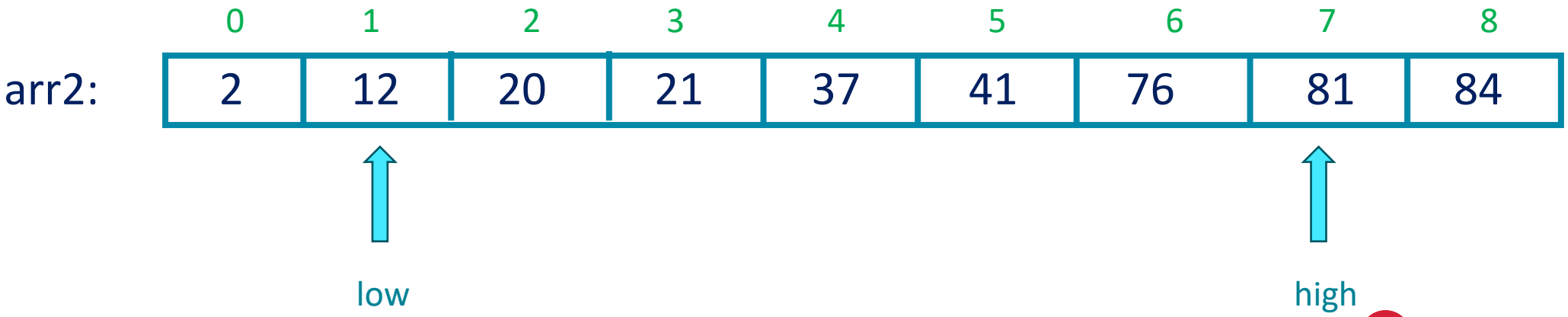
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

12 + 84 = 96 > 88 → Move “high” left



Step 5.2 – Target Sum function

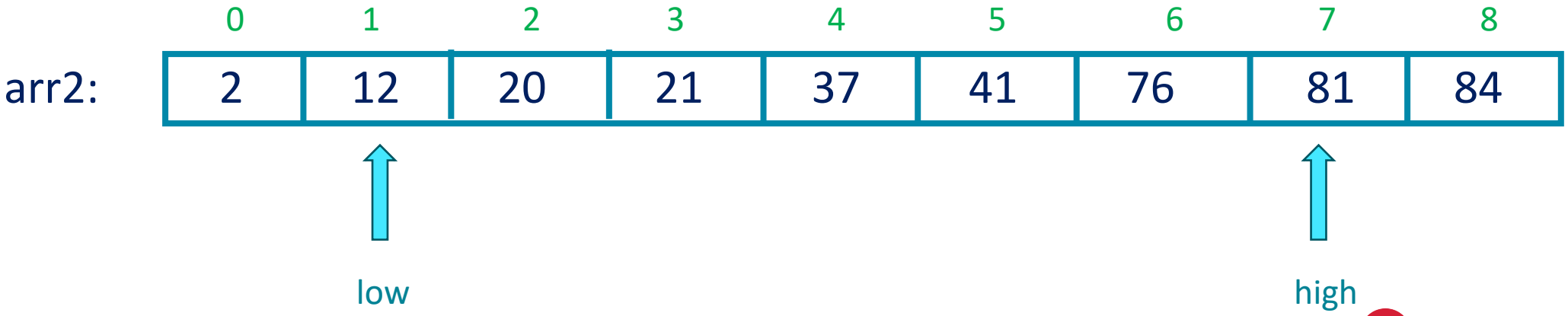
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

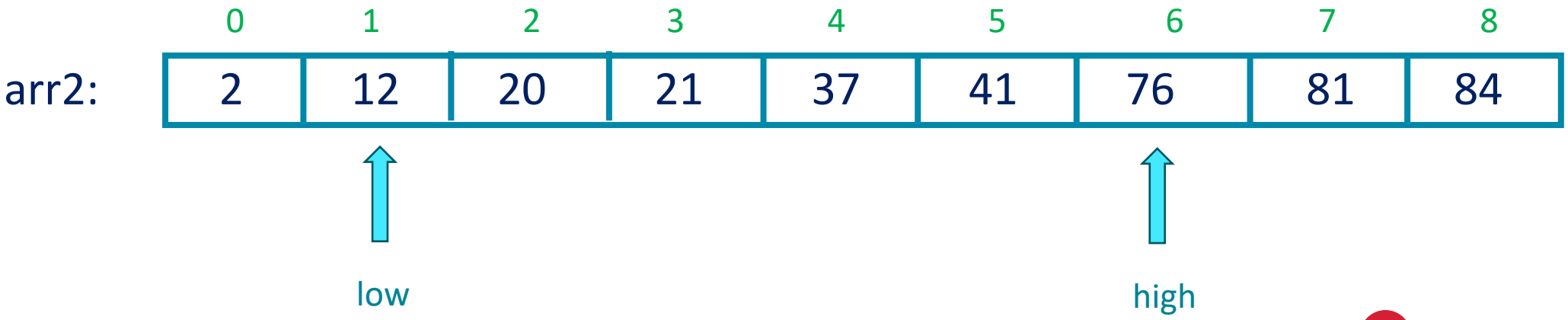
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

12 + 81 = 93 > 88 → Move “high” left



Step 5.2 – Target Sum function

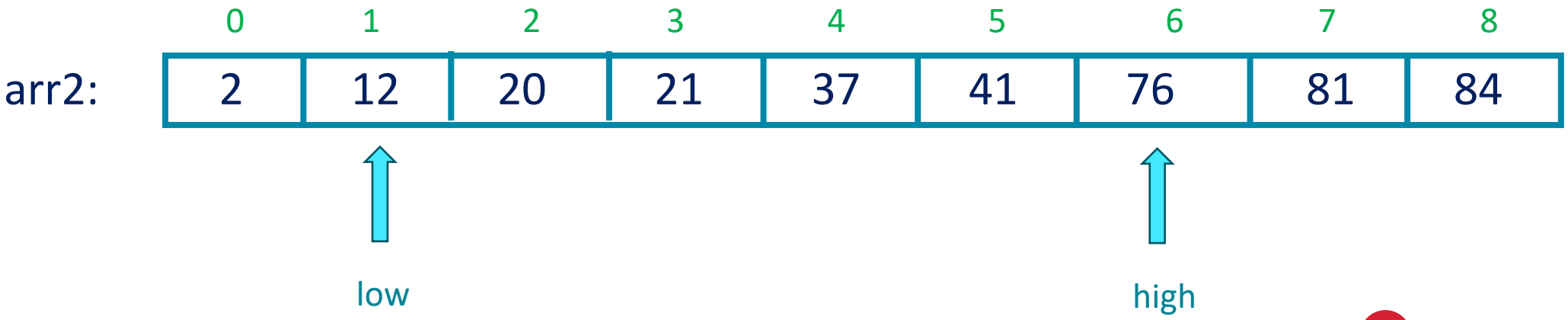
- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to



Step 5.2 – Target Sum function

- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

12 + 76 = 88 → **Target matched, return and print statements**



Step 5.2 – Target Sum function

Success! Elements at indices 1 and 6 add up to 88.

- Second value is 88
  - Set the iterators (index 0 and index size-1)
  - Check the sum of values “low” and “high” are pointing to

12 + 76 = 88 → **Target matched, return and print statements**