PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

pdst.ie

An Roinn Oideachais
Department of Education

NW4 Session 2

Algorithms II

LEAVING CERTIFICATE
COMPUTER SCIENCE

Overview of the Session

Part 1	Unconscious bias and algorithmic bias
Part 2	Introduction to algorithms (revisited)
Part 3	Searching algorithms

By the end of this session participants will have:

- participated in an interactive dialogue on unconscious bias and algorithmic bias
- revisited the definition and characteristics of algorithms, as well as the ubiquitous nature of algorithms in today's society.
- participated in a coding activity relating to the linear search algorithm
- developed their understanding of the binary search algorithm

PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

Section 1

Unconscious bias & algorithmic bias

The builder, the nurse and the doctor

A builder, leaning out of the van, shouts “nice legs” to a nurse passing by. The same nurse arrives at work, and casually mentions this to a senior doctor. The doctor said,” I’d never say that”.

The doctor has two grown up children who are 22 and 30. They get on very well. One is a Sergeant in the Army; the other is training to be a beauty therapist. The doctor divorced last year and is currently dating someone else.

	True	False	Don't know
The builder was driving a van			✓
The van was travelling quicker than the nurse			✓
There was at least one man in the van			✓
Not every man mentioned would shout "nice legs"			✓
The doctor is no longer living with his wife			✓
The doctor has a new girlfriend			✓
The doctor's son is in the army			✓
The youngest child is training to be a beauty therapist			✓
At some point a man spoke to a woman			✓
At least two of the people mentioned are men			✓
A woman was shouted at			✓

What is unconscious bias?

Natural

Rapid categorization of people

Created by social influence

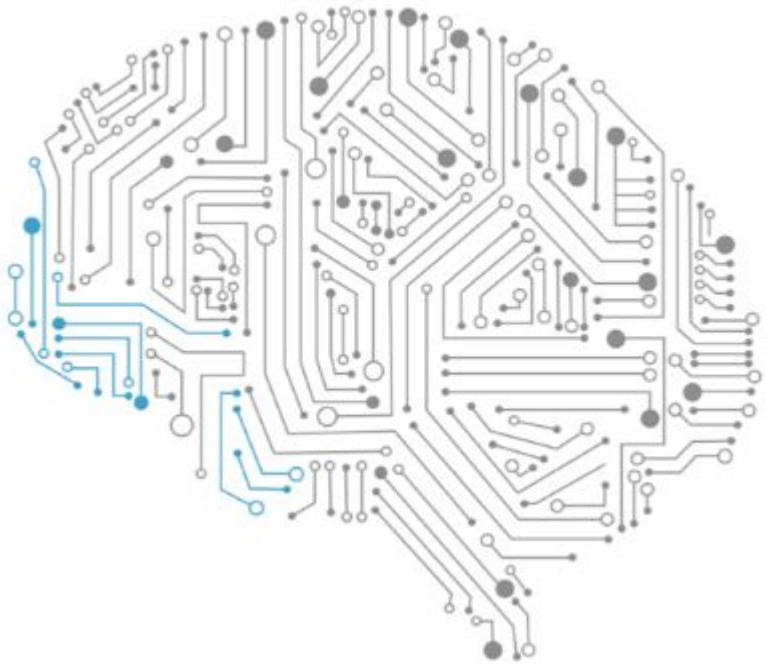
Unintentional

Used by everyone

Most likely to be acted on when we are stressed or tired

A bad thinking habit

What does unconscious bias have to do with algorithms?



"Some of the biggest problems in the industry aren't technical – they're people (egos etc.) …. diversity creates better solutions and opportunities because of wider experience set, perspectives and ***less bias …"***

James Whelton, CoderDojo Co-Founder

Algorithmic bias

“The nurse said that…”

“The pilot said that…”

“The computer programmer said that…”

The Feedback Loop

"You are not only allowing it (the computer system) to make a decision, you are also allowing it to determine/create the future they have predicted"

What does unconscious bias have to do with YOU?

Personal reflection

What are my values as a teacher?

"We do not learn from experience… we learn from reflecting on experience."

John Dewey

"Sincerity of intention does not guarantee purity of practice."

Stephen Brookfield

PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

Section 2

Introduction to algorithms - revisited

Algorithms and the Specification

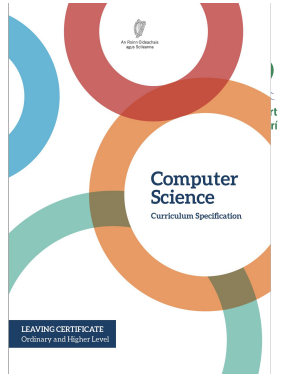
"Computer science is the study of computers and algorithmic processes. Leaving Certificate Computer Science includes how programming and computational thinking can be applied to the solution of problems, and how computing technology impacts the world around us. "

NCCA Curriculum specification, Page 1

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
▸ Computers and society ▸ Computational thinking ▸ Design and development	▸ Abstraction ▸ Algorithms ▸ Computer systems ▸ Data ▸ Evaluation/Testing	▸ Applied learning task 1 - Interactive information systems ▸ Applied learning task 2 - Analytics ▸ Applied learning task 3 - Modelling and simulation ▸ Applied learning task 4 - Embedded systems

NCCA Curriculum specification, Page 11

LCCS Learning Outcomes



2.5 use pseudo code to outline the functionality of an algorithm

2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

2.7 implement algorithms using a programming language to solve a range of problems

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

2.9 assemble existing algorithms or create new ones that use functions (**including recursive**), procedures, and modules

2.10 explain the common measures of algorithmic efficiency using any algorithms studied

S2: Algorithms

Programming concepts

Sorting: Simple sort, Insert sort, Bubble sort, **Quicksort**

Search: Linear search, Binary search

Algorithmic complexity

See also learning outcomes 1.6, 1.7 1.14, 1.22, 2.3, 3.4 and 3.7 … plus others

What is an algorithm?

A step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

Merriam-Webster

- ✔ A sequence of instructions
- ✔ A way of capturing intelligence and sharing it with others
- ✔ Provide general solutions to problems
- ✔ Some problems are so hard that they cannot be solved by algorithms (computability)
- ✔ Can be expressed in a variety of different ways
- ✔ Common elements include input, processing, output
- ✔ Close relationship between algorithms and data structures
- ✔ Essential features are correctness and effectiveness
- ✔ Rule-based algorithms vs. Machine learning algorithms (AI)

What is an algorithm?

"An algorithm is a set of rules for getting a specific output from a specific input. Each step must be so precisely defined that it can be translated into computer language and executed by a machine"

Source: Knuth, D The Art of Computer Programming (Vol. 1, Fundamental Algorithms, 3rd ed.)

According to Knuth, an algorithm has five important features:

Donald Knuth

Finiteness	An algorithm must always terminate after a finite number of steps.
Definiteness	Each step must be precisely defined.
Input	An algorithm has zero or more inputs.
Output	An algorithm has one or more outputs, which have a specified relation to the inputs.
Effectiveness	All operations to be performed must be sufficiently basic that they can in principle be done exactly and in finite length of time by someone using pencil and paper.

Activity: Features of algorithms

"What features of an algorithm are demonstrated in the video?"

In what other contexts do you think the Gale-Shapley algorithm could be used?

Reflection: Features of algorithms

What features of an algorithm are demonstrated in the video?

Go to menti.com and enter the code:

6476 3273

OR

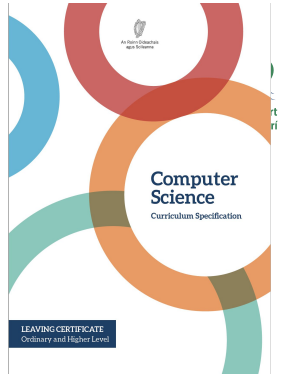
scan the QR code.

PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

Searching algorithms

Linear search

LCCS Learning Outcomes



2.5 use pseudo code to outline the functionality of an algorithm

2.6 construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement

2.7 implement algorithms using a programming language to solve a range of problems

2.8 apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm

2.9 assemble existing algorithms or create new ones that use functions (**including recursive**), procedures, and modules

2.10 explain the common measures of algorithmic efficiency using any algorithms studied

S2: Algorithms

Programming concepts

Sorting: Simple sort, Insert sort, Bubble sort, **Quicksort**

Search: Linear search, Binary search

Algorithmic complexity

See also learning outcomes 1.6, 1.7 1.14, 1.22, 2.3, 3.4 and 3.7 … plus others

Searching

An algorithm that maps the following input/output pair is called a search algorithm:

Input: An array, A, that contains n orderable elements (often called *keys*) A[0, 1, ... ,n-1] and some target value commonly referred to as an *argument*.

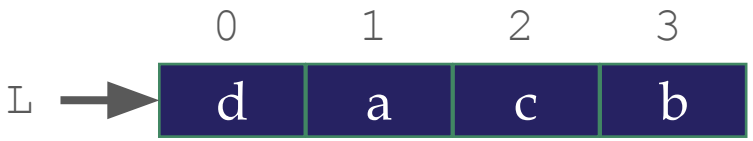
Output: If the argument is found in A it is conventional to return its zero-based positional offset (i.e. the index) and if the argument is not found some implementations return the length of the list while others return -1.

(Either of these two outputs can be used to indicate that the argument doesn't exist in A.)

Searching

For example, a search to find argument ‘c’ in the list L below would return 2.

A search to find argument ‘z’ (or any other value not on L) in L would return either -1 or 4 (the length of the list).



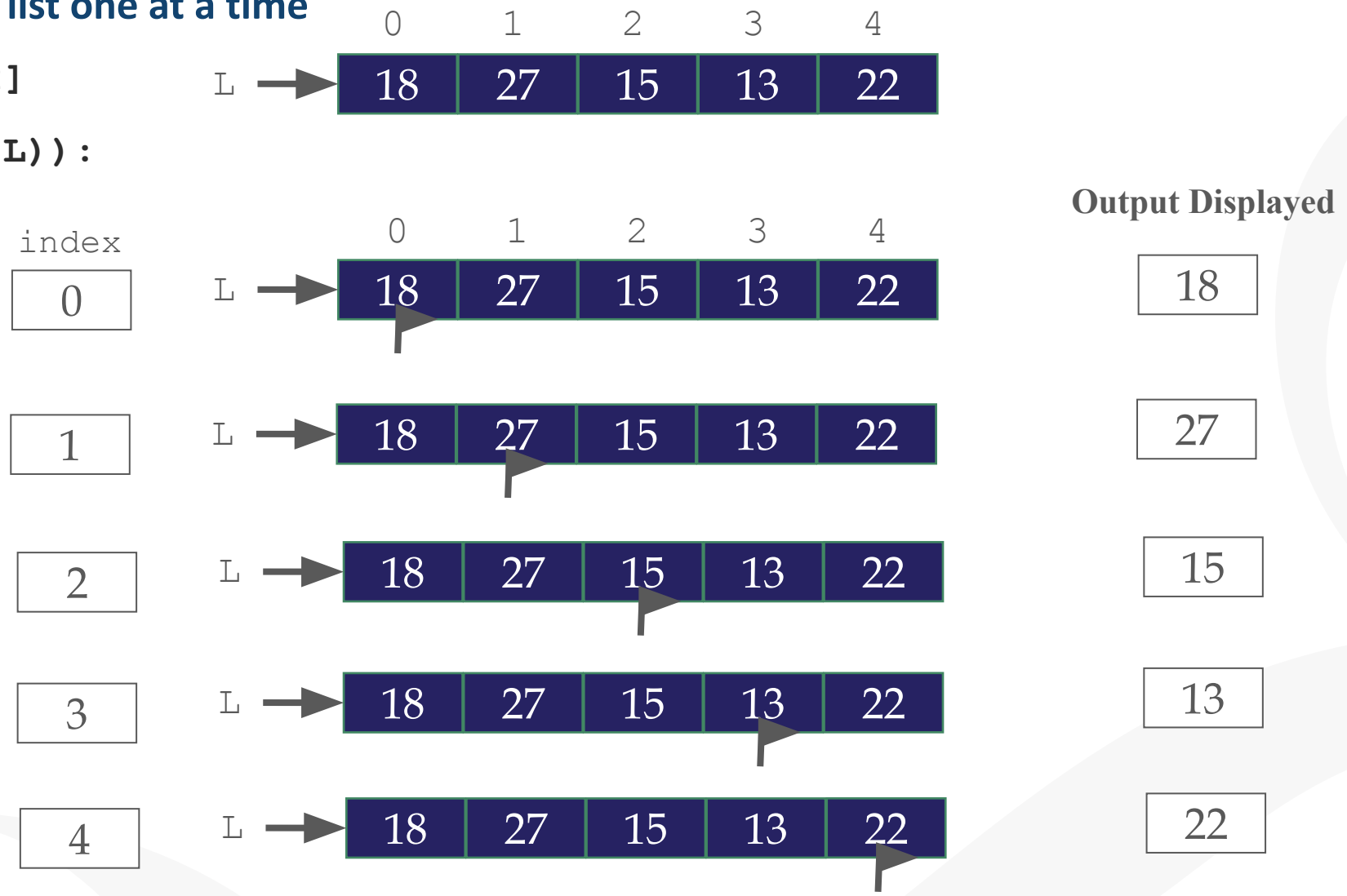
List traversal

Pass over each element in the list one at a time

```
L = [18, 27, 15, 13, 22]

for index in range(len(L)):
    print(L[index])
```

Equivalent to …

```
for v in L:
    print(v)


index = 0
while index < len(L):
    print(L[index])
    index = index + 1

print(L[0])
print(L[1])
print(L[2])
print(L[3])
print(L[4])
```



Linear Search

Question: does the list below contain the number 14?

If **14 is found**, what should the result be?
True?
The position (index) of 14?

If **14 is not found**, what should the result be?
False?
-1?
The length of the list?

Input:

A list L and a *target value* of 14

Input:

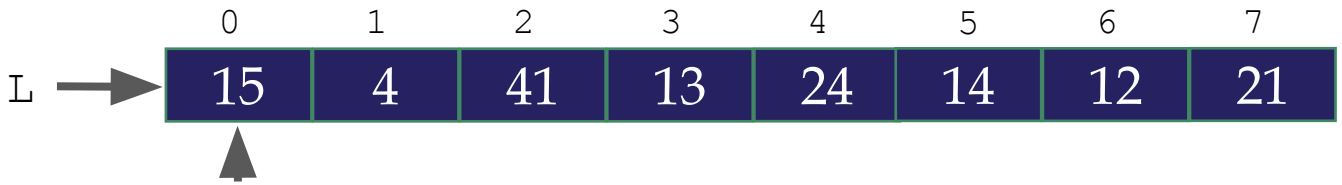
A list L and a *target value* of 14

Required Output:

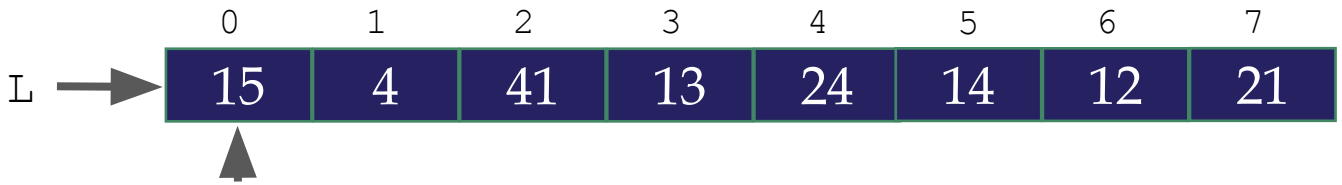
If the target value is found in L, its *index*

If the target value is not found in L, *-1* is returned

Start at the first element and ask is L[0] == the *target value*?

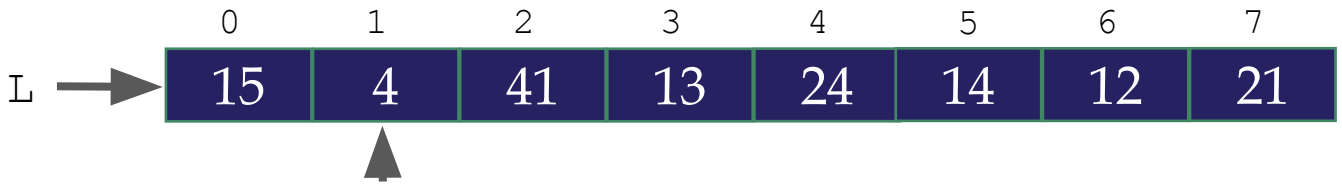


Start at the first element and ask is L[0] == the *target value*?



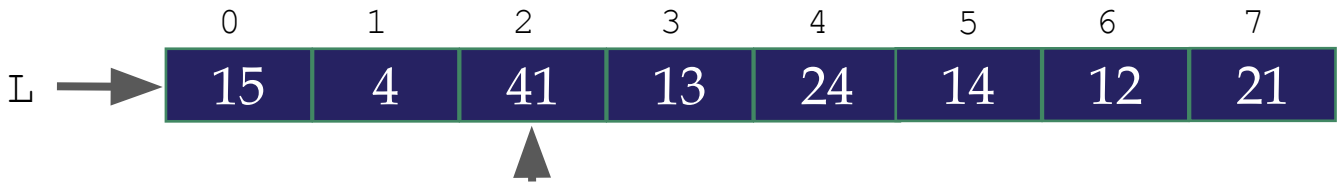
L[0] == 14? **NO!**

Move to the next element and ask is L[1] == the *target value*?



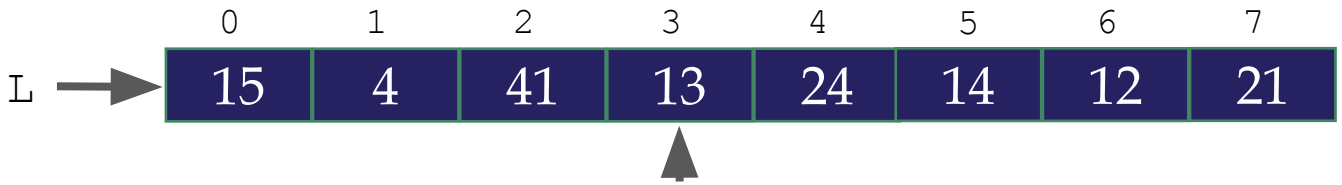
L[1] == 14? **NO!**

Move to the next element and ask is L[2] == the *target value*?



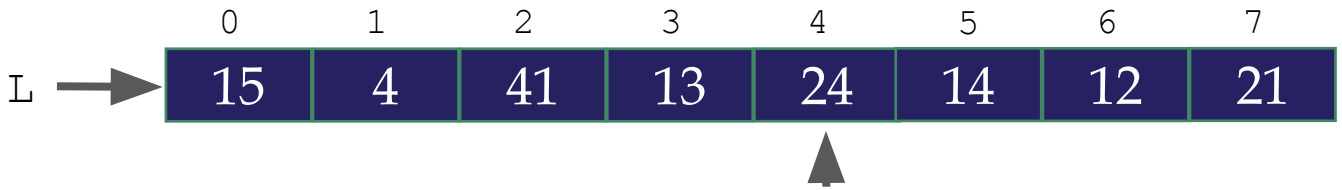
L[2] == 14? **NO!**

Move to the next element and ask is L[3] == the *target value*?



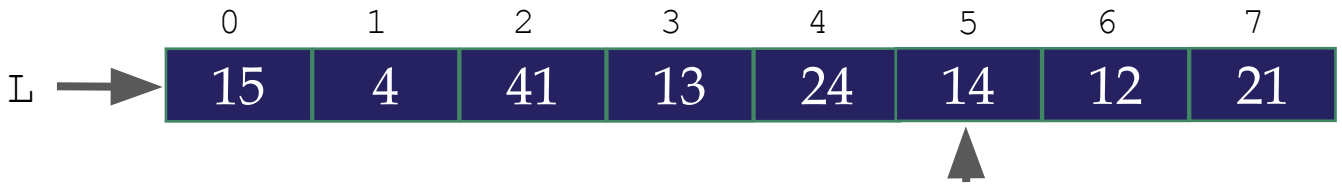
L[3] == 14? **NO!**

Move to the next element and ask is L[4] == the *target value*?

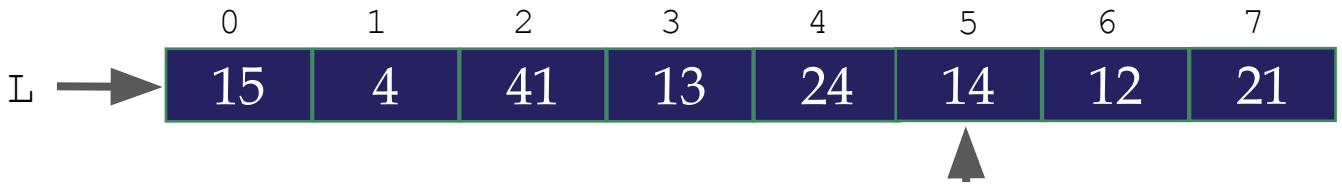


L[4] == 14? **NO!**

Move to the next element and ask is L[5] == the *target value*?



L[5] == 14? **YES!**



L[5] == 14? **YES!**

We have found the *target value* at index 5

The result of the search is 5

Group activity

PRIMM activity on linear search

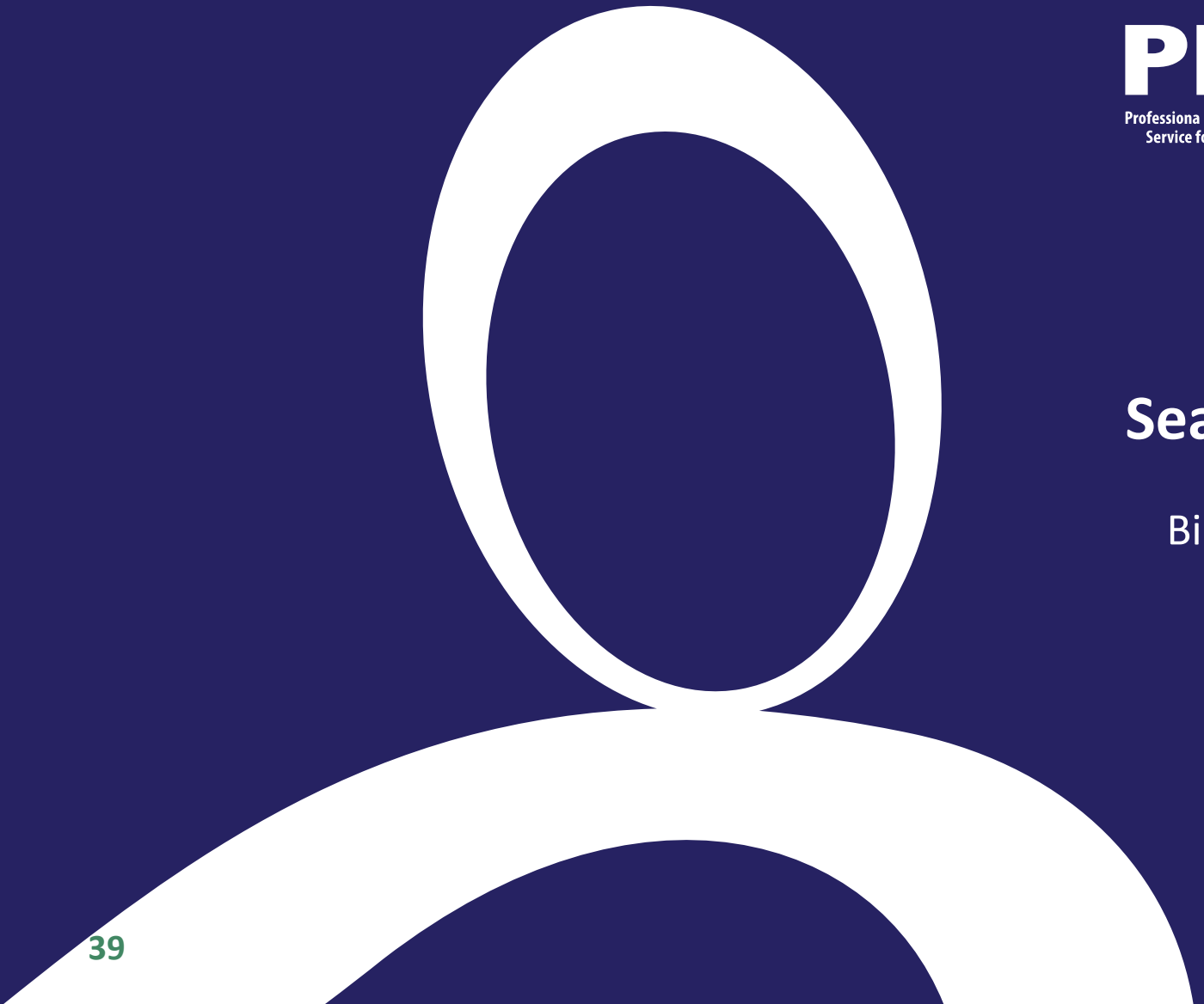
Linear Search

```
1. # A program to perform a linear search
2.
3. # linearSearch function
4. # takes two parameters: num is the target value, myList is the list to search
5. # if target value found, returns position (index) of 'num'; otherwise, returns
-1
6. def linearSearch(num, myList):
7.     for i in range(len(myList)):
8.         if myList[i] == num:
9.             return i
10.    return -1
11.
12. # PYTHON STARTS EXECUTING FROM HERE ...
13. listToSearch = [1,2,3,4,5,6,7,8,9,10]
14. userNum = int(input("Enter the number you want to find:"))
15.
16. # function call
17. position = linearSearch(userNum, listToSearch)
18.
19. # display the result of the linear search
20. print("The number is at position", position)
```

Breakout task

PRIMM activity on linear search

```
1. # A program to perform a linear search
2.
3. # linearSearch function
4. # takes two parameters: num is the target value, myList is the list to search
5. # if target value found, returns position (index) of 'num'; otherwise, returns  -1
6. def linearSearch(num, myList):
7.     for i in range(len(myList)):
8.         if myList[i] == num:
9.           return i
10.    return -1
11.
12. # PYTHON STARTS EXECUTING FROM HERE ...
13. listToSearch = [1,2,3,4,5,6,7,8,9,10]
14. userNum = int(input("Enter the number you want to find:"))
15.
16. # function call
17. position = linearSearch(userNum, listToSearch)
18.
19. # display the result of the linear search
20. print("The number is at position", position)
```



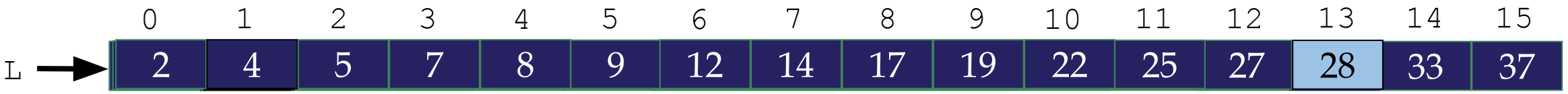
PDST
Professiona Deve opment Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúi do Mhúinteoirí

Searching algorithms

Binary search

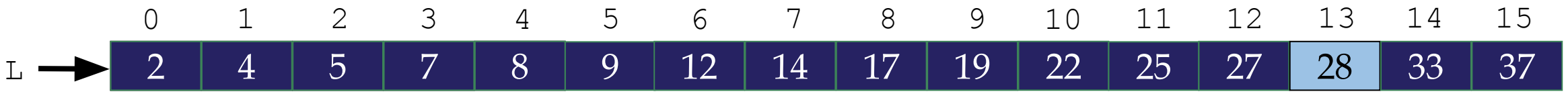
Input:

A list L and a target value of 28



Input:

A list L and a *target value* of 28

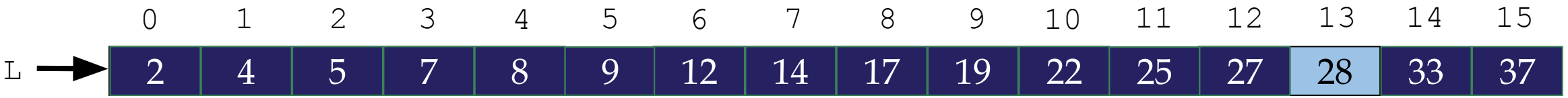


Required Output:

If the *target value* is found in L, its index

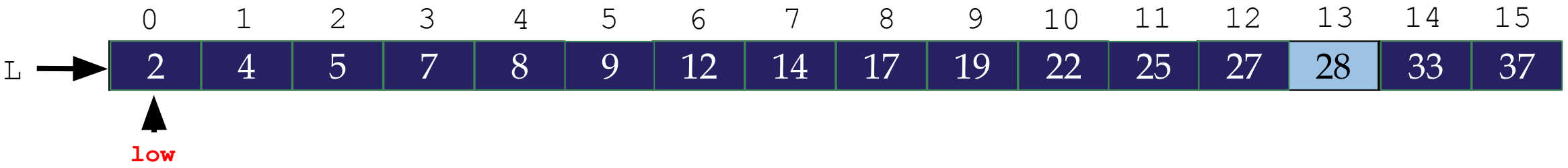
If the *target value* is not found in L, we return -1

Binary Search Algorithm



Pseudo-code: (target value is 28)

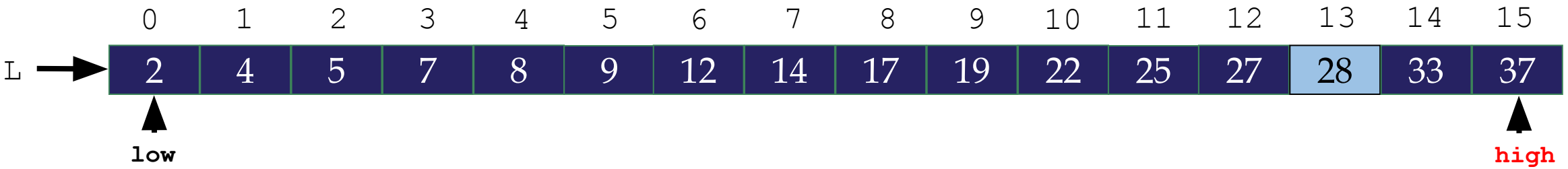
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0

Binary Search Algorithm

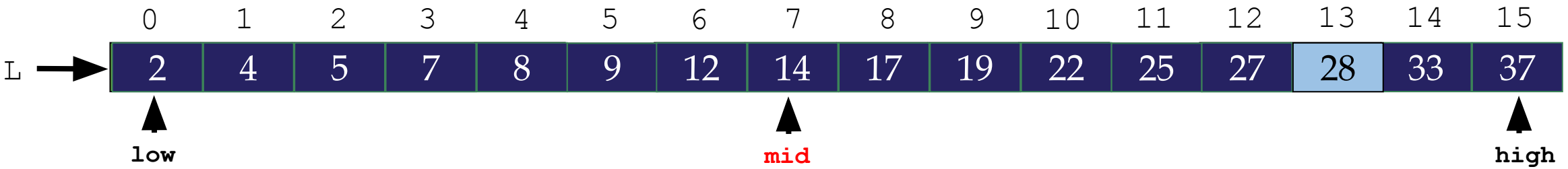


Pseudo-code: (target value is 28)

1. Set low = 0

2. Set high = length of list - 1

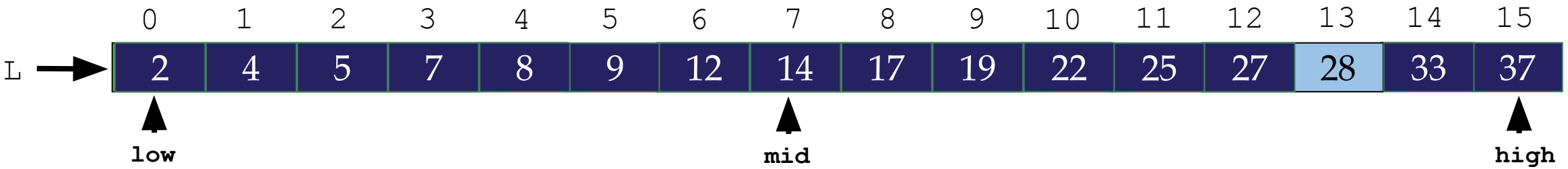
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. **Set low = 0**
2. **Set high = length of list - 1**
3. **Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer**

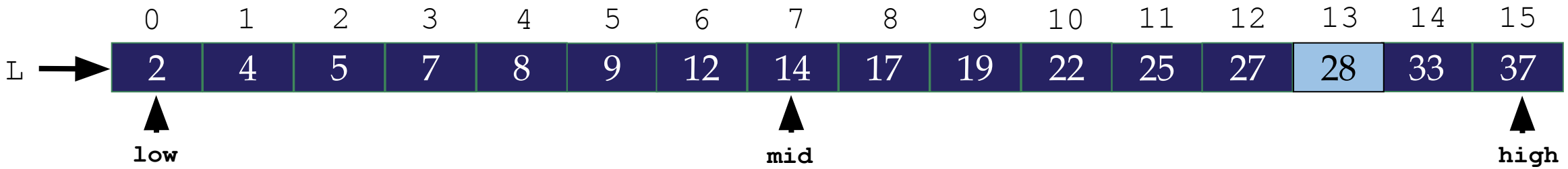
Binary Search Algorithm



Pseudo-code: **(target value is 28)**

1. **Set low = 0**
2. **Set high = length of list - 1**
3. **Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer**
4. **If the value at the mid position is the same as the target value**
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1

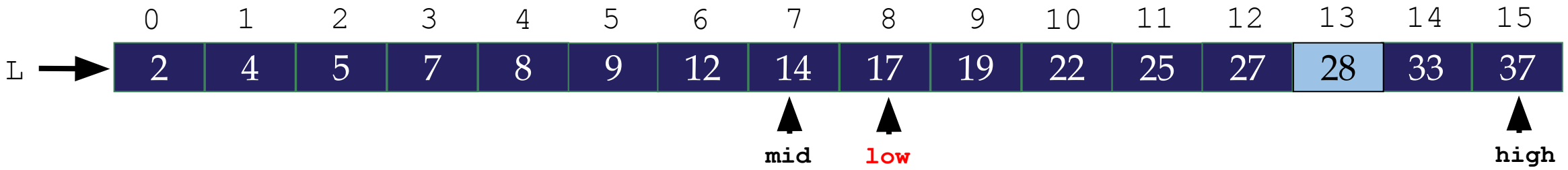
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1

Binary Search Algorithm



Pseudo-code: **(target value is 28)**

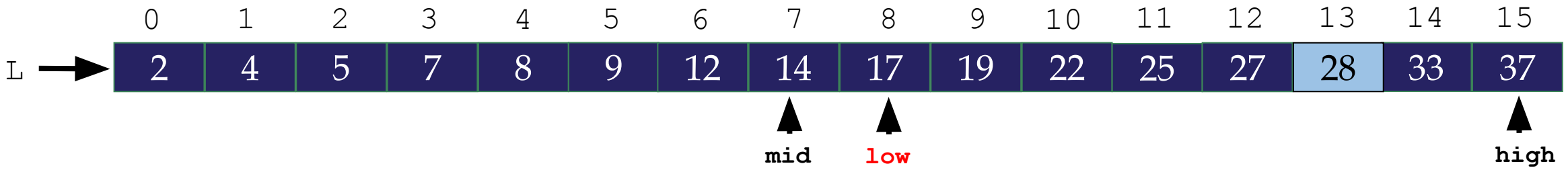
1. Set low = 0

2. Set high = length of list - 1

3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer

4. If the value at the mid position is the same as the target value
Return mid
Else If the value at the mid position is less than the target value
Set low = mid + 1
Else If the value at the mid position is greater than the target value
Set high = mid - 1

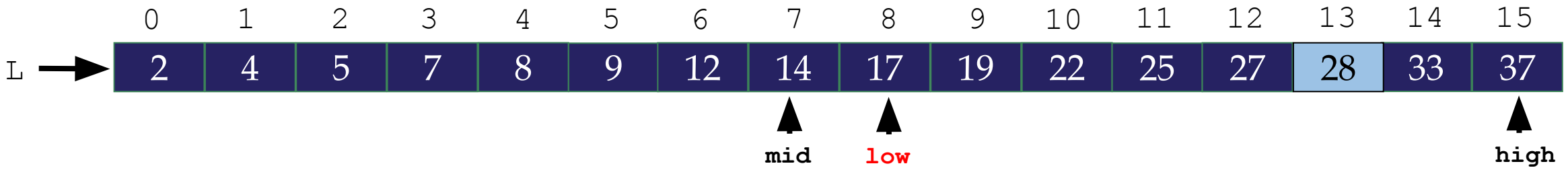
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

Binary Search Algorithm

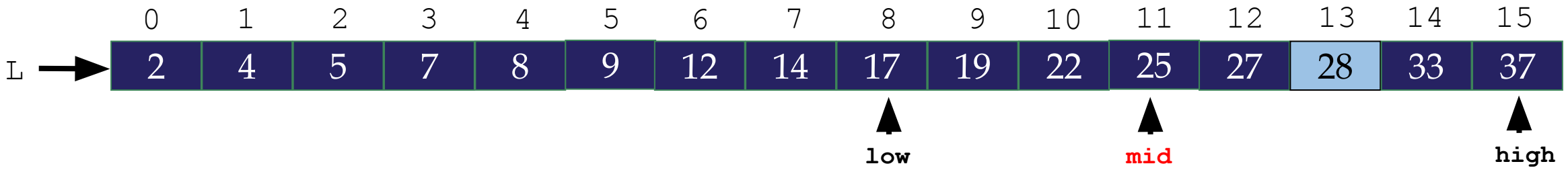


Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

In Python this means, `while low <= high:`

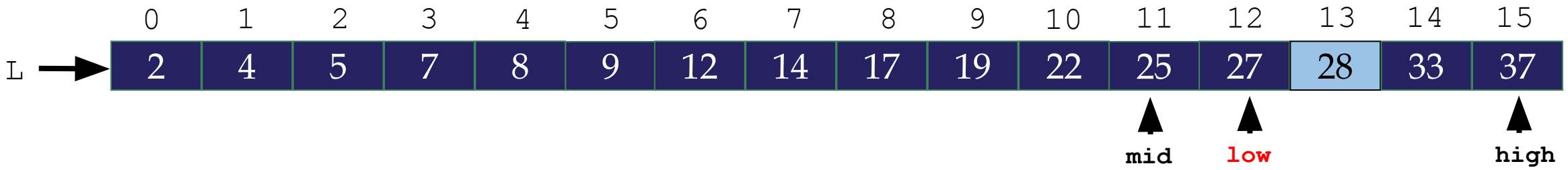
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

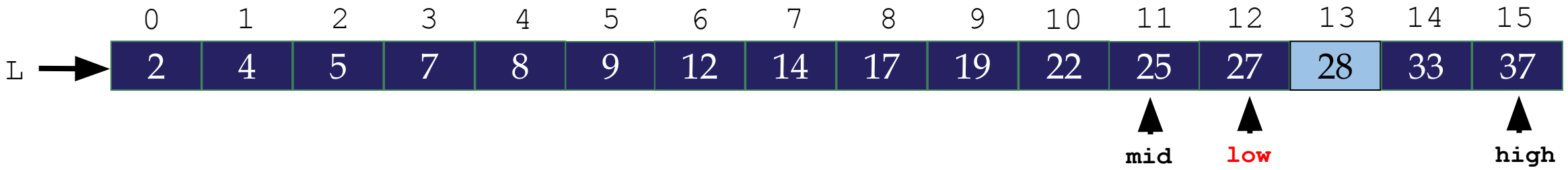
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

Binary Search Algorithm



Pseudo-code: **(target value is 28)**

1. Set low = 0

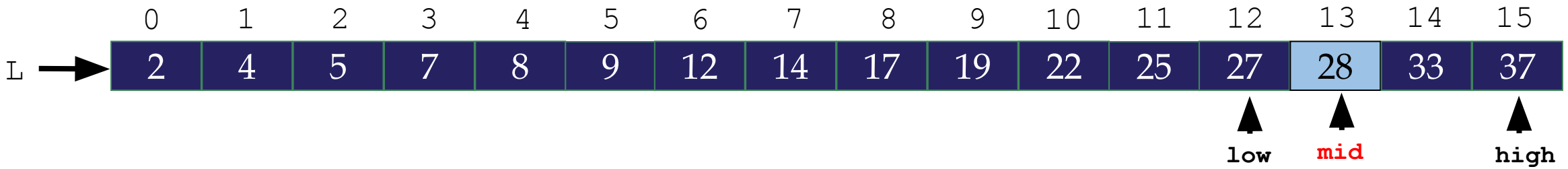
2. Set high = length of list - 1

3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer

4. If the value at the mid position is the same as the target value
Return mid
Else If the value at the mid position is less than the target value
Set low = mid + 1
Else If the value at the mid position is greater than the target value
Set high = mid - 1

5. As long as low doesn't 'cross over' high, go back to step 3 above

Binary Search Algorithm



Pseudo-code: **(target value is 28)**

1. Set low = 0

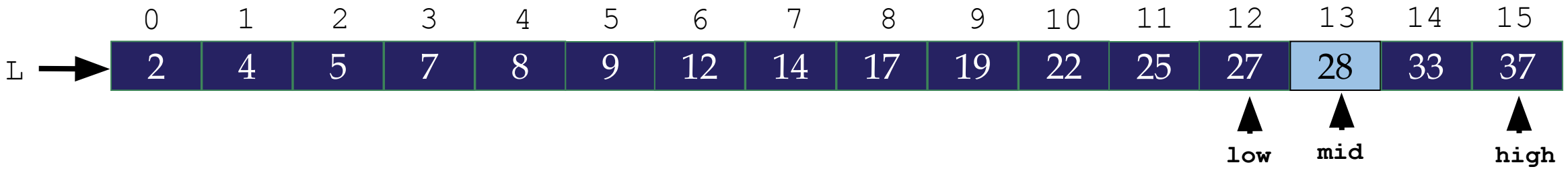
2. Set high = length of list - 1

3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer

4. If the value at the mid position is the same as the target value
Return mid
Else If the value at the mid position is less than the target value
Set low = mid + 1
Else If the value at the mid position is greater than the target value
Set high = mid - 1

5. As long as low doesn't 'cross over' high, go back to step 3 above

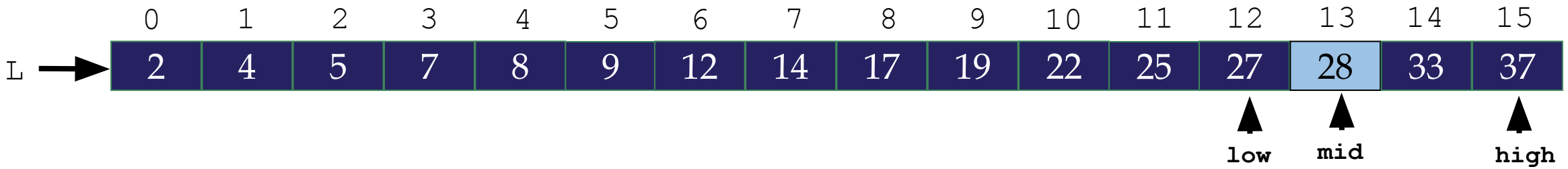
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

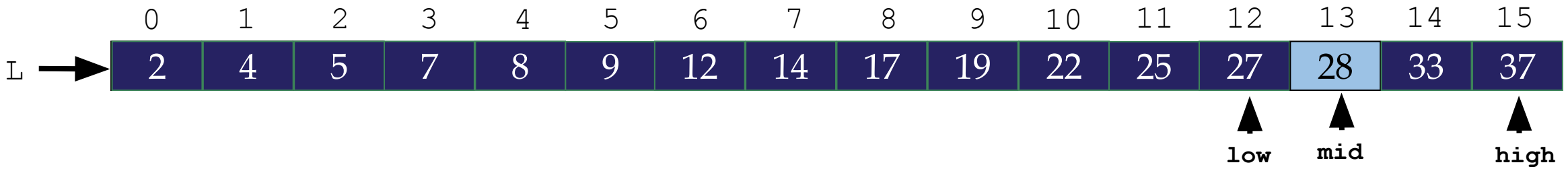
Binary Search Algorithm



Pseudo-code: (target value is 28)

1. Set low = 0
2. Set high = length of list - 1
3. Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer
4. If the value at the mid position is the same as the target value
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. As long as low doesn't 'cross over' high, go back to step 3 above

Binary Search Algorithm



Pseudo-code: **(target value is 28)**

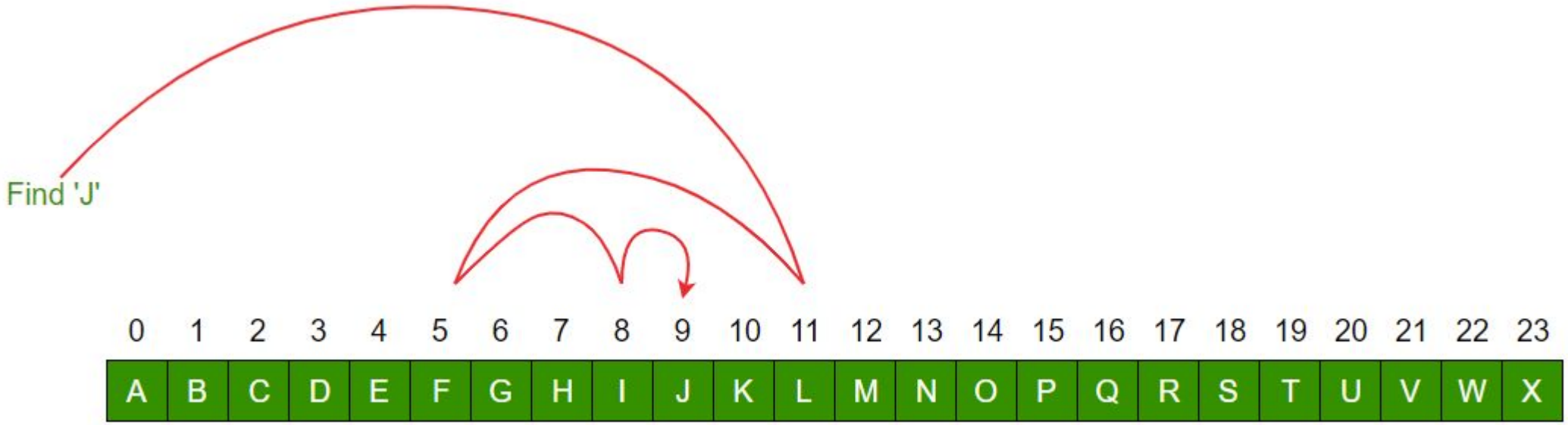
1. **Set low = 0**
2. **Set high = length of list - 1**
3. **Set mid = $\frac{\text{low+high}}{2}$, rounded down to an integer**
4. **If the value at the mid position is the same as the target value**
   Return mid
   Else If the value at the mid position is less than the target value
   Set low = mid + 1
   Else If the value at the mid position is greater than the target value
   Set high = mid - 1
5. **As long as low doesn't 'cross over' high, go back to step 3 above**

13 is returned (as it is the value of mid)
This is the index of the target element.

Q. How many comparisons were needed?

Q. How many comparisons would be needed for the linear search?

Another Example



Lunch

An Roinn Oideachais
Department of Education