# Introduction to Algorithms

What is an algorithm?

What is computer science?

What are some features of algorithms studied by computer scientists?

What is the **search problem**?

How does the built-in list index function work? How does it signal a value was not found?

What is the **idea** of linear search?

How can linear search be **implemented** in Python?

Graphical user interface, text, application

Description automatically generated

Does the order of the elements in lst matter to linear\_search?

What is reverse linear search? When is different than regular linear search? Why might you use it?

Graphical user interface, text, application

Description automatically generated

What are some other ways you could implement linear search?

What is binary search? What does it assume about the list it’s searching?

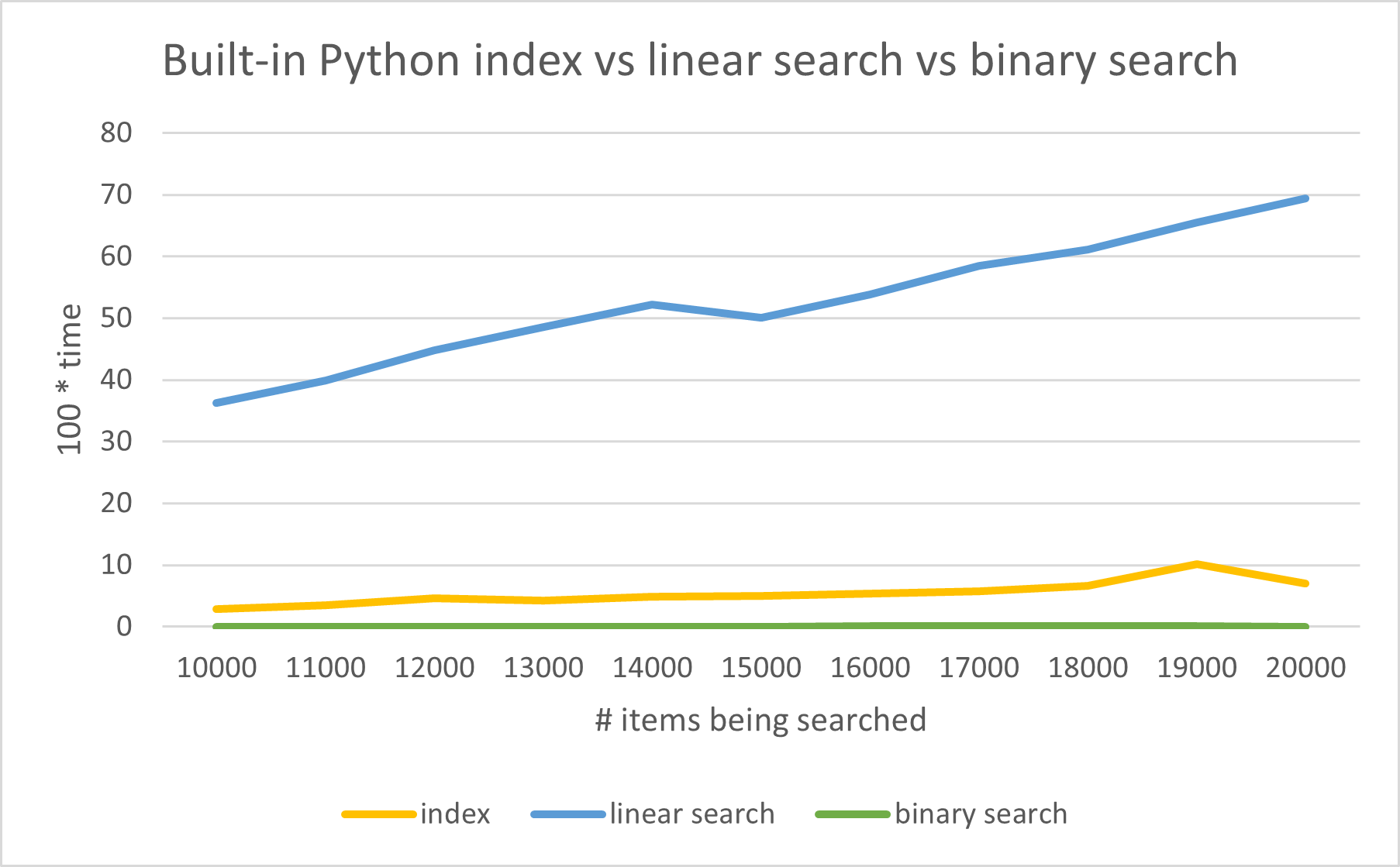
How would binary search find 5 in the list [0, 2, 3, 4, 8, 9, 9] ?

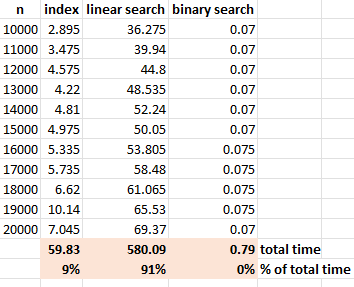
How can you implement binary search in Python?

Text

Description automatically generated

How does the performance of linear search and binary search compare?





Why do computer scientists often **not** use **actual time** when analyzing the theoretical performance of an algorithm?

What is a key instruction? What is the usual key instruction for search algorithms?

If you do **linear search** on a list of 100 elements, what is:

* The **best case** number of comparisons?
* The **worst case** number of comparisons?
* The **average case** number of comparisons?

If you do **linear search** on a list of n elements, what is:

* The **best case** number of comparisons?
* The **worst case** number of comparisons?
* The **average case** number of comparisons?

Which of the three cases do we usually assume practice?

If you do **binary search** on a list of n elements, what is:

* The **best case** number of comparisons?
* The **worst case** number of comparisons?
* The **average case** number of comparisons?

How much smaller is compared to ?

|  |  |
| --- | --- |
| n |  |
| 16 | 4 |
| 32 | 5 |
| 64 | 6 |
| 128 | 7 |
| 1048576 | 20 |

In the worst case, about how many comparisons does binary search do on a list of length 1 million?

## Sorting

What is the **sorting problem**?

How does Python’s built-in list sort function work?

What is **selection sort**? How does it work?

How can you implement selection sort in Python?

Text

Description automatically generated

What are some ways selection\_sort differs from the built-in list sort?

What is merging? How would you merge these two lists together:  
  
[1, 3, 6, 9, 11]  
  
  
  
[0, 4, 8, 9]

How can we merge two lists in Python?

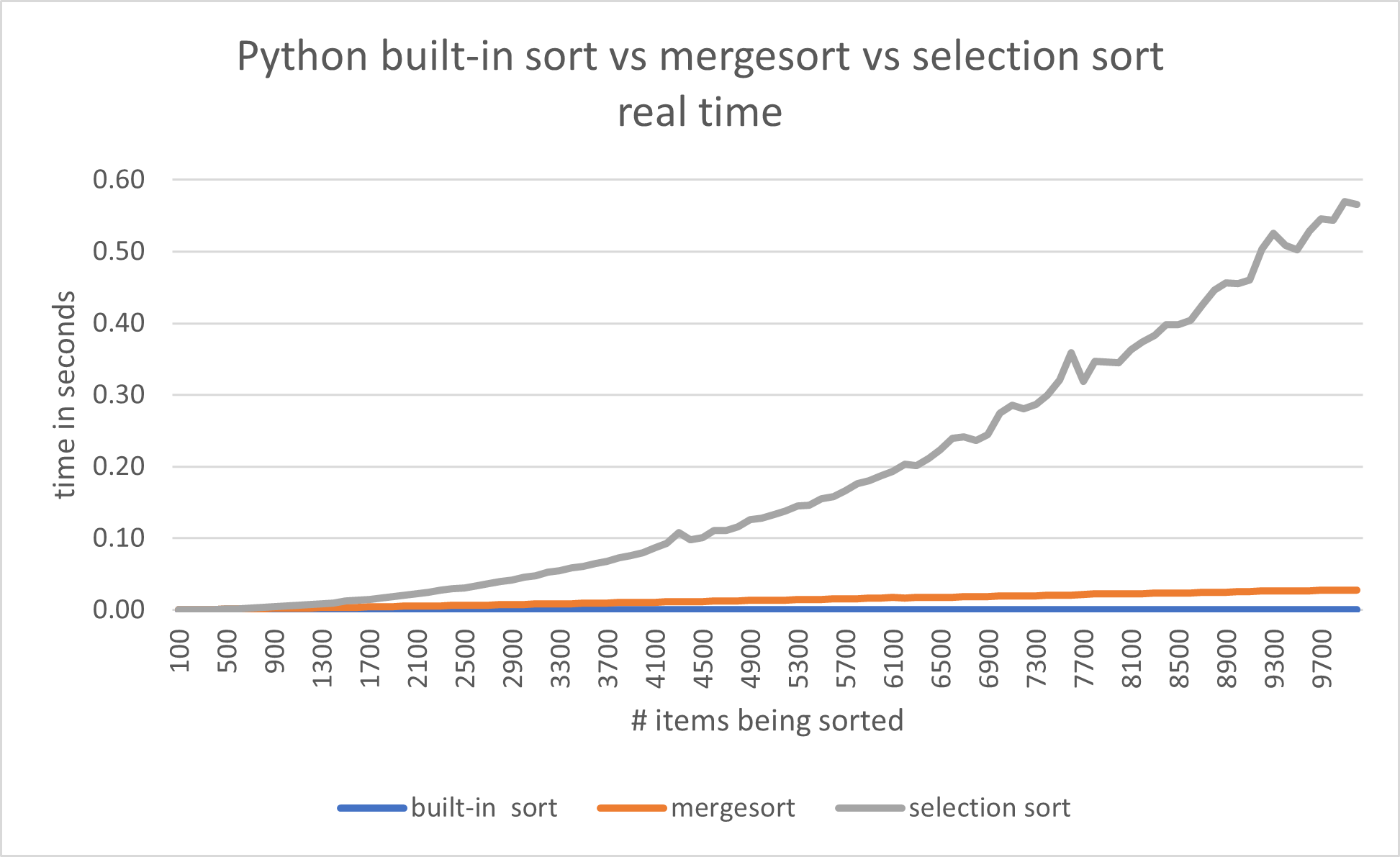
Text

Description automatically generated

What is **mergesort**? How could you use it to sort this list:

[9, 5, 2, 8, 1, 2, 7]

How do the running times of built-in list sort, selection sort, and mergesort compare?



The time it takes for **selection sort** to sort n items is proportional to what mathematical expression?

The time it takes for **mergesort** to sort n items is proportional to what mathematical expression?

Suppose it takes 10 seconds for **selection sort** to sort n items. About how long would it take to sort 2n items? 3n items?

Suppose it takes 10 seconds for **mergesort** to sort n items. About how long would it take to sort 2n items?

What kind of sorting algorithm is Python’s built-in list sort?

In practice, which sorting algorithm should you usually use in Python?