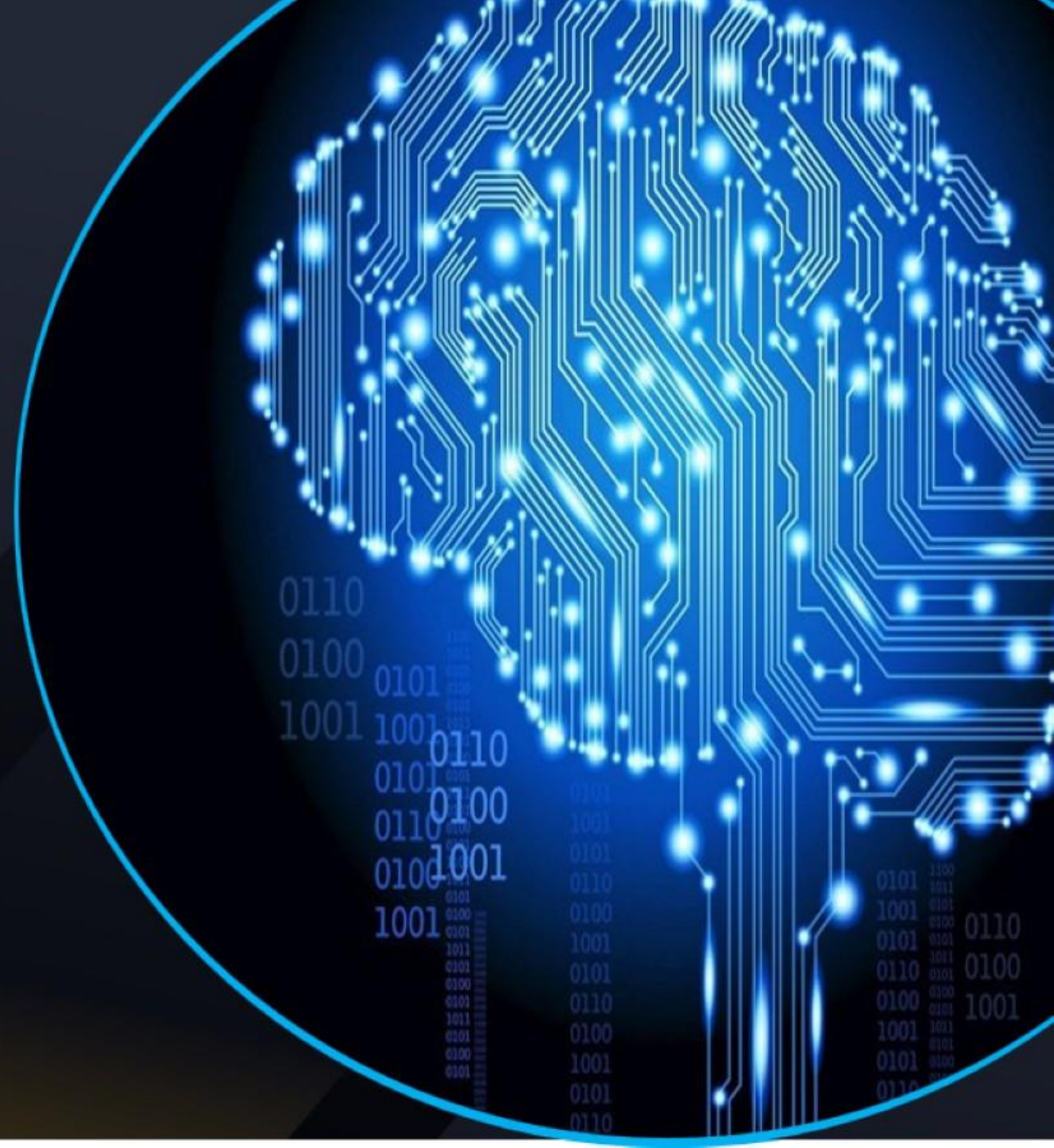


Discrete Mathematics

Dr. Shahira Ahmed

IT241

Lecture 9



Discrete Mathematics

IT241

Lecture 9 : Algorithms

Dr. Shahira Ahmed



LEARNING OUTCOMES

- ◉ Understand the fundamental concepts and importance of algorithms in computer science and problem-solving.
- ◉ Implement and analyze sorting and searching algorithms, including bubble sort, insertion sort, selection sort, merge sort, and binary search.
- ◉ Evaluate and compare the efficiency and effectiveness of different algorithms for solving specific problems.



LECTURE OUTLINE

- ◉ Algorithms
 - ◉ Properties of Algorithms
 - ◉ Algorithms for Searching and Sorting
- ◉ Complexity of Algorithms

- In many domains there are key **general problems** that ask for output when given valid input.
- The first step is to **precisely state the problem**, using the appropriate structures to specify the **input** and the desired **output**.
- We then solve the general problem by specifying **the steps of a procedure that takes a valid input and produces the desired output**. This procedure is called an ***algorithm***.

Definition: An *algorithm* is a **finite** set of **precise instructions** for performing a computation or for **solving a problem**.

- **“Step by Step procedure for solving a problem”**

- The scientist, astronomer and mathematician Muḥammad ibn Musa **Al-Khwarizmi** ““father of algebra” ”, formerly Latinized as **Algorithmi**, under the patronage of the Caliph Al-Ma'mun of the Abbasid Caliphate. Around 820 AD he was appointed as the astronomer and head of the library of the House of Wisdom in Baghdad.



- Al-Khwarizmi's in his book, “The Compendious Book on Calculation by Completion and Balancing”, presented the first systematic solution of linear and quadratic equations. الكتاب المختصر في حساب الجبر والمقابلة
- He wrote several influential books “On the Calculation with Hindu Numerals”. The original manuscript was lost, though a Latin translation from the 1100’s introduced this number system to Europe. The old French word algorisme meant “the Arabic numerals system”, and only later did it come to mean a general recipe for solving computational problems.





- Input



- Steps



- Output

- Algorithms can be specified in English or in *pseudocode*.
- Pseudocode is **an intermediate step between** an **English language description** of the steps and a coding of these steps using a programming language.
- It uses some of the structures found in popular languages such as **python** , C++ and Java.

- ***Input***: An algorithm has input values from a specified set.
- ***Output***: From the input values, the algorithm produces the output values from a specified set. The output values are the solution.
- ***Correctness***: An algorithm should produce the correct output values for each set of input values.
- ***Finiteness***: An algorithm should produce the output after a finite number of steps for any input.
- ***Effectiveness***: It must be possible to perform each step of the algorithm correctly and in a finite amount of time.
- ***Generality***: The algorithm should work for all problems of the desired form.

Definition: The general *searching problem* is to locate an element x in the list of distinct elements $a_1, a_2, \dots, a_n$, or determine that it is **not in the list**.

- We will study two different searching algorithms;
 - ❑ **Linear search**
 - ❑ **Binary search.**

Linear vs Binary Search

- <https://www.youtube.com/watch?v=YzT8zDPihmc>

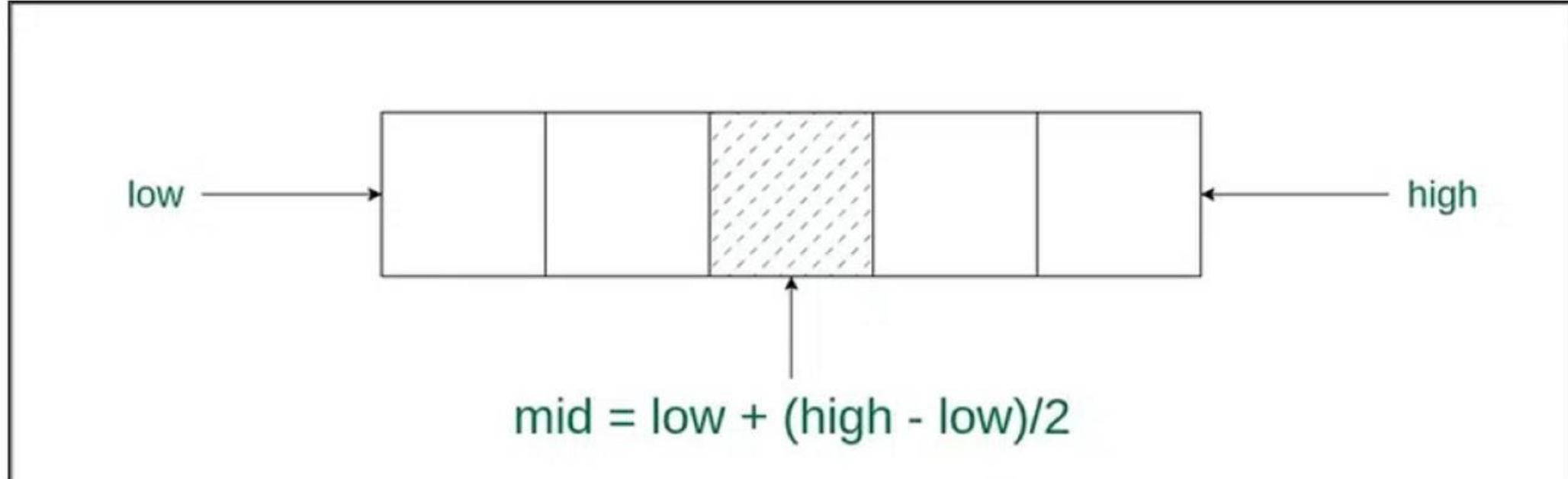
- The linear search algorithm locates an item x in a list $a_1, a_2, \dots, a_n$:
 - **compare** x with a_1 . If they are **equal**, return the position 1.
 - If not, **try** a_2 . If $x = a_2$, return the position 2.
 - Keep going, if it **is not found** , return **not found**.

OR

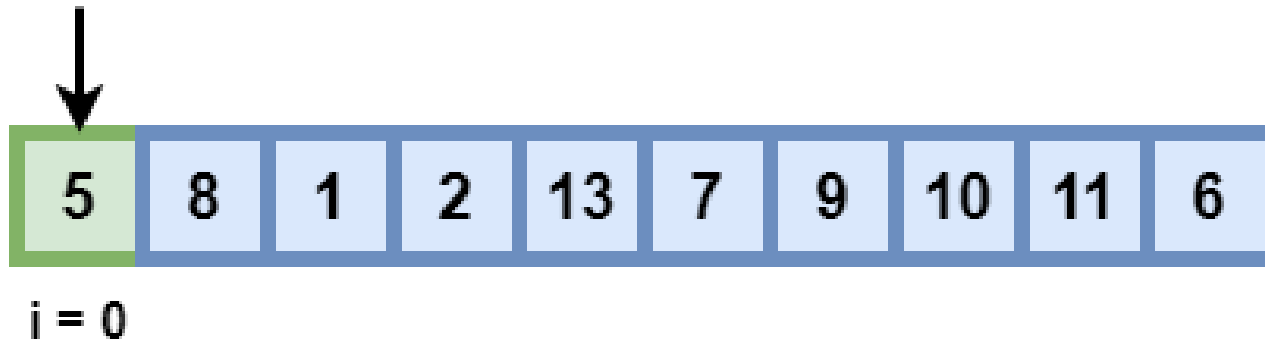
```
def search(arr, x):  
    for i in range(0, len(arr)):  
        if (arr[i] == x):  
            return i  
    return -1
```

- The input is a sorted list of items in increasing order, an element x .
- Divide the search space into two halves by finding the middle index “mid”.
- Compare the middle element of the search space with the x .
 - If the x = middle element, the process is terminated.
 - If the x < middle element, then the left side is used for next search.
 - If the x > middle element, then the right side is used for next search.
- This process is continued until the x is found or the total search space is exhausted.

Sorted



Value to Search = 10



`arr[i] == 10`

FALSE

Linear Search vs Binary Search

<https://www.cs.usfca.edu/~galles/visualization/Search.html>

Binary Search

Search for 47

0	4	7	10	14	23	45	47	53
---	---	---	----	----	----	----	----	----

- **Linear** Search: How many **iterations** in 1024 element array?
- **Binary** Search: How many **iterations** in 1024 element array?
- Solution: in worst-case scenario
 - 1024
 - 10

Linear vs Binary Search

TARGET: 9

BINARY SEARCH



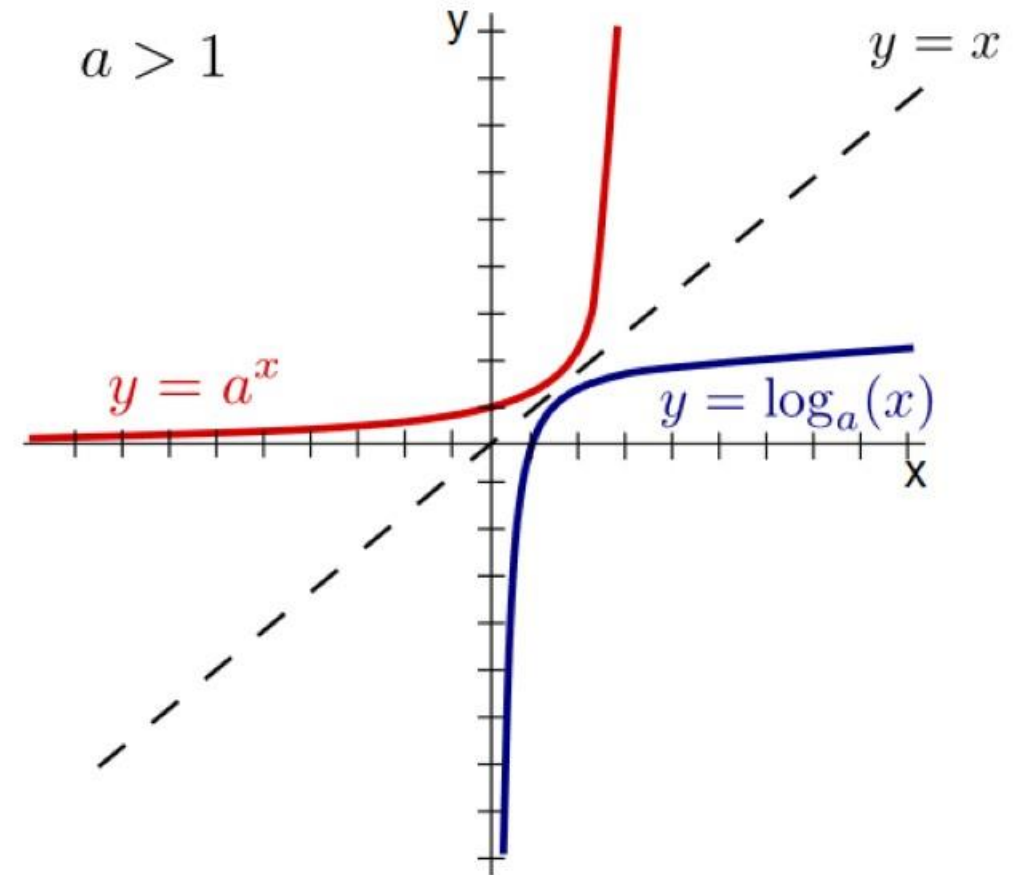
TARGET: 9

LINEAR SEARCH



Logarithm function

- $2^3 = 8 \implies \log_2 8 = 3$



Worst-case → the **largest** number of operations needed to solve the given problem using this algorithm it will produce a solution.

Best case → the **smallest** number of operations needed to solve the given problem using this algorithm it will produce a solution.

- a) *Best case scenario for linear search*
- b) *Best case scenario for binary search*
- c) *Worst case scenario for linear search*
- d) *Worst case scenario for binary search*

Solution a) First element b) middle element

c) last element(not found) d) first or last element(not found)

- To sort the elements of a list is to put them in increasing order (numerical order, alphabetic, and so on).
 - Bubble
 - Selection
 - Insertion
 - Merge

Bubble sort

6 5 3 1 8 7 2 4

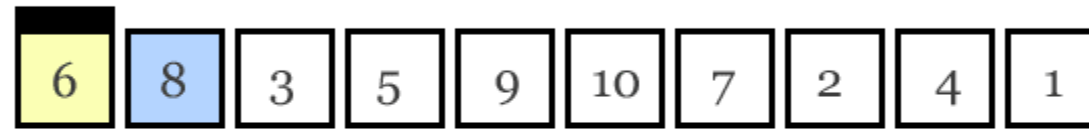
<https://upload.wikimedia.org/wikipedia/commons/c/c8/Bubble-sort-example-300px.gif>

- <https://www.youtube.com/watch?v=14La4BKNoqc>

- *Bubble sort* makes multiple passes through a list. Every pair of elements that are found to be out of order are interchanged.

```
procedure bubblesort( $a_1, \dots, a_n$ : real numbers  
                    with  $n \geq 2$ )  
  for  $i := 1$  to  $n - 1$   
    for  $j := 1$  to  $n - i$   
      if  $a_j > a_{j+1}$  then interchange  $a_j$  and  $a_{j+1}$   
  { $a_1, \dots, a_n$  is now in increasing order}
```

Selection Sort



Yellow is smallest number found

Blue is current item

Green is sorted list

Selection Sort

<https://www.youtube.com/watch?v=0-W8OEwLebQ>

Insertion Sort



hobbyartandcraft

<https://blog.devgenius.io/insertion-sort-algorithm-c8c6aee65c8c>

Step 1 : Divide this array into two part :- Sorted Array and Unsorted Array 👍

Note : First element of the array is always sorted 😊

<https://blog.devgenius.io/insertion-sort-algorithm-c8c6aee65c8c>

- **Step 2** : Store the first element of unsorted array in a constant (Name of constant : key)



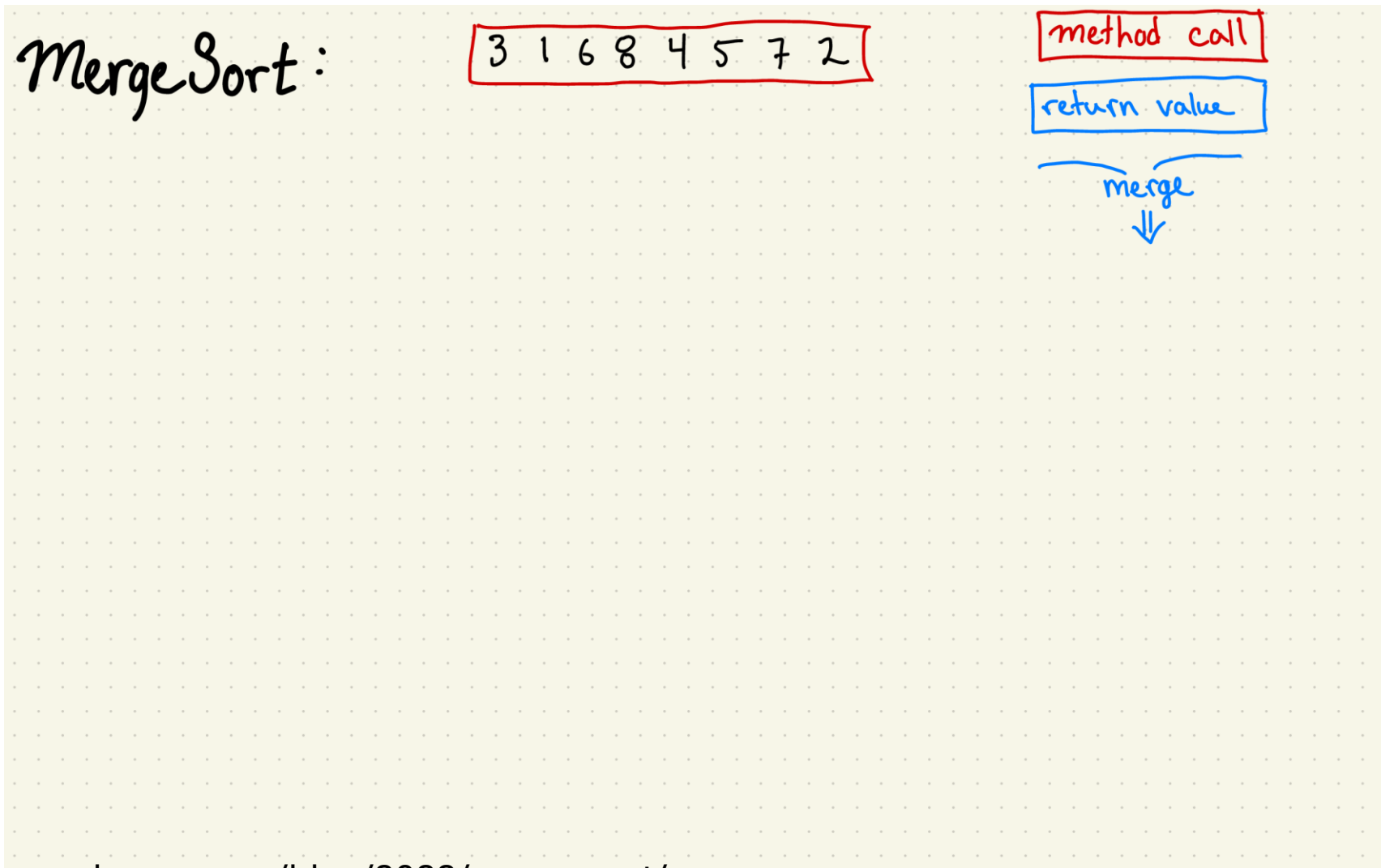
Insertion Sort



Insertion Sort

<https://www.youtube.com/watch?v=EdIKlf9mHk0>

Merge Sort



<https://willrosenbaum.com/blog/2022/merge-sort/>

Merge Sort

<https://www.youtube.com/watch?v=dENca26N6V4>

- Using Timer to measure efficiency of a program



<https://www.pinterest.com/pin/708261478892553559/>

Use your stopwatch

```
2
3 start = timer()
4 sum=0
5 for i in range(1000000):
6     sum+=i
7 end = timer()
8 print("Start time:", start)
9 print("End time:",end)
10 print("-----")
11 print("Time of Running : ",(end - start)*1000)
12
```

Start time: 212.083850562

End time: 212.211368037

Time of Running : 0.127517474999999055

TypeError Traceback (most recent call last)
<ipython-input-9-33f1de0e891d> in <cell line: 11>()
 9 print("End time:",end)
 10 print("-----")
--> 11 print("Time of Running : ",(end - start))*1000

TypeError: unsupported operand type(s) for *: 'NoneType' and 'int'

Why?

1. Hardware
2. Input Data
3. Compiler
4. Operating system

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

The **number of operations or steps** the algorithm will take to complete its task



Recap

1. In **Linear** Search: How many **iterations** in 32 element array?
2. In **Binary** Search: How many **iterations** in 32 element array?
3. In **Linear** Search: How many **iterations** in 32768 element array?
4. In **Binary** Search: How many **iterations** in 32768 element array?
5. Which algorithm is more **efficient** to search for an element in an **unsorted list**?
6. Which algorithm is more **efficient** to search for an element in an **sorted list**?
7. Which algorithm requires the list to be **sorted** prior to searching?

Solution

1. 32 2. 5 3. 32768 4. 15 5. Linear 6. Binary 7. Binary

1. _____ “Step by Step procedure for solving a problem”
2. _____ Sort: The largest element always bubbles up to the end of the array in each iteration
3. _____ Sort :The smallest element always bubbles up to the end of the array in each iteration
4. _____ sort: divides the array into two halves, sorts them separately, and then merges them.
5. (T/F)Time complexity of an algorithm measure: The actual execution time of the algorithm

Solution 1. Algorithm 2. Bubble 3. Selection 4. Merge

5.False→The number of operations or steps the algorithm takes to complete task

- 1. In insertion sort, each element is compared and inserted into its correct position in the**
 - a) Sorted portion of the array**
 - b) Unsorted portion of the array**
 - c) Middle of the array**
 - d) End of the array**
- 2. _____ The algorithm's growth rate as the input size increases**

Solution 1. a

2. Time Complexity

- Which is better : Linear Search or Binary Search
 1. Worst case scenario, sorted array
 2. Best case scenario, sorted array
 3. Worst case scenario, unsorted array
 4. Best case scenario, unsorted array

Solution

1. Binary 2.Linear 3. Linear 4. Linear

- In the bubble sort [10,22,3]
 - There is a swap between 10,22
 - In the second path (iteration) ,There is a swap between 22,3
- In the bubble sort [300,212,3]
 - There is a swap between 300,212
 - In the second path (iteration), There is a swap between 3,300
- Solution
 - No, yes
 - Yes , Yes

1. how many comparison to find 5 in the list [2,12,444,5] using Linear Search
2. how many comparison to find 5 in the list [2,12,444,5] using Binary Search
3. how many comparison to find 5 in the list [2,3,5,7,8] using Linear Search
4. how many comparison to find 5 in the list [2,3,5,7,8] using Binary Search

Solution

1. 4 2. We cannot use binary search 3. 3 4. 1

Choose sort algorithm(part of the iterations)

- 3, 1, 5, 7, 4,
 - at the end of the first pass: 1, 3, 5, 4, 7;
 - at the end of the second pass: 1, 3, 4, 5, 7;
 - at the end of the third pass: 1, 3, 4, 5, 7;
 - at the end of the fourth pass: 1, 3, 4, 5, 7
-
- Solution: Bubble

Choose sort algorithm(part of the iterations)

- 1, 5, 4, 3, 2;
- At the end of the first pass 1, 2, 4, 3, 5;
- At the end of the second pass 1, 2, 3, 4, 5;
- At the end of the third pass 1, 2, 3, 4, 5

Solution: Selection

Choose sort algorithm(part of the iterations)

- 3, 2, 4, 1, 5
 - 2, 3, 1, 4, 5.
 - 2, 1, 3, 4, 5.
 - 1, 2, 3, 4, 5.
 - 1, 2, 3, 4, 5.
-
- Bubble

Choose sort algorithm(part of the iterations)

- 3, 1, 5, 7, 4,
- At the end of the first passes: 1, 3, 5, 7, 4;
- At the end of the second passes: 1, 3, 5, 7, 4;
- At the end of the third passes: 1, 3, 5, 7, 4;
- at the end of the fourth pass: 1, 3, 4, 5, 7

Solution Insertion

Choose sort algorithm(part of the iterations)

- 3, 5, 4, 1, 2
 - At the end of the 1 pass 1, 5, 4, 3, 2;
 - At the end of the 2 pass 1, 2, 4, 3, 5;
 - At the end of the 3 pass 1, 2, 3, 4, 5;
 - At the end of the 4 pass 1, 2, 3, 4, 5
-
- Solution: Selection

Choose sort algorithm(part of the iterations)

- 3, 2, 4, 1, 5
- At the end of the 1 pass 2, 3, 4, 1, 5;
- At the end of the 2 pass 2, 3, 4, 1, 5 ;
- At the end of the 3 pass 1, 2, 3, 4, 5;
- At the end of the 4 pass 1, 2, 3, 4, 5.

- Solution: Insertion



Thank You