Algorithms & Data Structures

Taibah University

Dr. Sameer M. Alrehaili

College of Science and Computer Engineering, Yanbu

Course Overview:

Data structures and algorithms are fundamental to programming and to understanding computation. The purpose of this module is to provide students with a coherent introduction to techniques for using data structures and commonly used algorithms for solving problems. The course is taught using the Java or Python programming language.

Course Overview:

Having successfully completed this course, the student will be able to:

1. Develop an appreciation of the relationship between data structures and algorithms.
2. Examine and experiment a variety of techniques for designing algorithms.
   a. To help you to estimate the running time, T(n).
   b. To help you to write an efficient algorithm.
   c. Compare the running time for two algorithms.
   d. To analyse an algorithm.
3. Select and implement data structures for a given problem.
4. Distinguish, differentiate and experiment different searching and sorting algorithms.
5. Explore the concept of an abstract data types (ADT) and the tradeoffs between different implementations of ADTs.

Topics Covered:

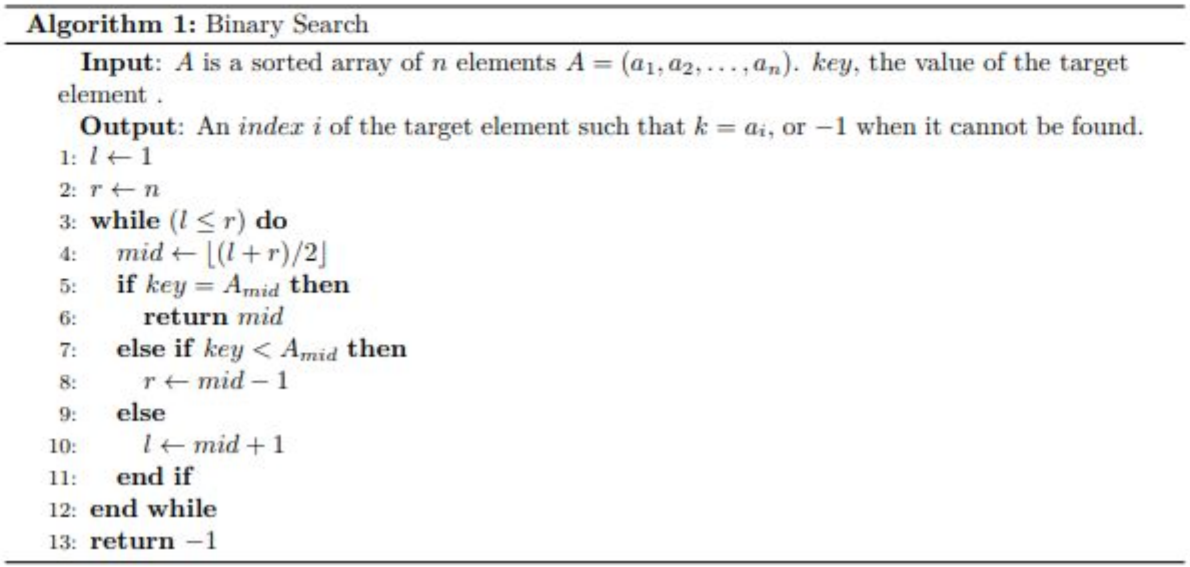
- Introduction Algorithms & Data Structures
- Algorithm Analysis
- Recursion
- Arrayes, Dynamic Arrays
- Stacks, and Queues
- Linked Lists
- Trees
- Graphs
- Hashing
- Searching Algorithms
- Sorting Algorithms

Algorithms & data structures

- An **algorithm** is “a **finite** sequence of instructions, each of which has a **clear meaning** and can be performed with a finite amount of effort in a finite length of time”.
- **Data structure** is a particular way of organising data for particular types of operation.

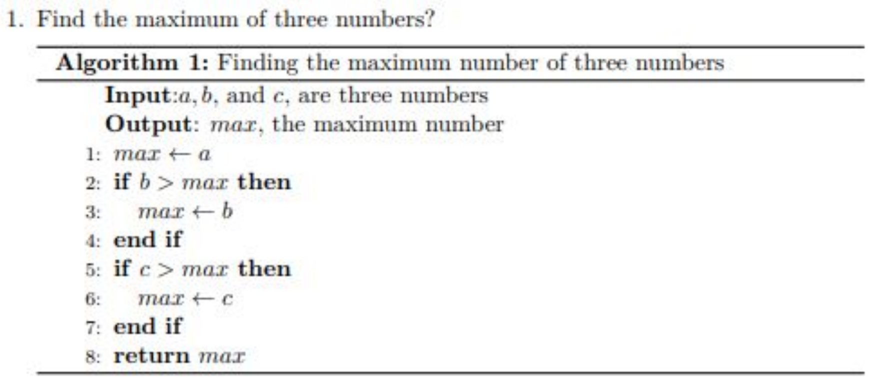
Pseudo-code

- Mixture of **Natural language** and **Programming language.**



```
Algorithm 1: Binary Search
Input: $A$ is a sorted array of $n$ elements $A = (a_1, a_2, \ldots, a_n)$. $key$, the value of the target element .
Output: An $index$ $i$ of the target element such that $k = a_i$, or $-1$ when it cannot be found.
$l \leftarrow 1$
$r \leftarrow n$
while $(l \leq r)$ do
    $mid \leftarrow \lfloor (l + r)/2 \rfloor$
    if $key = A_{mid}$ then
        return $mid$
    else if $key < A_{mid}$ then
        $r \leftarrow mid - 1$
    else
        $l \leftarrow mid + 1$
    end if
end while
return $-1$
```

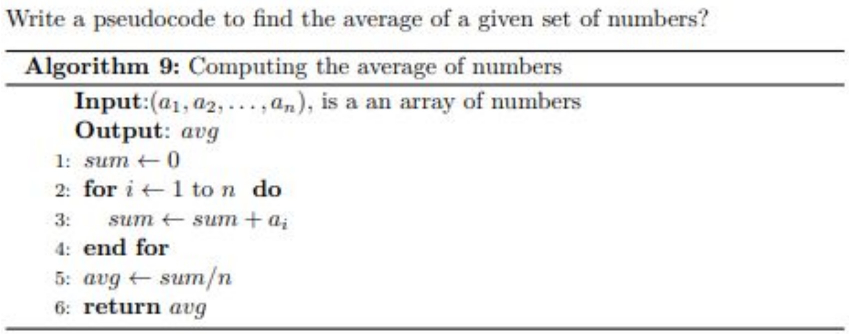
More examples



1. Find the maximum of three numbers?

Algorithm 1: Finding the maximum number of three numbers

```
**Input**:$a, b$, and $c$, are three numbers
**Output**: $max$, the maximum number
$max \leftarrow a$
**if** $b > max$ **then**
    $max \leftarrow b$
**end if**
**if** $c > max$ **then**
    $max \leftarrow c$
**end if**
**return** $max$
```



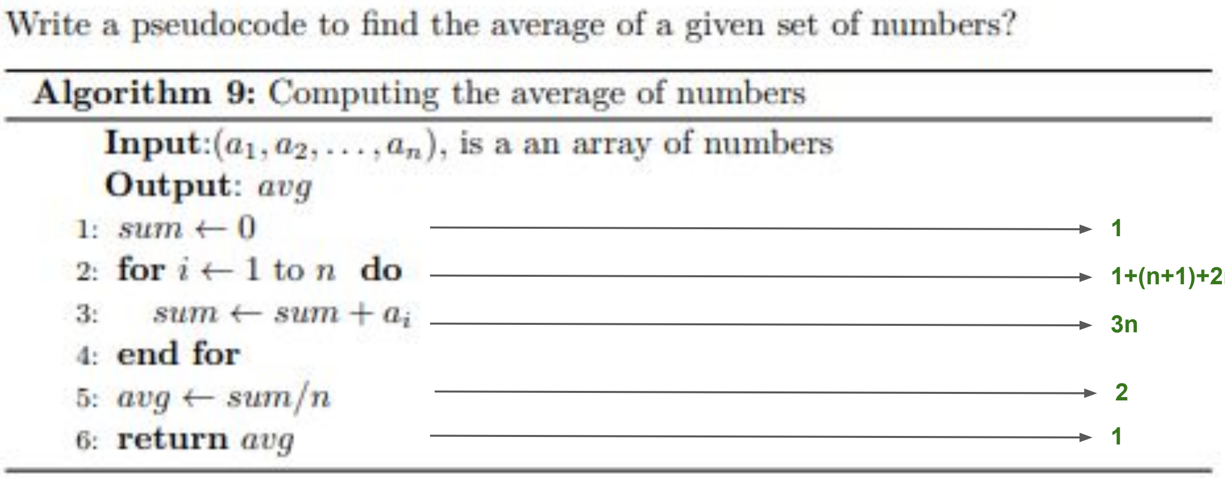
Write a pseudocode to find the average of a given set of numbers?

Algorithm 9: Computing the average of numbers

```
**Input**:$(a_1, a_2, \ldots, a_n)$, is a an array of numbers
**Output**: $avg$
$sum \leftarrow 0$
**for** $i \leftarrow 1$ to $n$ **do**
    $sum \leftarrow sum + a_i$
**end for**
$avg \leftarrow sum/n$
**return** $avg$
```

Algorithm Analysis

- Running time.
  - Same environment needed.
- Count the primitive operations used in an algorithm.
  - Independent from the hardware.
  - Analyse an algorithm without running it.



Write a pseudocode to find the average of a given set of numbers?

Algorithm 9: Computing the average of numbers

```
Input:$(a_1, a_2, \ldots, a_n)$, is a an array of numbers
Output: $avg$
$sum \leftarrow 0$
for $i \leftarrow 1$ to $n$ do
    $sum \leftarrow sum + a_i$
end for
$avg \leftarrow sum/n$
return $avg$
```

1

1+(n+1)+2n = 3n+2

3n

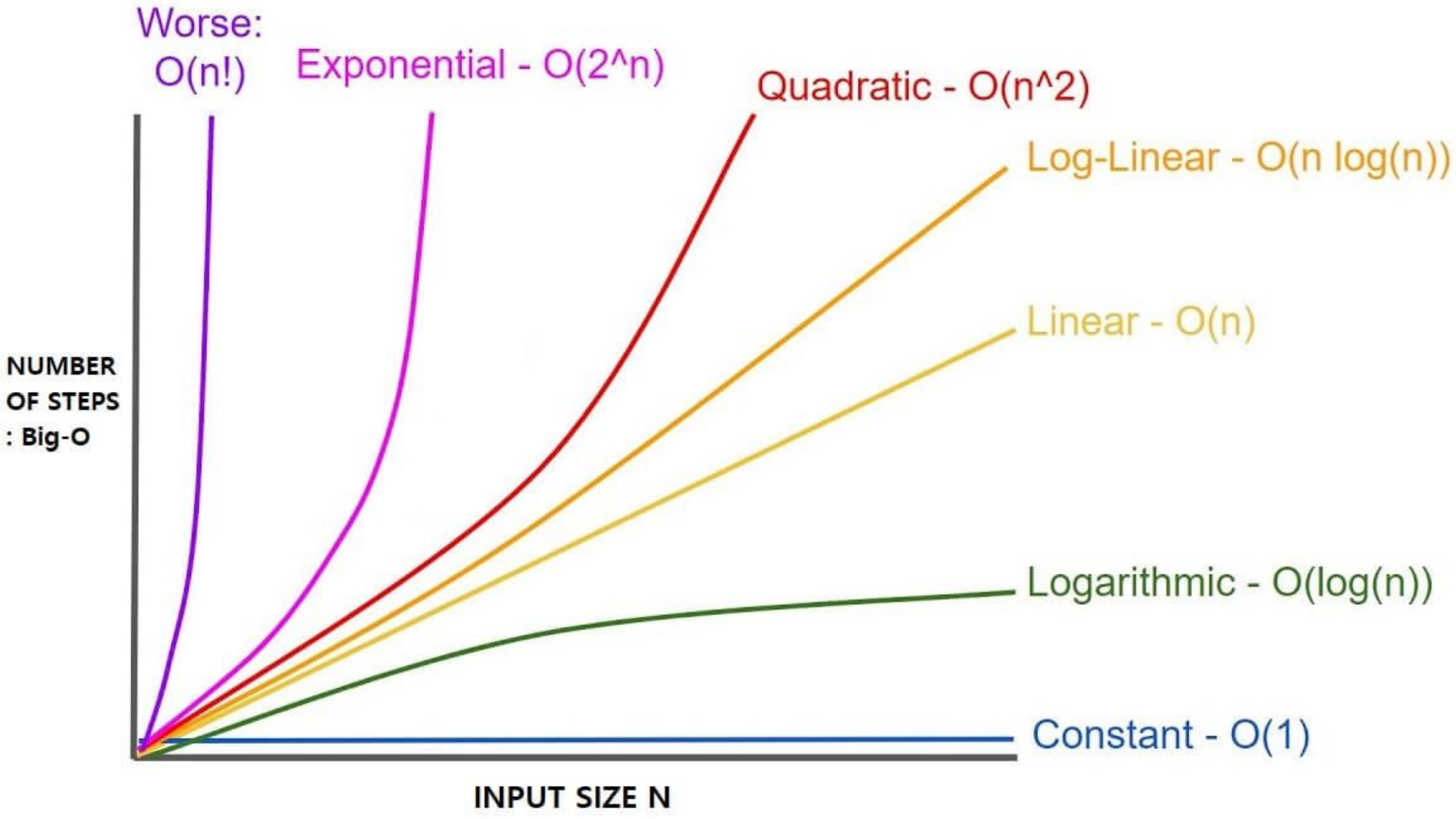
2

1

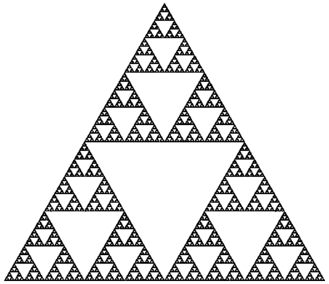
Running time estimation T(n)

$$T(n)=1+1+(n+1)+2n+3n+2+1$$
$$= 1+3n+2+3n+2+1$$
$$=6n+6 = O(n)$$

Algorithm growth rate



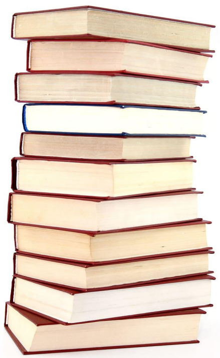
Recursion

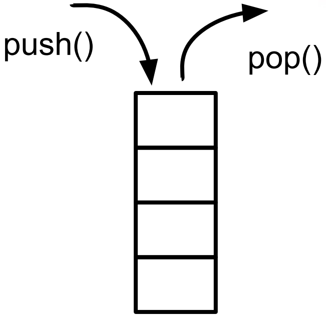


```
int factorial ( int n ) {
   if ( n == 0)   // base case
      return 1;
   else           // general/ recursive case
      return n * factorial ( n - 1 );
}
```

Stacks

- Stacks are follow a "Last-In-First-Out" (LIFO) model.

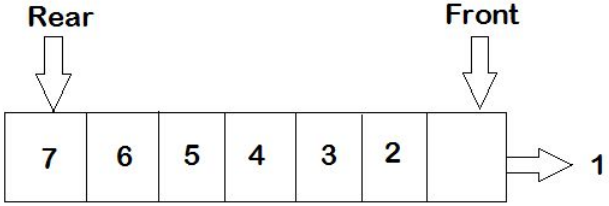




operation	Time Complexity
push	O(n)
pop	O(n)
top or peek	O(1)

Queues

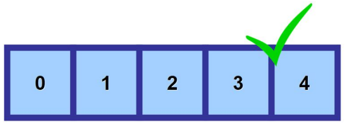
- Queues are follow a “First-In-First-Out” (FIFO) model.

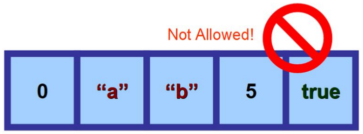


operation	Time Complexity
queue	O(n)
dequeue	O(n)
front or peek	O(1)

Arrays

- The most fundamental data structure.

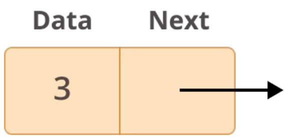


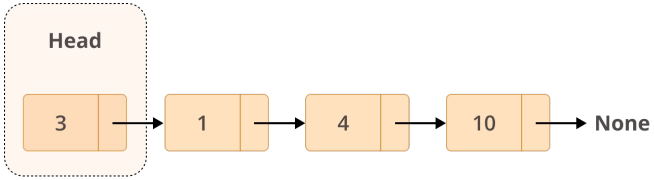


operation	Time complexity
lookup (get)	O(1)
append	O(1)
insert	O(n)
delete	O(n)
update	O(1)
Traverse	O(n)

Linked Lists

- Can be extended or reduced.



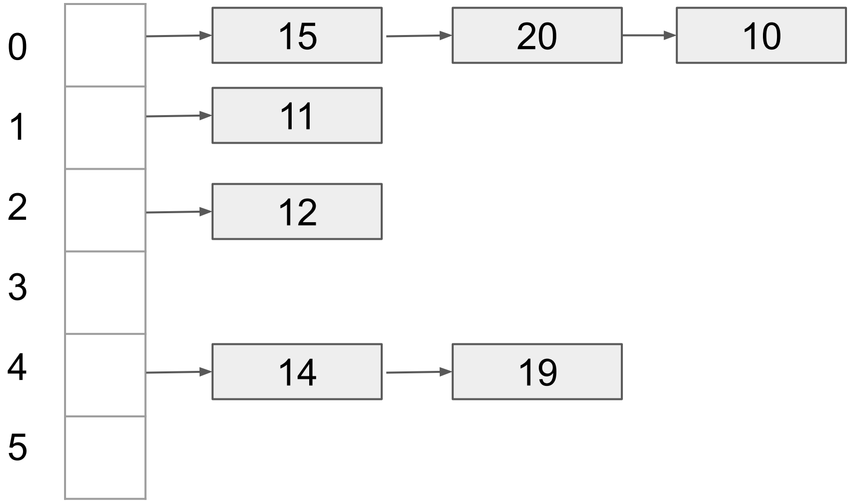


Operation	Time Complexity
lookup (get)	O(n)
add	O(n)
remove	O(n)
update	O(n)

Hashing

- Allow us very fast retrieval of data regardless the size of data and position of the targeted item.
- Is widely used in database indexing, cashing, compilers, error checking, password authentication, search engines, and more.

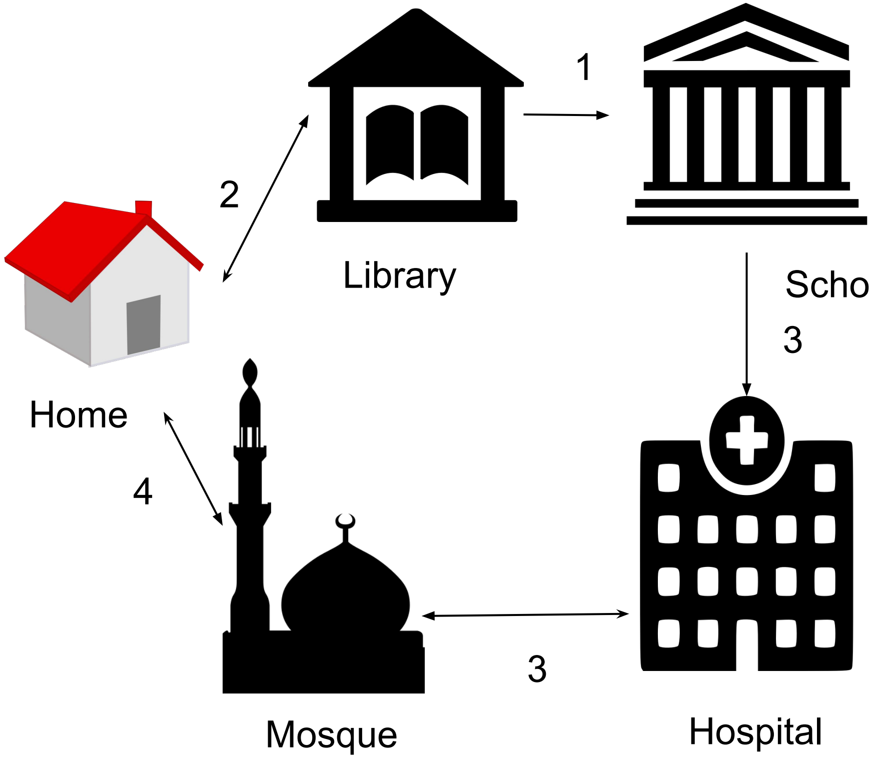
15	1					3	4			5



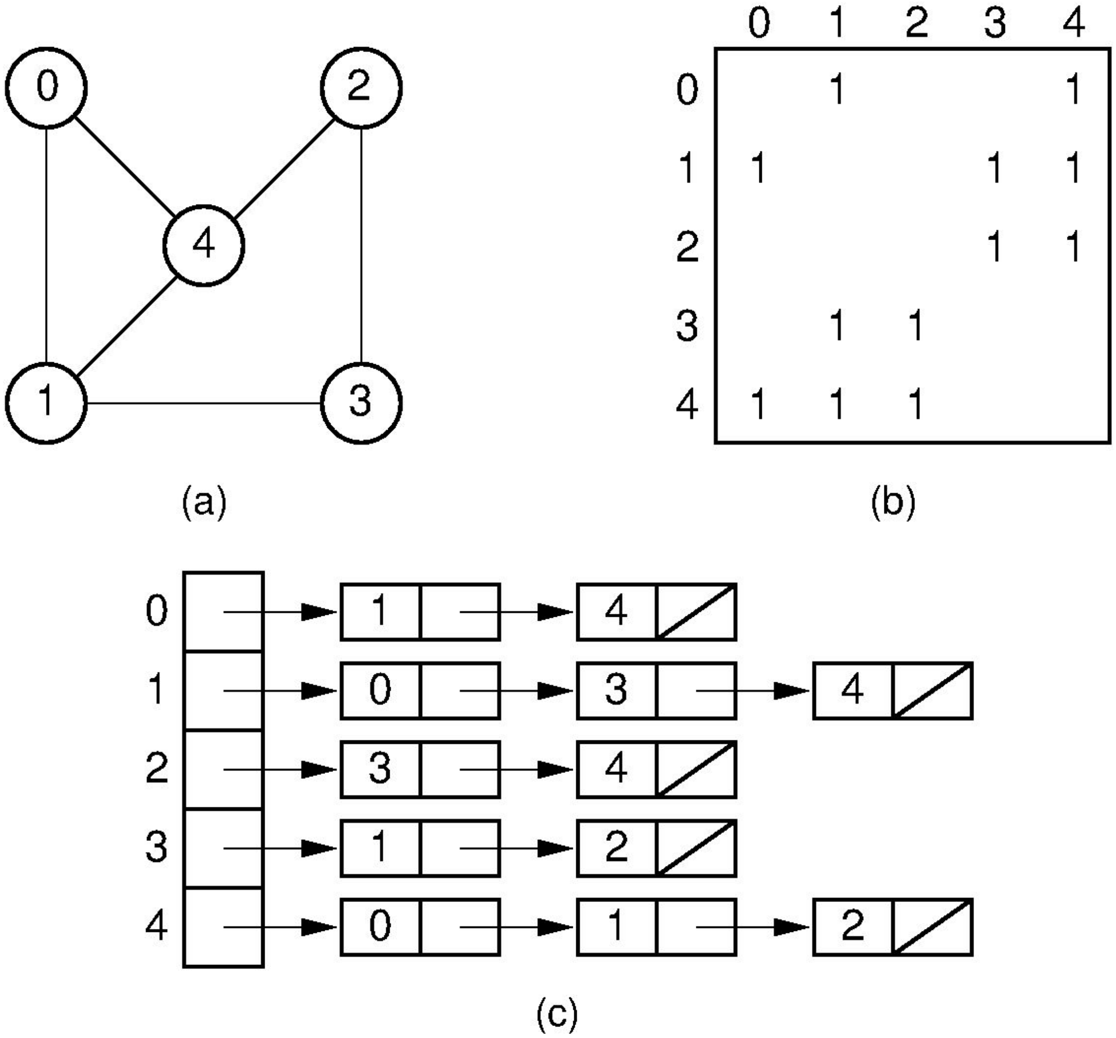
Graphs

- Can be extended or reduced.
- Non-linear data structure.

Find the shortest path from home to hospital?



Graph representation



(a)

(b)

	0	1	2	3	4
0		1			1
1	1			1	1
2				1	1
3		1	1		
4	1	1	1		

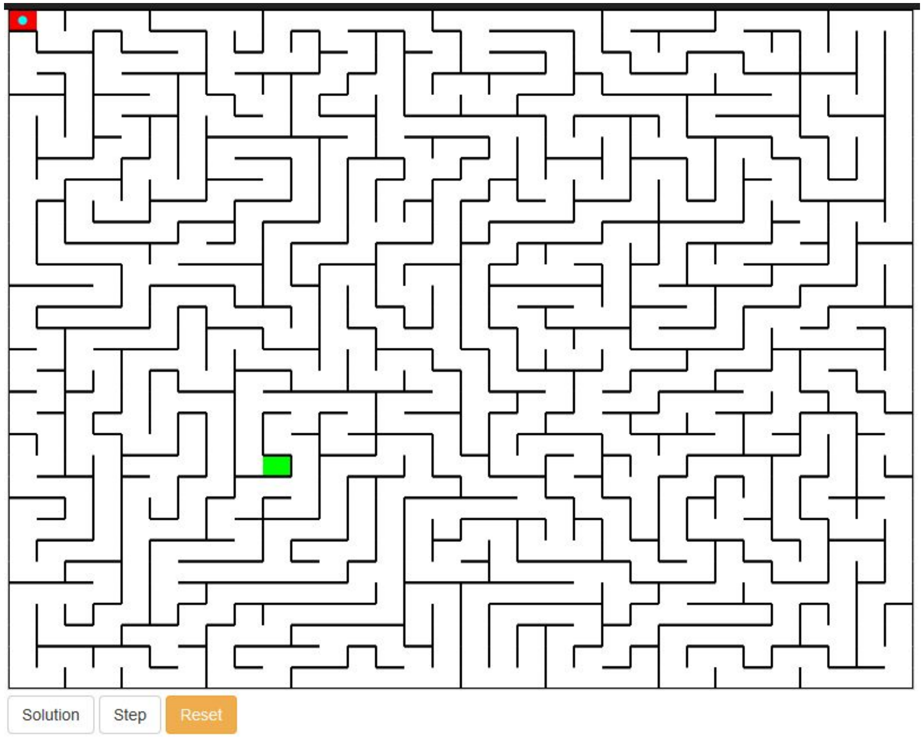
(c)

Common graph algorithms

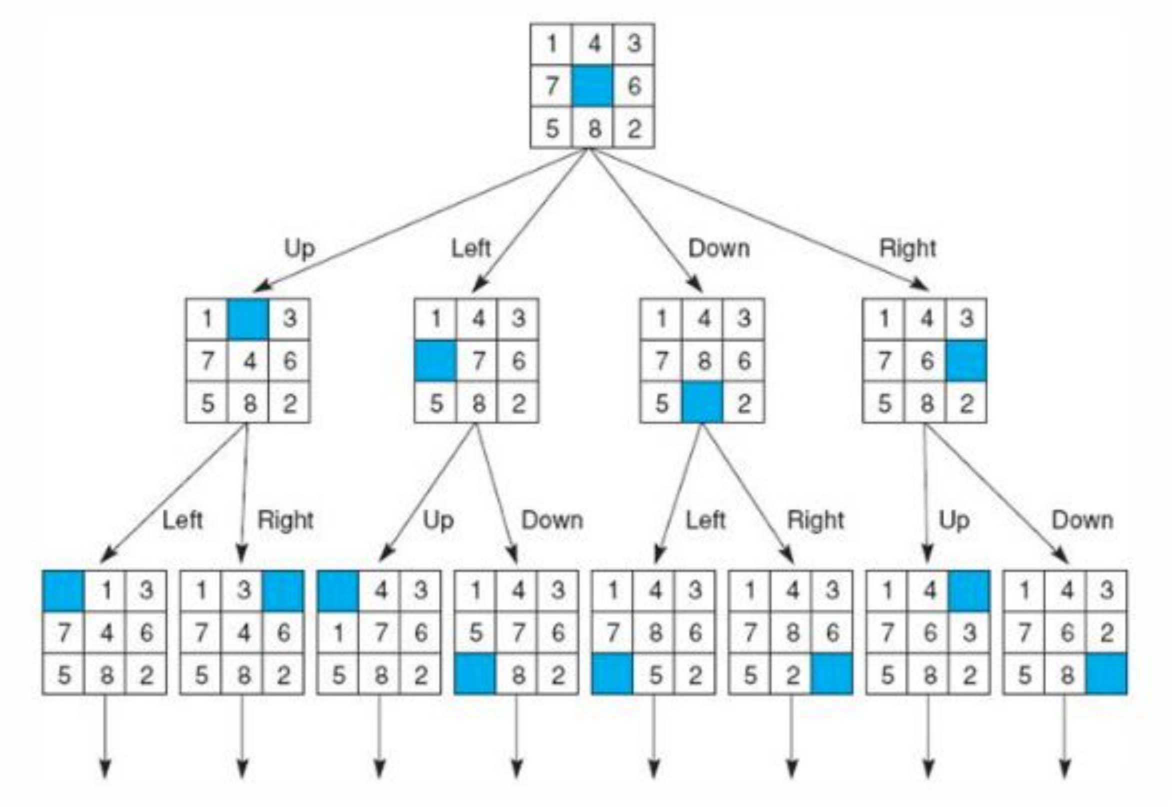
- Finding the shortest path between nodes.
    - Dijkstra’s algorithm
- Spanning Tree
    - Find the minimum path to connect all nodes in a graph.
        - Kruskal’s algorithm
        - Prim’s algorithm
- Traversing
    - Depth-First Search (DFS)
        - Pre-order
        - In-order
        - Post-order
    - Breadth-First Search (BFS)

Trees

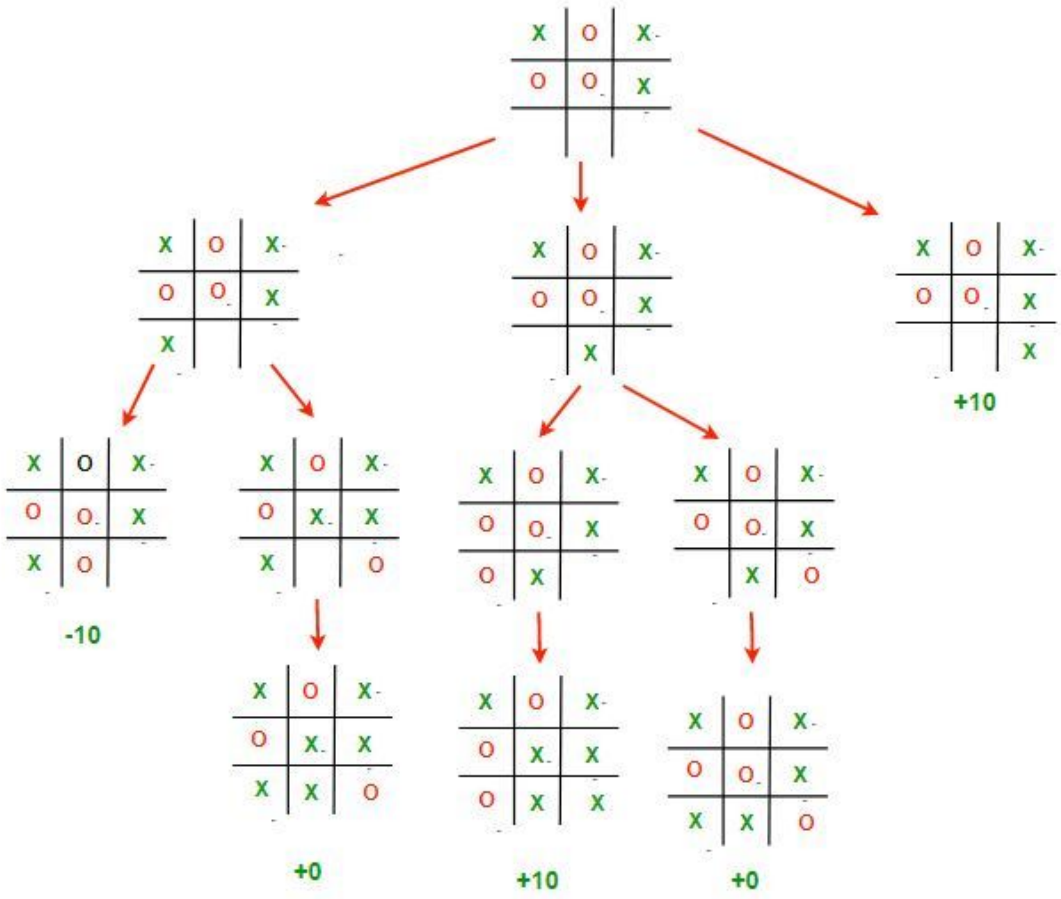
1	2	5
3	4	
6	7	8



8-puzzle

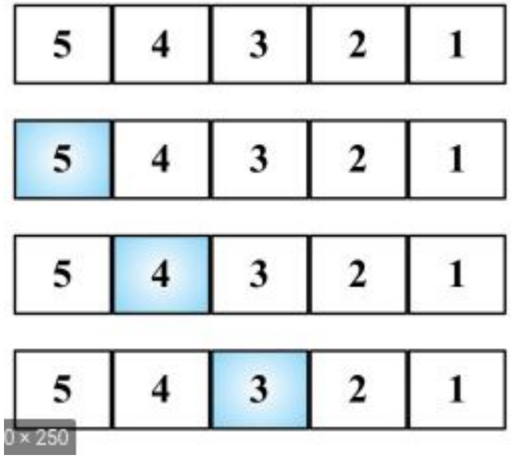


Tic toc toe

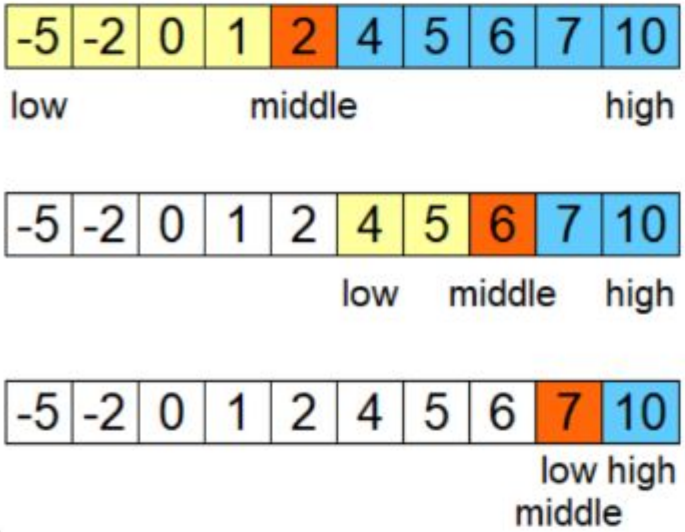

X O X
O O X
X O X
O O X
X
X O X
O O X
X
X O X
O O X
X
+10
X O X
O O X
X O
-10
X O X
O X X
X O
X O X
O O X
O X
X O X
O O X
X O
X O X
O X X
X X O
+0
X O X
O X X
O X X
+10
X O X
O O X
X X O
+0

Searching algorithms

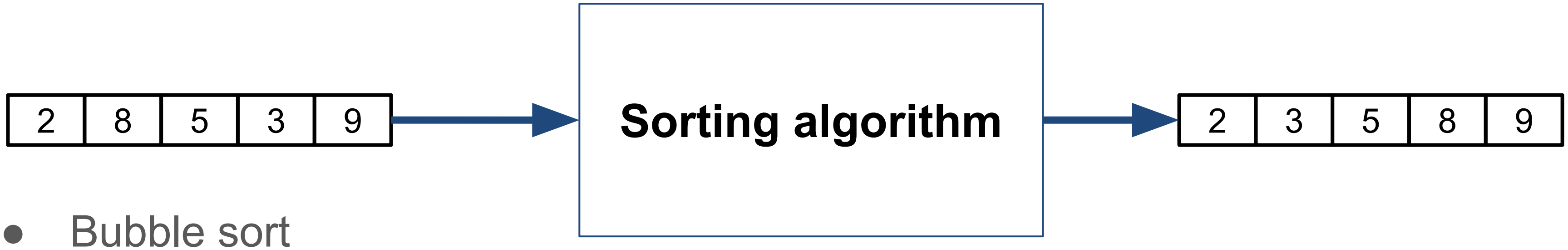
- Linear Search



- Binary Search



Sorting Algorithms



2	8	5	3	9

Sorting algorithm

2	3	5	8	9

- Bubble sort
- Quick sort
- Selection sort
- Insertion sort

Decision on choosing technique

- **Recursion** is used to solve problems that their definition include themselves.
- **Arrays** are homogeneous, fixed sized.
- **Hashing** is used for large amount of data.
- Stacks
- Queues
- Linked List
- Graph
- Tree
- Traversing
  - LInear Search
  - Binary Search
- Insertion sort
- Bubble Sort
-

References

1. Mark A. Weiss, Data Structures and Algorithm Analysis in Java, Addison Wesley, 3rd Edition, 2011.
2. Adam Drozdek, Data Structures and Algorithms in Java, 4th Edition, Cengage Learning, 2013.
3. Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein, “Introduction to algorithms”, 3rd Edition, MIT Press, 2009, ISBN 978-0-262-53305-8.
4. Ryuhei Uehara, “First Course in Algorithms Through Puzzles”, Springer, 2019, ISBN 978-981-13-3187-9.