Data Structures & Algorithms

What is data structures ? මොකක්ද data අරන් store කරලා තියන structure එක.
data handling
read /search

* A data structure is an arrangement of data in a computer's memory or even disk storage.

* It has a different way of storing and organizing data in a computer so that it can used efficiently.

* Efficient data structures are key to designing efficient algorithms.

Data structure ⟶ storage that is used to store and organize data.

Ex :- linear (arrays, stacks, queues)
non-linear (trees and graphs)

Interface - the set of operations that a data structure supports.

Linear Data Structure

* Linear data structures organize their data elements in a linear fashion, where data elements are attached one after the other.

Ex :-
* Array
* Linked List
* Stack
* Queue

Date

No

x Array

An array is a collection of data elements where each element could be identified using an index.

x Linked List

A linked list is a sequence of nodes, where each node is made up of data element and a reference to the next node in the sequence.

x Stack

A stack is actually a list where data elements can only be added or removed from the top of the list.

x Queue

A queue is also a list, where data elements can be added from one end of the list and removed from the other end of the list.

Non-Linear Data Structure

* Data elements are not organized in a sequential fashion.

Ex :- * multidimensional arrays
* trees and graphs.

The Need for Data Structures

Data structures
⟹ more efficient programs.

More powerful computers
⟹ more complex applications.

More complex applications demand more calculations.

Selecting a Data Structure

Analyze the problem to determine the resource constraints a solution must meet.

Determine the basic operations that must be supported. Quantify the resource constraints for each operation.

Select the data structure that best meets these requirements.

Data Types

1. **Built-in Data Type**
   integers, Boolean, Floating, String

2. **Derived Data Type**
   list, array, stack, queue

Basic operations ⟶ Traversing
Searching
Insertion
Deletion
Sorting
Merging

What is an algorithm?

Step by step procedure, which define a set of instructions to be executed.

Categories of algorithms

* Search, sort, insert, update, delete

How to write an algorithm

1) START
2) Declare three integers a, b & c
3) Defines values of a & b
4) Add value of a & b
5) Store output of step 4 to c
6) Print c
7) STOP

1. START ADD
2. get values of a&b
3. c ⟵ a+b
4. display c
5. STOP

What is an Array ?

Collection of similar data elements stored at contiguous memory locations. අඛණ්ඩ මතක

It is the simplest data structure where each data element can be accessed directly by only using index number.

An array is an object that is used to store a list of values.

* එක array එකකට එකම data types තියෙන්න ඕන.

Write an algorithm to find the largest among 3 numbers.

Step 01 - Identify 3 numbers

Step 02 - Compare 1st and 2nd number and find the largest.

Step 03 - Compare the found largest with the 3rd number.

Step 04 - Display the largest.

Memory

Variable — One data at a time.

Data structure — different types of data at a time.
different ways of storing data.

Linear
Arrays
Linked Lists
Stacks
Queues

Data elements are attached 1 after the other.
Easy to impliment, since the memory of the computer is also organized in linear fashion.

Non - Linear

Data elements are not organize in a sequential fashion.

Date No

Array

- Collection of data elements where each element could be identified using an index.

Selecting a data structure

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported.
3. Select the data structure that best meets these requirements.

Characteristics of data structures.

DATA Structure	Advantages	Disadvantages
Array	Quick inserts if index known (Fast access)	slow search slow deletes fixed size
Ordered Array	Fast search than unsorted array	slow inserts slow deletes fixed size
Stack	Last in, First-out access	slow access to other items
Queue	First-in, First-out access	slow access to other items
Linked List	Quick inserts, Quick delete	slow search

Arrays

Date

No

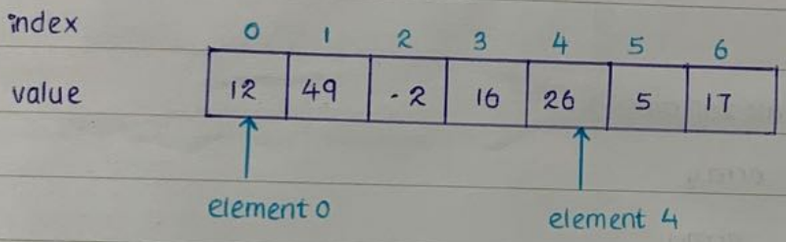
✓ Array is an object that used to store a list of values.

✓ It is made out of a contiguous block of memory that is divided into a number of cells.

✓ Each cell will hold a value, and all those values are of the same type.

* Arrays are a collection of data elements where each element could be identified by an index.

* Assuming the below array's name is ANC:



Elements - data stored in particular cell,

```
ANC [2] = -2
ANC [0] = 12
```

Array length,

```
ANC.Length = 7
```

Array size / length is determined when it created, It cannot be modified.

Array length එක බලන්න, elements ගාන not index ගාන.

```
System.out.println Carr length);
```

```
import java.util.Scanner;
public class ArrayImp {
  public static void main (String [] args){
   int [] arr = new int [5];
   int  size = 5;
```

← scanner class එකට obj හදනවා.

```
Scanner UserIn = new scanner (System.in);

for (int j=0 ; j<size ; j++ ) { //data 5ක් ගන්න කියා
  arr [j] = UserIn.nextInt ();   loop දානවා
}
```

user input ගන්නවා

0, 1, 2, 3, 4

```
for (int i=0 ; i<size ; i++ ) {
  SOP (arr [i]);
}
```

input → 1, 3, 4, 5

output → 1, 2, 3, 4, 5

```
}
```

1. Array declaration & initialization
2. Traverse through an array
3. Print elements in an array
4. Insert data into an array.
5. Delete data into an array.
6. Update data in an array.

Using your prefered programming average value of array elements.

1. Initialize a variable "sum" to 0.
2. Get no of elements using 'array length'
3. Use for loop to iterate through the array.
4. sum += [i]
5. Calculate average = sum / array.Length
6. Return the average

Date No

s whether an array is correct

ng to a value of the array length)

d Exception" is thrown.

ANC [6] ✓

ANC [1.5] ×

ANC [j] ×

```
public class ArrayImp {
  public static void main (String [] args ){

    char [] arr = {'A','B','C'};
                   0   1   2
    for (int i =0; i<array.length; i++){
                   array size එකට වඩා ඉන් අඩුවෙන්
      SOP (array [i]);
                   ↑print i
    }
  }
}
```

output → A, B, C

Intial

Defini

Initial

Take

Take

Date No

✓ When a program is running, it checks whether an array is correct. This is called Bounds checking.

✓ If the array is illegal (consoling to a value of the array length) an expection "Array Index Out Of Bound Exception" is thrown.

ex:- ANC [-1] ✗
ANC [0] ✓
ANC [4] ✓
ANC [7] ✗
ANC [6] ✓
ANC [1.5] ✗
ANC [j] ✗

Array operations

1. Array declaration & initialization
2. Traverse through an array
3. Print elements in an array
4. Insert data into an array.
5. Delete data into an array.
6. Update data in an array.

```
public class ArrayImp {
  public static void main (String [] args ){

    char [] arr = {'A','B','C'};
                   0   1   2
    for (int i =0; i<array.length; i++){
                   array size එකට වඩා 1ක් අඩුවෙන්
      sop (arrn [i] );
    }
  }
```

Using your prefered programming language, write a program to calculate average value of array elements.

1. Initialize a variable "sum" to 0.
2. Get no of elements using "array length"
3. Use for loop to iterate through the array.
4. Sum += [i]
5. Calculate average = sum / array.Length
6. Return the average

```
public class Average Cal {
    public static void main (String [] args ) {

        double int [] array = { 10, 25, 24, 15, 17, 35, 86 }
            int sum = 0, avg = 0 ;

        System.out.println (" The average value of the array is : " + average );
    }
```

Intializing an Array — `int Array [] = { 1,2,3,4 } ;`

Defining an Array — `int Array [5];`

Initializing of array elements

```
Array [0] = 1 ;
Array [1] = 2 ;
   ⋮
Array [5] = 16 ;
```

Take out data from an Array — `Array [1];`

Take all the data out from an Array

```
for (int a=0 ; a<= array.Length - 1 ; a++) {
    Console.WriteLine (array [a]);
}
```

Write a program to initialize an array with 10 character elements (A.... J) & display characters in the reverse order.

1. Initialize a character array with elements 'A' to 'J'.
2. Use for loop to get elements from 9 to 0.
3. Display the reverse array.

```
class Program {
    static void Main (String [] args ) {
        char Array [] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J' }
        for (int i = Array.Length - 1 ; i >= 0 ; i-- ) {
            Console.WriteLine (Array [i] + " " );
    }
  }
}
```

Array declaration → Char vowels [5];

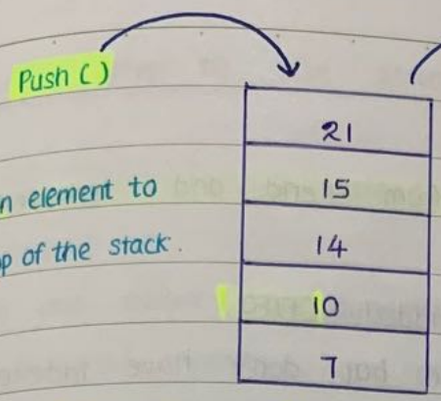
input data into elements → vowels [0] = 'a';

initializing an array → char vowels [5] = {'a', 'e', 'i', 'o', 'u'}

Stacks and Queues

Stack

- ✓ A list where data elements can only be added and removed from the top of the list.
- ✓ The data is accessed using LIFO method.
- ✓ Elements are stored in order of insertion, top to bottom.
- ✓ Stacks do not have indexes.
- ✓ Clients can only add / remove / examine the Last (top) element.



Add an element to the top of the stack.

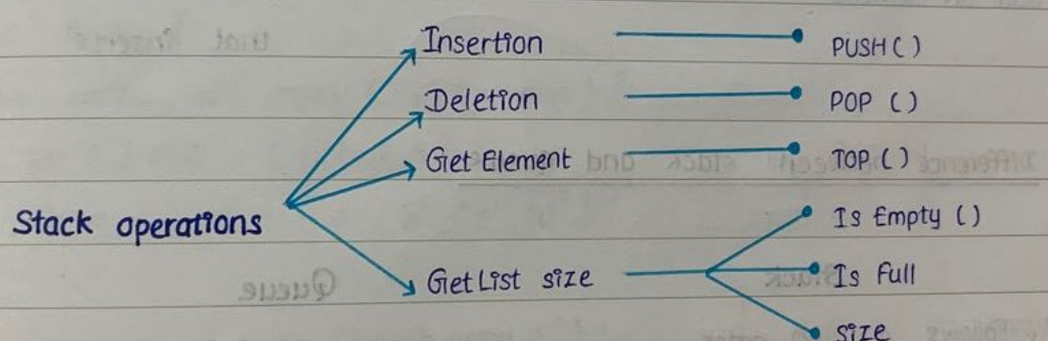
Remove the top element & returns it.

Data → 7, 10, 14, 15, 21

Peek () : Examine the top element & without removing it.

size () : How many items are in stack?

is Empty () : False if there are 1 or more items in stack, true otherwise

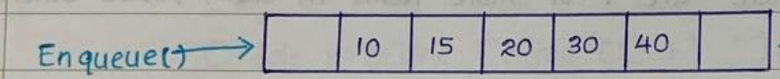


* In stack only the top value can be accessed, other middle values or the last value cannot be accessed.

Date No

Queue

- A list where data elements be added from 1 end and removed from another.
- Retrives elements in the order they were added. (FIFO)
- Elements are stored in order of insertion but don't have indexes.
- Clients can only add to the end of the que & remove from the front of que.
- Commonly used in computers for scheduling resources such as printers, disk drivers etc.



add an element

remove the 1st element that inserts

Difference between stack and queue

Stack	Queue
✓ Follows a LIFO order	✓ Follows a FIFO order
✓ Both insertion & delete takes place in a 1 end (top)	✓ Has 2 ends to insertion & deletion. (Last end → insert, front end → delete)
✓ Insertion - PUSH () Deletion - POP ()	✓ Insertion - enqueue () Deletion - dequeue ()
ex:- undo option spelling a word	ex:- ATM que OS process scheduling queue. Printing documents.

Why & when to use stack and queues.

- We use stack / queue instead of arrays when sequential access is required.
- Stack and queue are dynamic while arrays are static.
- To efficiently remove any data from the start or end of data structure.
- When you want to get items out in the same order that you put them in (FIFO)
- When you need to bring things out in the opposite order that they were put in (LIFO)

```
import java.util.Scanner;
Public class Array {
    public static void main (String args) {
        int [] ANC = {1,2,3,4};          // Array initialization
        string [] ABC = {"a","b","c","d"};

        System.out.print ("ANC array:");
        for (int i=0; i<ANC Length; i++) {
            SOP (ANC [i] + " ");
        }                                  // Print the int Array
                                           // ANC array: 1 2 3 4

        System.out.print ("ABC array:");
        for (int i = ABC Length; i>=0; i--) {
            SOP (ABC [i] + " ");
        }                                  // Print the string array
                                           // ABC array: a b c
                                           // (in reverse form)
}
```

```
int [] BCD = new int [4];
BCD [0] = 11;          # declaring an int array
BCD [1] = 12;          # assign values for each index of the
BCD [2] = 13;            array
for (int i=0; i< BCD.Length ; i++ ) {
    SOP (BCD [i] + " ");      # 11 12 13
}
```

```
Scanner scan = new Scanner (System.in); // Get input from
String [] colors = new String [3];          keyboard
colors [0] = scan.next ();
colors [1] = scan.next ();
colors [2] = scan.next ();
```

```
SOP (colors [0]);
SOP (colors [1]);
SOP (colors [2]);
```

```
for (int i=0; i < colours.length ; i++){
    SOP (colors [i] + " ");
}
```

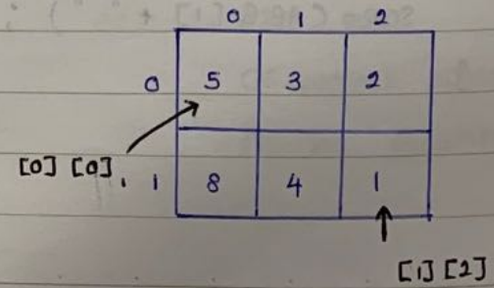
2D Arrays (Multi Dimensional Arrays)

- ✓ Can describe as "arrays of arrays"
- ✓ Made up of same data type.

```
int twoDMatrix [2] [3];
int twoDMatrix [2] [3] = { {5,3,2}, {8,4,1} };
```

Each element can be changed as;

```
twoDMatrix [0] [0] = 5;
twoDMatrix [1] [2] = 1;
```



① Write a program to calculate the average value of array elements.

```
public class AverageArray {
    public static void main (String []args ) {
        int [] array = { 1,2,3,4,5} ;
        int sum = 0 ;

        for (int i=0 ; i<array.length ; i++ ) {
            sop (array [i] +" " ); // 1
                                   // 2
            sum += array [i] ;     // 3
                                   // 4
        }                          // 5
        double avg = sum /array.Length ;
        sop (avg);                 // 3.0
```

array = {1,2,3,4,5}

sum	i	i<a.L	avg	out
0	0	✓		1
0+1 = 1	1	✓		2
1+2 = 3	2	✓		3
3+3 = 6	3	✓		4
6+4 = 10	4	✓		5
10+5 = 15	5	X	15/3	3.0

② Write an array to initialize 10 characters (a-j) and display the characters in reversed order.

```
public class arrayReverse {
    public static void main (String []args ) {
        char [] array = {'a' ,'b','c','d','e','f','g','h','i','j'} ;
        for (int i=0 ; i< array.length ; i++ ) {
            sop (array [i] +" " ); // a b cd e f gh i j
        }
        for (int a = array.length - 1 ; a >= 0 ; a-- ){
            sop (array [a] +" " ); // j i h g f e d c b a
        }
    }
}
```

Create stacks in Java

```
import java.util.Stack;  // package

Stack <Type> stacks = new stack <> (); // initialize a stack
```

stack name

stack type

(Integer, String)

```
stack.push ("value");   // Add elements to stack [Push]
```

stack name

```
String a = stacks.pop();   // remove elements from stack [POP]

String b = stacks.peek();   // Access the top of the stack [Peek]
```

variable

```
int position = stacks.search("element"); // search the element (search)
boolean result = Stacks.empty(); // check the stack empty
```

```
import java.util Stack;
public class stack {
    public static void main (String [] args) {
    stack <string> animals = new stack <> ();
    // Add elements to stack
    animal.push ("Dog");
    animal.push ("Cat");
    animal.push ("Owl");

    SOP ("stack :" + animal);   // stack : [Dog, Cat, Owl]
    // Remove & return elements from stack
    string removed = animal.pop();

    SOP ("removed element :" + removed);   // removed element : owl
    // Access the top of the stack   [Dog, Cat]
```

2 1

```
        String Top = animal.peek();
        SOP("Top of the stack : " + Top);  // Top of the stack : Cat
        // Search an element in the stack
        int position = animal.search("Dog");
        SOP("Position of Dog is : " + position); // Position of Dog is : 2
        // Check whether the stack is empty
        boolean result = animal.empty();
        SOP("Is the stack empty ? " + result); // Is the stack empty ? No
        // Get stack size
        int value = animal.size();
        SOP("The stack size is : " + value); // The stack size is : 2
    }
}
```

'Empty stack Exception' ⟶ thrown if we call pool() when the invoking stack is empty.

```
import java.util.stack;
public class stack {
    // pushing elements on the top of the stack
    public static void stack.push(stack<Integer> stack) {
        for(int i=0; i<5; i++){
            stack.push(i);
        }
    }
    // poping element from the top of the stack
```

```
public void stack_POP (stack<Integer> stack) {
    for (int i=0; i<5; i++) {
        Integer y = (Integer) stack.pop();
        sop (y);
    }
}
```

Cast operation

variable that assign the cast

```
// Display element at top of stack
static void stack_peek (stack<Integer> stack) {
    Integer element = (Integer) stack.peek();
    sop ("Element at the top of the stack : " + element);
}
// search element in stack
static void stack_search (stack<Integer> stack, int element) {
    int pos = (Integer) stack.search (element);
    if (pos == -1) {
        sop ("Element not found");
    } else {
        sop ("Element is in : " + pos);
    }
}
```

must

int

```
public static void main (String [] args) {
    stack<integer> stack = new stack<Integer> ();
    stack_push (stack);
    stack_pop (stack);
    stack_peek (stack);
    stack_search (stack, element : 2);
    stack_search (stack, element : 6);
}
}
```

no need

✓ must

Input =
4
3
2
1
0

Element at top of stack : 4

Element is in : 3

Element not found.

Create Queue in Java

Date

No

✓ since the queue is an interface, we cannot provide the direct implimentation of it.

✓ In order to use the functionalities, we need to uses classes that impliment it,

- Array Deque
- Link list
- Priority Queue.

```
import java.util.Queue ;        // package

Queue <String> animal1 = new LinkedList<>();        // implimentation of queue
Queue <string> animal2 = new ArrayDeque<>();
Queue <string> animal3 = new PriorityQueue<>();
```

Methods of Queue

✓ add() } - Insert the specified elements into the queue.
offer() }

✓ element() } - Return the head of the queue.
peek() }

✓ remove() } - Return & remove the head of the queue.
poll() }

```
import java.util.Queue;
public class queue {
    public static void main (String [] args ) {
        Queue <Integer> numbers = new LinkedList <> ();
        // insert elements to the Linked list queue
        numbers.offer (1);
        numbers.offer (2);
        numbers.offer (3);

        SOP ("Queue : " +numbers );    // Queue : [1,2,3]
        // return head of the queue
        int accessNum = numbers.peek ();
        SOP ("Accessed Number : " + accessNum ); // Accessed Number : 1
        // Reverse the head of the queue
        int removed = numbers.poll ();
        SOP ("Removed Number : " + removed); // Removed Number : 1

        SOP ("Updated queue : " + numbers ); // Updated Queue : [2,3]
    }
}
```

Problem set 01 - Arrays, Stacks & Queues

① Write an algorithm to calculate the average value of array elements.

1. Initialize an Integer type array.
2. Initialize variable sum = 0.
3. Get no. of elements using arrayname.Length
4. Use for loop to iterate through the array.
5. Make sum += [i]
6. Calculate avg = sum /array.Length
7. Return the avg.

② Find the difference in the operation of reversing a string of words using (a) stack
(b) queue

(a) stack

① Get a string variable (sentence) & assign a value to it
② Get an empty stack
③ Split the sentence into words.
sentence.split(" ");
④ Push the words into stack
⑤ Repeat it until all the words in the sentence is pushed
⑥ Then pop the words from stack in reverse order.

b) Queue

① Get a string variable (sentence) & assign a value to it.
② Initialize an empty queue.
③ Split the sentence into words. using a space as the delimiter.
④ Enqueue each word into the queue in the order they are encounted.
⑤ Dequeue words from the queue are by one and concatenate them to form the reversed string.
⑥ Return the reversed string.

```
import java.util.Stack;
public class stack {
p.s.v.m (string [Jargs ) {
 string sent = "Programming expert";
SOP (sent);
 reversedsentence (sent);
}
 s v reversedsentence (string sent){
stack<string> stack = new stack<>();
String [Jword = sentence.split(" ");
for (int i=0; i< word.length; i++){
   stack.push (word [i]);
}
```

```
        while (!stack.isEmpty()) {
            sop (stack.peek()+" ");
            stack.pop();
        }
    }
}
```

output → Programming expert
expert Programming

③ Write an algorithm that checks for duplicates in a date column of a spread sheet.

1. Initialize an empty set to store encountered values.
2. Iterate through each value in the data column.
    a. If the item already in the set, return true. (duplicate found)
    b. Otherwise add the value to set.
3. While the loop is completes without finding any duplicates, return ~~true~~ false. (no duplicates)

Problem Set 02 - Array Stack & Queue

① Write an algorithm to check if a word is a Palindrome or not using:

a) Array

b) Stack

c) Queue

According to your opinion what is the most efficient data structure. Justify your answer. Impliment it with java.

using Array

```
public class Array {
    psvm (String [] args) {
     String word = "madam";
     if (Palindrome (word)) {
        sop (word + " is a palindrome");
     } else {
        sop (word + " is not a palindrome");
     }
    }

    Public static boolean palindrome (string word) {
      char [] a = word.to char Array ();
      int length = a.length ();

for (int i=0; i < length /2; i++) {
     if (a [i] != a [length -i -1]) {
       return false;
     }
    }
      return true;
    }
}
```

madam

a = ['m', 'a', 'd', 'a', 'm']

length = 5

i	i < length/2	a[i]
0	0 < 5/2 ✓	am
1	1 < 5/2 ✓	la
2	2 < 5/2 ✓	ad
3	3 < 5/2 ✗	

a[i] != a[length - i - 1]

0 = 4 (5-0-1)

m = m ✓

1 = 3 (5-1-1)

a = a ✓

output ⟶ madam is a palindrome

using stack

```
import java.util.Stack;
public class stack {
    public static void main (String [] args) {
        String word = "Palindrome";
        if (Palindrome (word)) {
            SOP (word + " is a palindrome");
        } else {
            sop (word + " is not a palindrome");
        }
    }
    public static boolean Palindrome (String word) {
        char [] a = word.toCharArray ();
        Stack <character> stack = new stack <> ();
        for (char c:a) {
            stack.push (c);
        }
        for (char c: a) {
            if (stack.pop () != c) {
                return false;
            }
        }
    }
    return true
```

queue queue linked list

For palindrome checking, using an array is most efficient - It has the simplest implementation & required the last amount of memory & operations.

② Write an algorithm to find the minimum & maximum element of an array.

1. Initialize an array
2. Sort the array in ascending order.
3. 1st element will be the minimum & the last element will be the maximum.
4. Print the min & max element.

```
public class array {
    psvm (String [] args ){
    int [] array = {4,2,5,4,3,8,9};
     int max = array [0];
     int min = array [0];

    for (int i=0 ; i< array.Length ; i++ ) {
        if (array [i] > max ){
            max = array [i];
        }
        if (array [i] < min ) {
            min = array [i];
    }
  }
    SOP ("min value : " + min + " Max value : " + max );
  }
}
```

i	i<7	array [i] > max	array [i] < min	min	max
0	✓	4>4	4<4	4	4
1	✓	2>4	2<4	2	4
2	✓	5>4	5<2	2	5
3	✓	4>5	4<2	2	5
4	✓	3>5	3<2	2	5
5	✓	8>5	8<2	2	8
6	✓	9>5	9<2	2	9

Date No

③ Write pros and cons when implementing stack using ques & array.

- **Stack using queue**

Pros	Cons
✓ Simple	✓ slower performance
✓ Dynamic size	✓ Consumes more memory
✓ No fixed capacity	✓ complexity of operation.

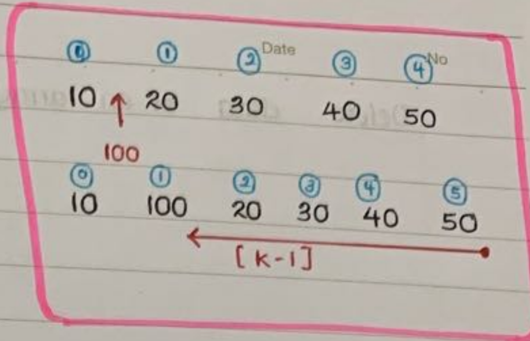
- **Stack using array**

Pros	Cons
✓ Efficiency	✓ Fixed size
✓ Predictable space usage	✓ Memory management
✓ Simple	✓ Potential overflow or underflow

④ Java Virtual Machine (JVM) follows a stack-based approach. Why did the developers go for a stack-based architecture.

Because it simplified memory management and execution flow control, making it easier to impliment features like probability, security & optimization across different platforms.

Insert data to an array

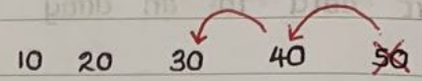



```
import java.util.Scanner;
public class ArrayImp {

  public static void main (String []args) {

    int [] arr = new int [10]; // array එක 10 වෙනකල් වැඩි කරන්න පුළුවන්.
    int size = 5;

    Scanner UserIn = new Scanner (System.in); → sop ("Enter Data :");
    for (int j=0 ; j< size ; j++) {
        arr [j] = UserIn.nextInt();
  }
    sop ("Output of Array ");
     for (int i=0 ; i<size , i++) {
         sop (arr [i]);
     }

      sop ("Enter the Index :"); // Asking for index
      int ind = UserIn.nextInt();

      sop ("Enter the Element :"); // Asking for element value
      int ele = UserIn.nextInt();

      size ++ ;   // array එකේ size වැඩි කරනවා (6)
     for (int k = size ; k >= ind ; k-- ) {   ← k අඩු කරගෙන යනවා මොකද 6වන index එකේ ඉඳලා ඉස්සරහට එන්නේ
               ↑ අලුතින් insert කරන index
          k=6
     arr [k] = arr [k-1];
              ↑ kට assign කරනවා කලින් කොටුවේ value එක

     arr[ind] = ele;   data දාගන්නවා
```

ProMate

Date No

Delete data of an array.

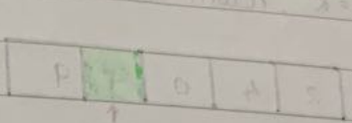


Previous size = 5

new size = 4

Algorithms & Algorithmic Comparison

Searching - Locating information

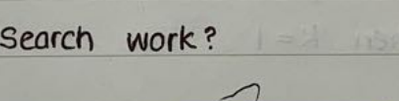


store in arrays and linked lists.

Types of searching algorithms.

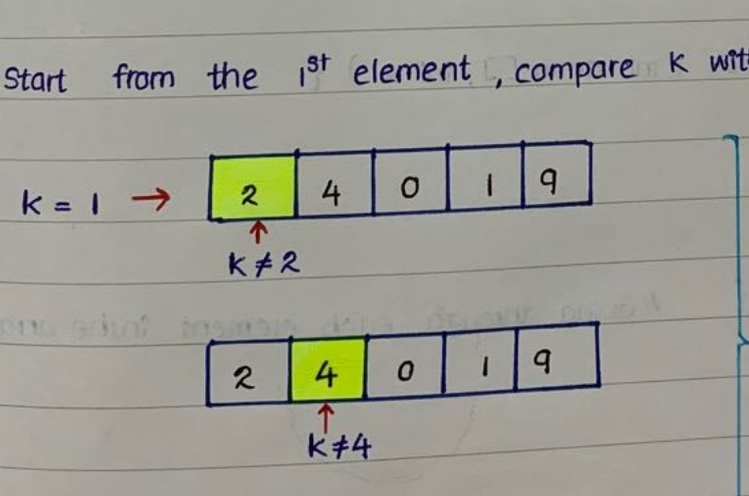
1. Linear
2. Binary
3. Bredth First
4. Greedy

How Linear Search work?



Search for element K = 1

1. Start from the 1st element, compare K with each element.



2. If $x == k$, return the index

Element found

3. Else, return not found.

Linear search Algorithm

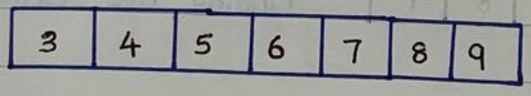
```
Linear Search (array, key)
    for each item in the array
        if item == value        // when k=1
        return its index
```

Linear search in Java

```
class Linear Search {
    public static int linear Search (int array [], int x) {
    int a = array.length ;

    for (int i=0; i<n; i++) {    // Going through each element in the array.
      if (array [i] == x)
      return i ;
    }
    return -1
}
```

```
psvm (String args[]) {
 int array[] = {2,4,0,1,9};
 int x=1;

 int result = linearSearch(array, x);

 if (result == -1)
    SOP("Element not found");
 else
    SOP("Element found at index: "+result);
 }
}
```

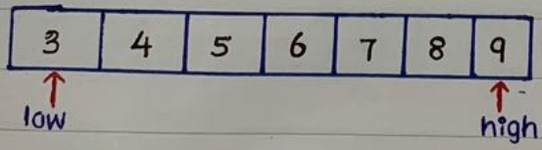
How Binary search work?



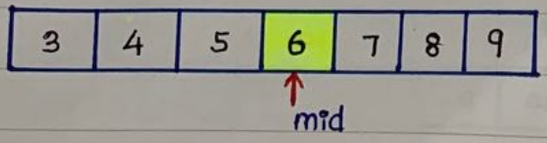
arrange elements in ascending order

Searching for element x = 4

1. Set 2 pointers low and high at the lowest and highest positions respectively.



2. Find the mid element (mid) of the array arr [(low+high)/2] = 6

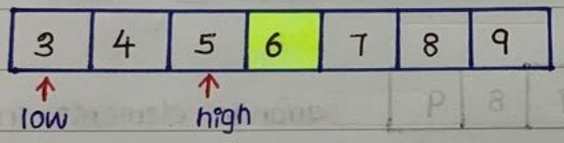


3. If $x == mid$, then return mid. Else, compare the element to searched with m.

4. If $x > mid$ compare x with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.

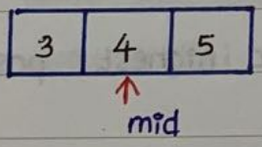
5. Else, compare x with the middle element of the elements of the elements on the right side of mid. This is done by setting low to low = mid + 1.

6. Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high high = mid - 1



Finding mid element

7. Repeat step 4 to until low meets high.



8. $x = 4$ is found.

Bubble Sort

Date No

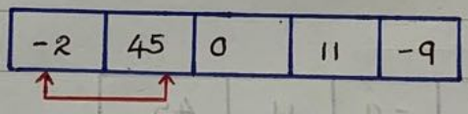
* Not efficient.
* It is a sorting algorithm that compares 2 adjacent elements and swaps them until are in the intended order.

1. First Iteration (Compare and swap)

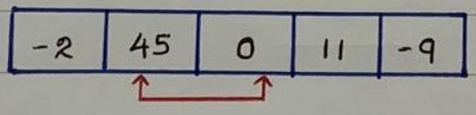
1. Starting from the first index, compare the first and the second element.
2. If 1st element > 2nd element, they are swapped.
3. Now compare 2nd and 3rd element. Swap them if they are not in order.
4. Above process goes until the last ~~process~~ element.

step = 0

i = 0

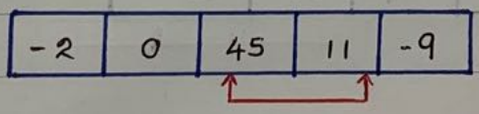


i = 1

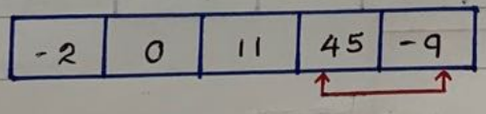


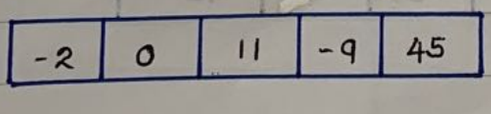
Compare the Adjacent Elements

i = 2



i = 3





Date

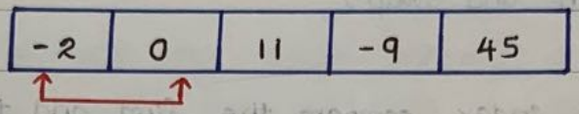
No

2. Remaining Iteration

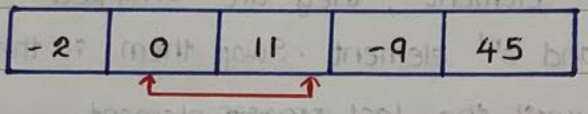
* The same process goes on for the remaining iterations.
* After each iteration, the largest element among the unsorted elements is placed at the end.

step = 1

i=0

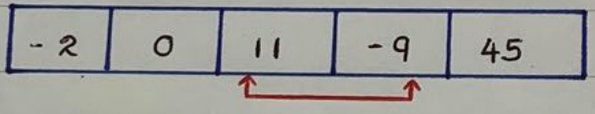


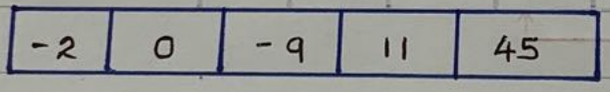
i=1



Put the largest element at the end.

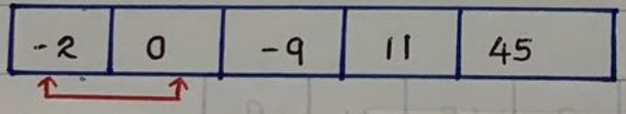
i=2





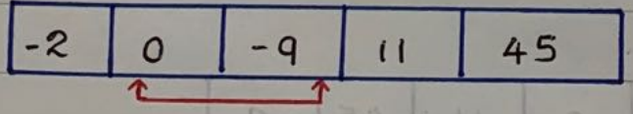
step = 2

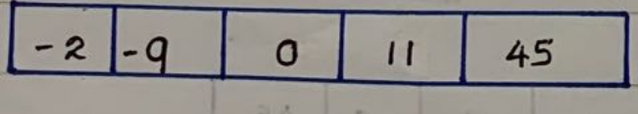
i=0



Compare adjacent elements

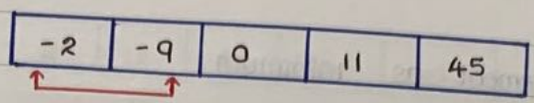
i=1

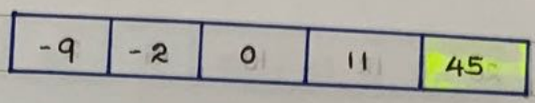




step = 3

i=0





Bubble Sort code in Java

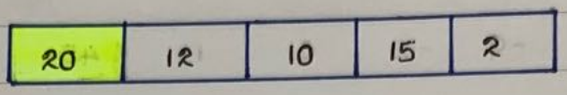
```
class Main {
// perform bubble sort

static void bubbleSort (int array []) {
    in size = array.length;

// loop to access each array element
  for (int i=0 ; i< size-1 ; i++ )

// loop to compare array elements:
 for (int j=0 ; j< size-i-1 ; i++)

// compare two adjacent elements
 if (array [j] > array [j+1]) {

int temp = array [j];
array [j] = array [j+1];
array [j+1] = temp ;
}
}

psvm (String args []) {

int [] data = {-2,45,0,11,-9};
// call method using class name
Main.BubbleSort (data);

SOP ("Sorted Array in Ascending order:");
SOP (Arrays.toString (data));
```

Date

No

Selection Sort

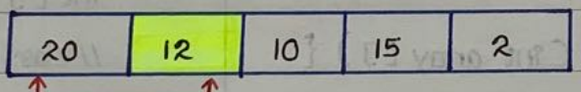
1. Set the first element as minimum.

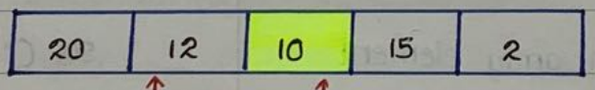


Select 1st element as minimum

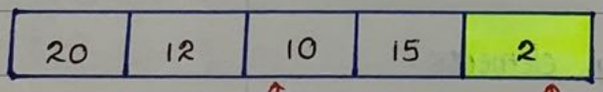
2. Compare minimum with the 2nd element. If the 2nd element is smaller than minimum, assign the element as minimum

Compare minimum with 3rd element. Again if the 3rd element is smaller, then assign minimum to the 3rd element otherwise do nothing.
The process goes until the last element.

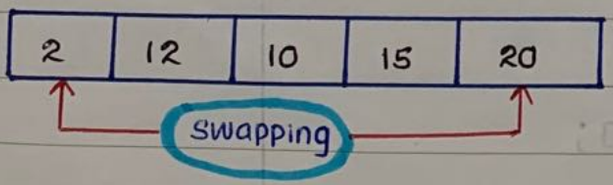




Compare minimum with remaining element.



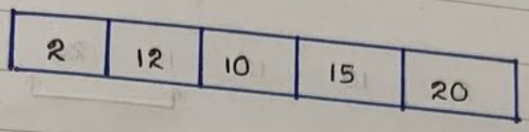
3. After each iteration minimum is placed in the front of the unsorted list.



Swap 1st with minimum

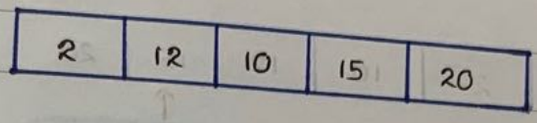
step = 1

i = 0



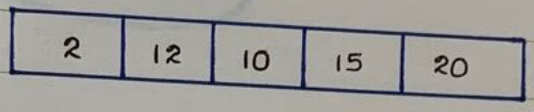
min value at index 2

i = 1

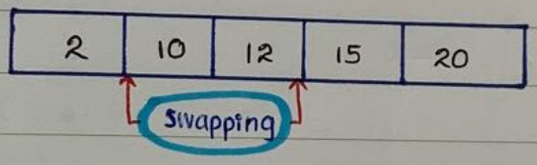


min value at index 2

i = 2

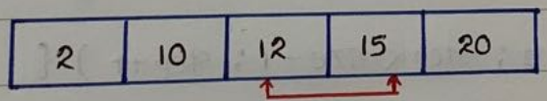


min value at index 2



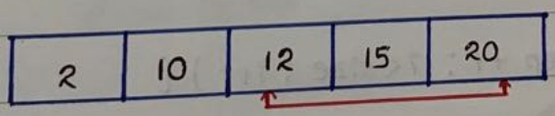
step = 2

i = 0

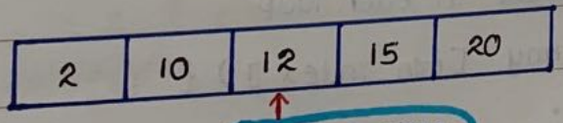


min value at index 2

i = 2

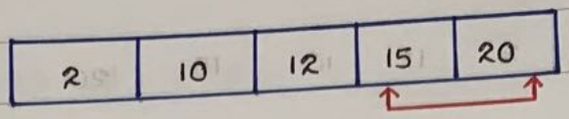


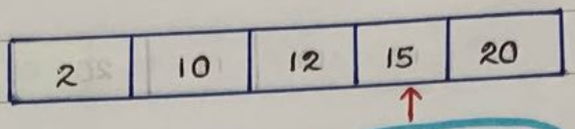
min value at index 2.



step = 3

i = 0





already in place

Selection Sort code in Java

```
import java.util.Arrays;

class SelectionSort {
    void selectionSort(int array[]) {
        int size = array.length;

        for (int step = 0; step < size - 1; step++) {
            int min_index = step;

            for (int i = step + 1; i < size; i++) {

                // select minimum element in each loop
                if (array[i] < array[min_index]) {
                    min_index = i;
                }
            }
        }

        // put min at correct position
        int temp = array[step];
        array[step] = array[min_index];
        array[min_index] = temp;
    }
}
```

```
psvm (String args[]){
    int[] data = {20,12,15,2};
    SelectionSort ss = new SelectionSort();
    ss.selectionSort(data);
    SOP("Sorted Array in Ascending Order:");
    SOP(Arrays.toString(data));
  }
}
```

0	1	2	3	4
10	31	15	8	16

last unsorted index - 4

current index - 0

Largest index - ø 1 (10) ← Imagine

last unsorted index - 4

current index - 1 (current & large compare වෙනවා)

Largest index - දැන් current index එකට වඩා large index එක වැඩිනම්

if current > largest ඒ ගාන ලියනවා.

last unsorted index - 4

current index - 2

largest index - (1) මේක එකකින් වැඩි කරනවා

if largest > current