Arrays and Strings

Wiki

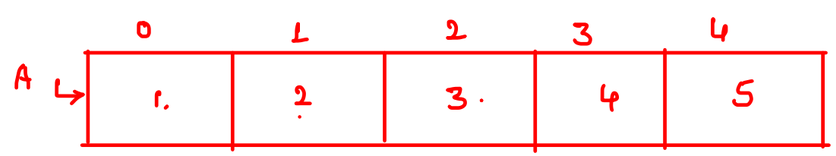
In computer science, an **array** is a data structure consisting of a collection of *elements* (values or variables), of same memory size, each identified by at least one *array index* or *key*. An array is stored such that the position of each element can be computed from its index tuple by a mathematical formula.[1][2][3] The simplest type of data structure is a linear array, also called one-dimensional array.

- Assume

  Size(int) = 4Bytes = 32 bits

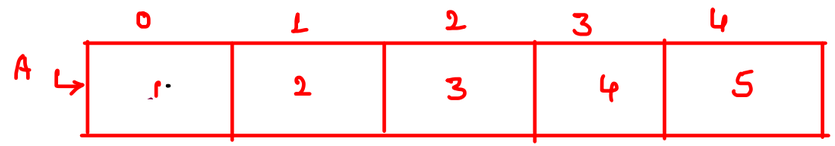
---

n = 5,



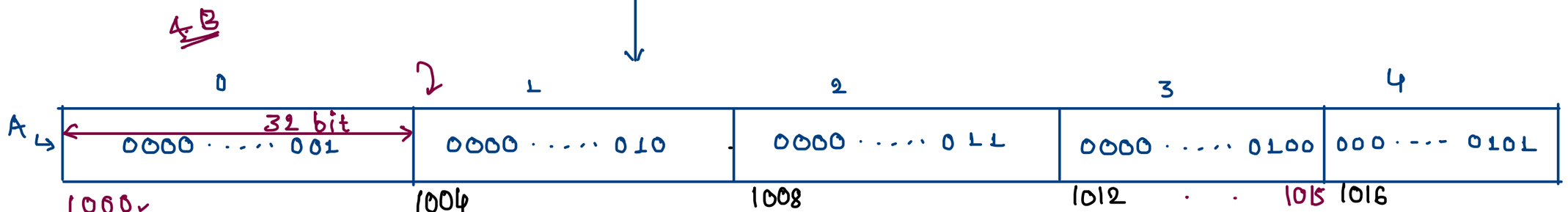
inside memory, How array will be stored?

0's and 1's



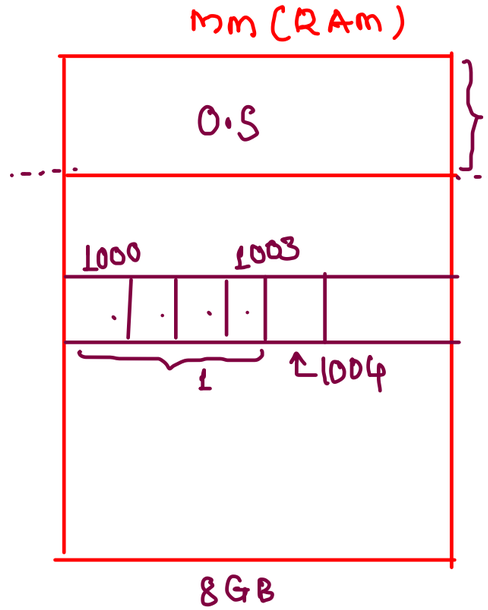
- Base address = 1000

↳ Assumption



1000
1001
1002
1003.

1005
:
1007



L.C → →

1. All the elements in the array MUST be of same type

what is use ?

*Imp

object

arr1=[1,"venu",12.5,'a'] // this not an array

// But programming languages like python, JS etc.. will allow those type arrays But that is not the standard definition of array

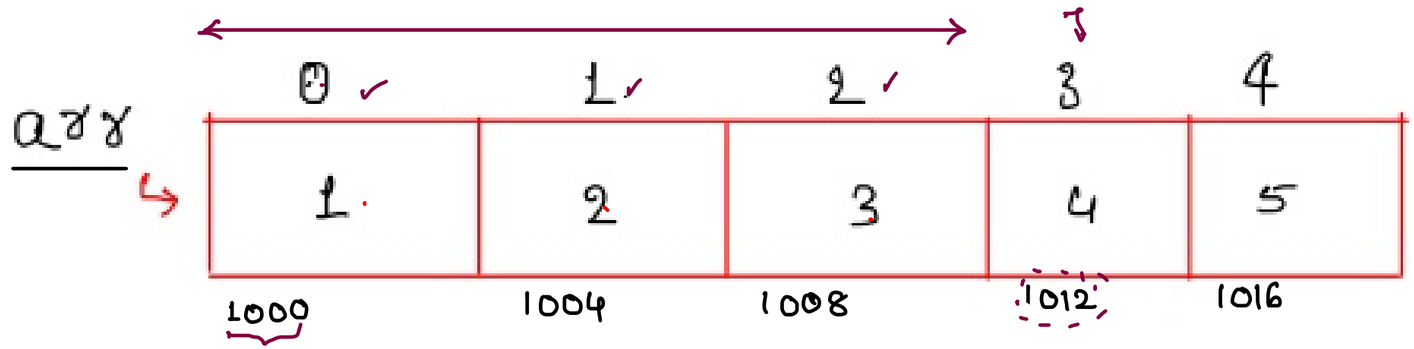
arr=[1, "venu", 10.5, 'a']

Standard Def:- All the elements must be of same type

Ex:

arr1= ['a', 'b', 'c', 'd'] ==> array of characters

arr2=["venu", "gopal", " masai", "pooja", "dhiraj"] ==> array of strings



B.A = Base Address

1000

Size (int) = 4 Bytes

2 print(arr[3]) = 4 ✓

Internally what will happen?

Step1:- Address computation

= B.A + (# of ele's needs to be skipped from starting) * sizeEle)

= 1000 + (3) x 4 = 1012

step2:- from 1012 address to 1015 address, printed

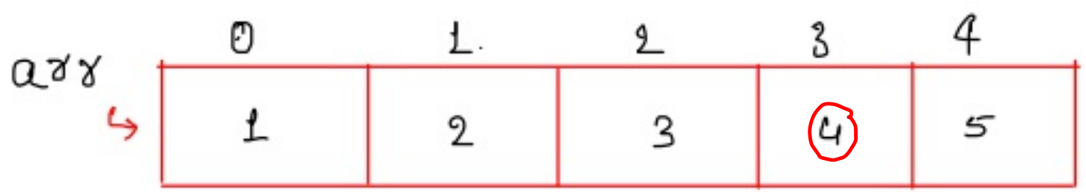
= 4

print(arr[5]) : UN-defined

= 1000 + (5) x 4

= 1000 + 20 = 1020 to 1023 } ∴ error

- Array:-
    1. All elements MUST be of same type



print(arr[3]) ==> 4

print(arr[5]) == > some kind of error

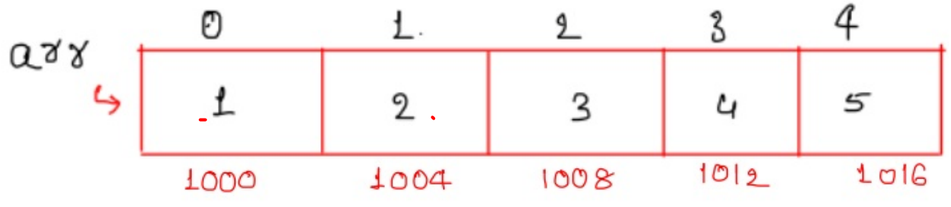
depends on. prog. language

How to access any particular element in array?

To access any element in the array, 1st it should know its

concept of address calculation

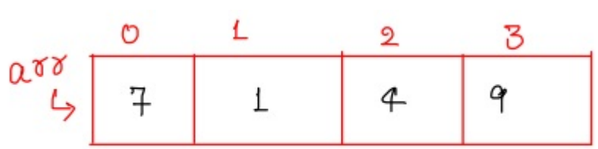
= Base_Address + [Number of Elements needs to be skipped]*size of element



address of arr[3] = 1012

Sub Array:

*A **subarray** is a contiguous part of array, i.e., Subarray is an array that is inside another array.*



i/p

n = 4

	0	1	2	3
arr →	7	1	4	9

Length(1) Sub. Arrays :-

4 SA's

0
7

1
1

2
4

3
9

Length(2) :-

3 SA's

0	1
7	1

1	2
1	4

2	3
4	9

Length(3) :-

2 SA's

0	1	2
7	1	4

1	2	3
1	4	9

Length(4) :-

1 SA.

	0	1	2	3
→	7	1	4	9

Length(5) :- Not possible

let

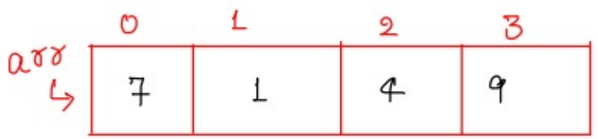
eles in array = 4

sub. arrays possible = L_1 SA + L_2 SA + L_3 SA + L_4 SA

$= 4 + 3 + 2 + 1 = 10$

---

in general, # of sub. arrays of different lengths (total) $= 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$



Sub-array: MUST be continious.

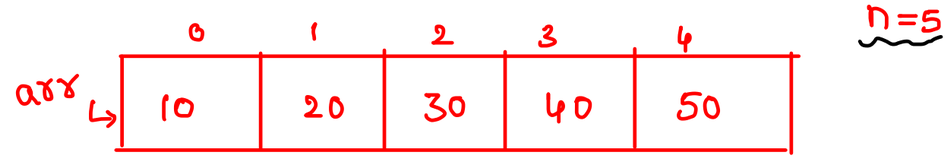
[1 4 9] ✓

[9 4 1] x

[7 1 9] x

Activity Time

For a Given array of size (n), how many subarrays are possible?



Length(1) : [10], [20], [30] . . . [50]	5 SA's
Length(2) : [10,20], [20,30], . . . [40,60]	4
Length(3) : [10, 20, 30], - - . [30,40,50]	3
Length (4) : [10, 20, 30, 40,], [20, 30, 40, 50]	2
Length (5) : [10, 20, 30, 40, 50]	1
	+
	1+2+3+4+5 = 15✓

in general: $1+2+\cdots+n = \frac{n(n+1)}{2}$ ✓

Ques) if an array is having n elements, then how many total number of sub arrays are possible ?

Ans) n*(n+1)/2 [this all the types of lengths, i.e len(1), len(2)..... len(n)]

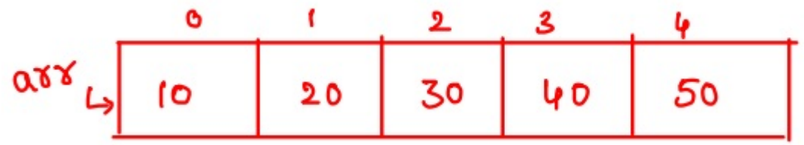
* Ques) if an array is having n elements, then how many number of sub arrays are possible of particular length k only?

given 'n', 'k'

n=10 elc, How many subarrays are having length 4? only

3-Nos general formula.

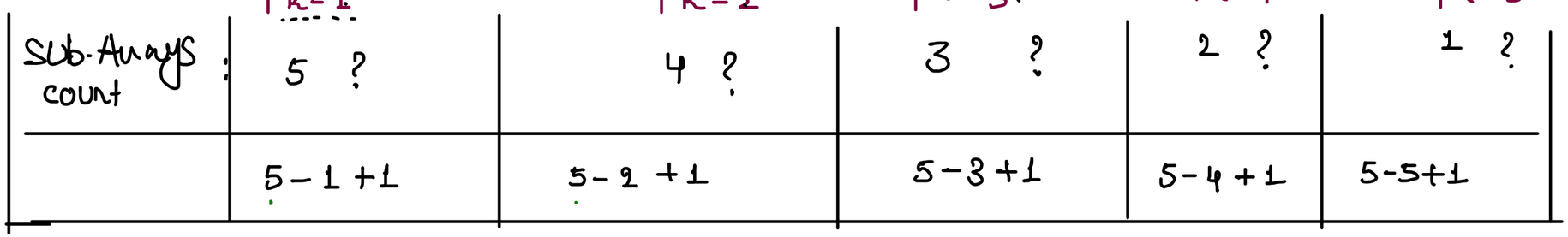
Eg:-



n=5

* we want only particular length(K) subarrays count.

given
n=5 ✓



Ans : n-K+1 ✓

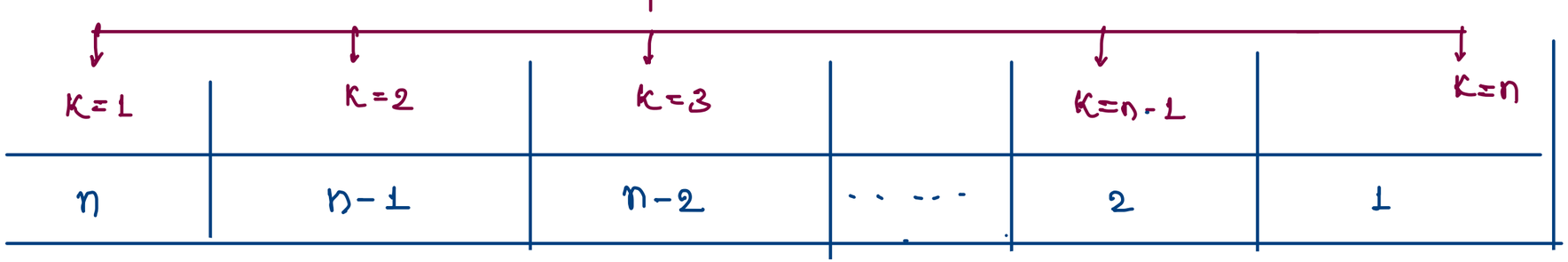
observation

given n, K

Sub. Array count -

Ans : n - K + 1 ✓

sliding - window



$\frac{n(n+1)}{2}$

$= \quad n + (n-1) + (n-2) + \cdots + 2 + 1$

$= \quad 1 + 2 + \cdots + n-1 + n \quad = \frac{n(n+1)}{2}$

if array is having n elements, then with length k, total n-k+1 sub arrays are possible

* Note:-

1) Given an array of size n, then total number of subarrays are possible = n(n+1) /2 . Which includes length-1 sub arrays, length-2 sub arrays and so on.... length(n) sub array also

2) Given an array of size n and How many number of sub arrays that are possible of particular lengh(k)?

ans : n-k+1

Sub sequence

Definition of a **Subsequence:**

A **subsequence** is a sequence that can be derived from another sequence by deleting some or no elements, without changing the order of the remaining elements. Importantly, a subsequence does not require the elements to be contiguous.

Key Points:

- A subsequence maintains the **relative order** of elements.
- The elements may or may not be consecutive.
- The original sequence is unchanged; some elements are simply removed.

0: delete

1: don't delete

$S = "a\ b\ c"$

- delete 0 (or) more characters.
- order must be maintained.

Sub. seq's of "abc"

	a	b	c	
1.	0	0	0	→ empty
2.	0	0	1	→ c
3.	0	1	0	→ b
4.	0	1	1	→ b.c
5.	1	0	0	→ a
6.	1	0	1	→ a.c
7.	1	1	0	→ a.b
8.	1	1	1	→ a b c

# of Non. Empty. sub. seq's in. general $= 2^n - 1$ (remove empty one)

$n = 3 \Rightarrow = 2^3 - 1 = 7$

Future:-

you will study about

“ ”

in this unit / module we gonna see the code for generating all the sub-array's

s = "abc"

empty
c
b
b.c
a
* a.c
a.b
abc

sub. seqs

F
T

✓

s = " abc"

All sub. strings = { a, b, c
ab, bc,
abc }

ac
↳ Not continous.

*Note:
- every. sub. seq's Need not be a sub. string.
- But every substring is a sub. seq's.

String:

In computer programming, a **string** is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation). A string is generally considered as a data type and is often implemented as an array data structure of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. *String* may also denote more general arrays or other sequence (or list) data types and structures.

s="venu"

s = "venu"

L. dedicated class.

probs on.

sub. array.

+

sub. string.

sub string :-

The following are substrings of string s

- Length(1) : v, e, n, u
- Length(2) : ve, en, nu

Length(3) : ven, enu

Length(4) : venu

Given string is having length(n) : then number of substrings are

_____

= n(n+1)/2

Sub Sequence : same as substring, Need not to be continous, must be non empty

Activity Time

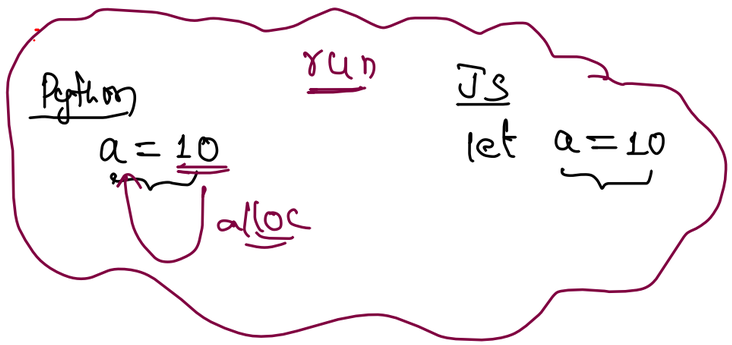
Primitives to Abstract Data Structure

Primitive Data Types : // System. Defined Datatypes.

↳ ex: int, long → integers (-10, 10, 2 . . .)

char → characters ('a', 'b', . . .)

float, double → floating point numbers (20.87, 10.4 .



wiki

In computer science, **primitive data types** are a set of basic data types from which all other data types are constructed.[1] Specifically it often refers to the limited set of data representations in use by a particular processor, which all compiled programs must use. Most processors support a similar set of primitive data types, although the specific representations vary.[2] More generally, "primitive data types" may refer to the standard data types built into a programming language (**built-in types**).[3][4] Data types which are not primitive are referred to as *derived* or *composite*.[3]

Primitive types are almost always value types, but composite types may also be value types.[5]

integer / float / char / ...

xboolean?

C
C++
Java
...

1. compile
2. execute

int, float, char...

- **Arrays:-**
  - ↳ derived from primitive data type.

one value → `int a = 10` → 4Byte

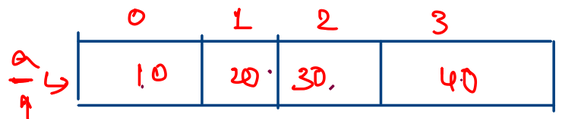
`int b = 20...`

  - 1. every element must be same type.
  - 2. elements has to be stored contigious mem. locations.

Array of 4 integers

`int a[4];` → type: array of integers.

Userdefined.



related to Data Structures:

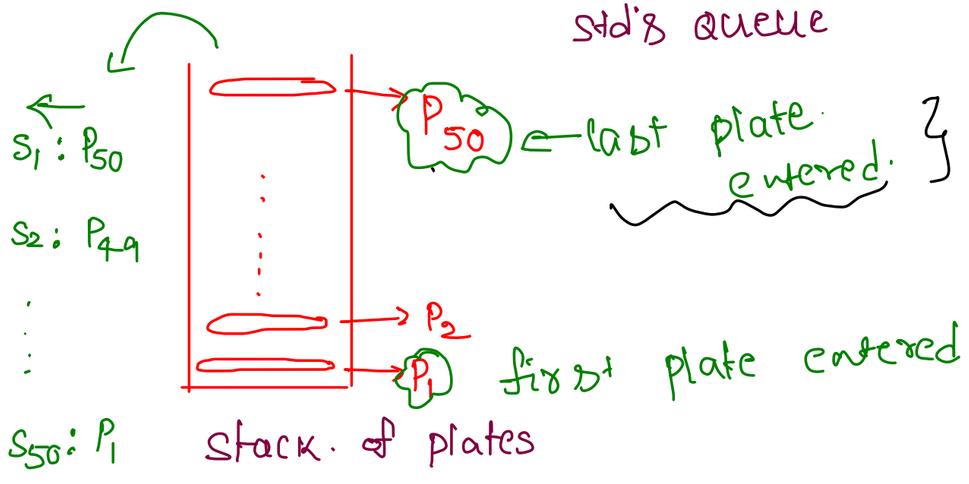
- In computer science, an **abstract data type** (**ADT**) is a mathematical model for data types, defined by its behavior (semantics) from the point of view of a *user* of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations. This mathematical model contrasts with *data structures*, which are concrete representations of data, and are the point of view of an implementer, not a user. For example, a stack has push/pop operations that follow a Last-In-First-Out rule, and can be concretely implemented using either a list or an array. Another example is a set which stores values, without any particular order, and no repeated values. Values themselves are not retrieved from sets; rather, one tests a value for membership to obtain a Boolean "in" or "not in".

first In First out → FIFO → Queue.

S_1 S_2 ... S_{50} ← entered

Std's Queue

out first P_{50}



Stack

Last In First Out → LIFO

(or)

First In Last out → FILO

Stack (abstract data type)

54 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

This article is about the abstract concept. For some concrete uses of the term in computing, see Stack (disambiguation) § Computing.

In computer science, a **stack** is an abstract data type that serves as a collection of elements with two main operations:

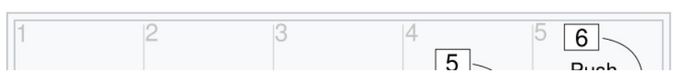
- **Push**, which adds an element to the collection, and
- **Pop**, which removes the most recently added element.

Additionally, a peek operation can, without modifying the stack, return the value of the last element added. The name *stack* is an analogy to a set of physical items stacked one atop another, such as a stack of plates.

Similarly to a stack of plates, adding or removing is only practical at the top.

The order in which an element added to or removed from a stack is described as **last in, first out**, referred to by the acronym **LIFO**.[nb 1] As with a stack of physical objects, this structure makes it easy to take an item off the top of the stack, but accessing a datum deeper in the stack may require removing multiple other items first.[1]

Considered a sequential collection, a stack has one end which is the only position at which the push and pop operations may occur, the *top* of the stack, and is fixed at the other end, the *bottom*. A stack may be implemented as, for example, a singly linked list with a pointer to the top element.



Appearance

Text
- Small
- Standar
- Large

Width
- Standar
- Wide

Color (beta)
- Automat
- Light
- Dark

ASCII ✓

problems. } → problems

sub. array, sub. strings,
Implementation, Brute force.

---

sub. seq's: we see in Recursion chapter.