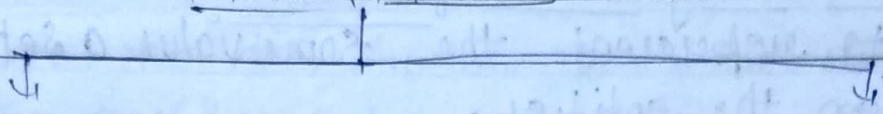
Dated :
Page :

5/8/19

Data structure



Built in Primitive DS

Non-Primitive DS

Integer Float Char Pointer

Array Linked list files

Linear Linked List

Non-Linear Linked List

Stack Queue Tree graph

* *P = Specific value
  &P = address

Unit : Basic Terminologies Elementary Data Organisation :

* Data entity and Information
* Primitive Data type
* Non Primitive data type
* Data structure : Definition, classification
* Data structure Operation

SBG STUDY

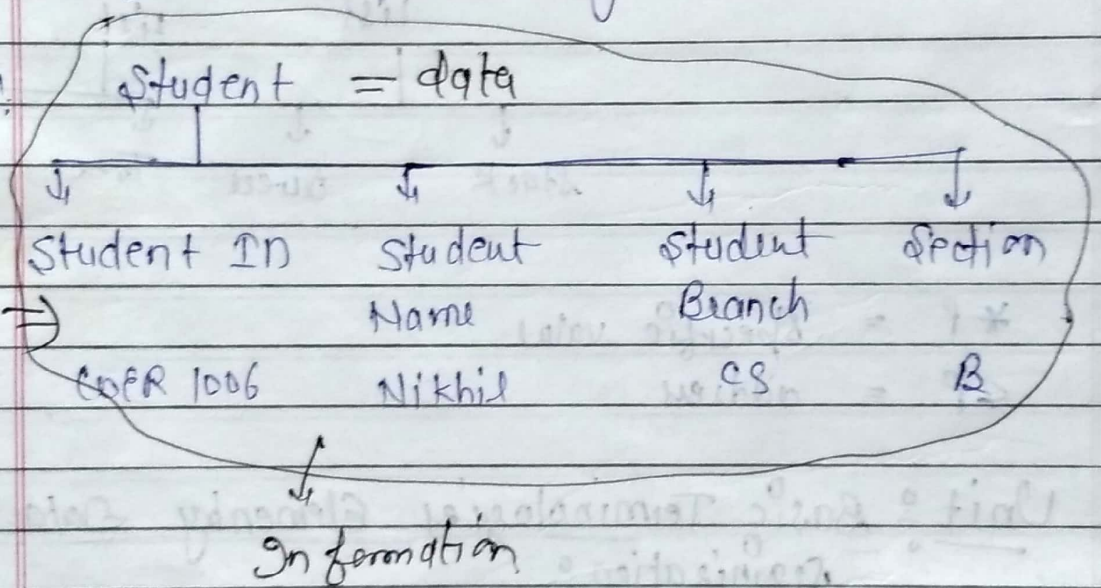
C + OS + DAA

Dated :

Page :

* Data, Entity and Information :

- Data represent the some value a set of value assign the entities.
- Data item is refers a single or group of values with in the data
- An entity is a thing has some properties which can take values.
- Process or meaningful data is called information for taking some action.



Dated :
Page :

6/8/19

(1) Primitive Data type :

- These are the data structure which are directly supported by the machine that is any operation can be perform on the data item.

- The diff. data types are
  int, float, double, character,
  boolean → T(1) / F(0)

* Non Primitive Data type :

- These data structure do not allow any specific instruction to be perform on the data item directly.

Ex: array

- The Non-Primitive DS are :
  Array, union, Structure

* Data Structure :

A DS is an arrangement of data in a Computer's memory or even Disk Storage. The
An Example of several common DS are :
arrays, LinkList, stacks, trees, Queues and hash table

Algorithms on the other hand are used to manipulate the data contain in these ds. as in searching and Sorting

When working with DS we need to know how to insert new the data, How a search for specific item and deleating the specific Item

Dated :
Page :

- Commonly used algorithm include
i) Searching for a particular item
ii) Sorting the data
iii) Traversing or ——— all the item in DS

Assignment

Q. Diff. b/w Recursion and iteration and Define type of recursion with Example

Ans:

Recursion	iteration
(1) Recursion achieves repetition through Repeated methods calls.	(1) Iteration explicitly uses repetition structure
(2) function call itself	(2) function Loops some part of code
(3) Recursion is selection structure	(3) Iteration is repetition structure
(4) Recursion terminates when a Base case is recognized	(4) Iteration terminates when the loop-continuation condition fail.
(5) Recursion return value to the calling function	(5) Iteration doesn't return value.
(6) Recursion make code smaller	Iteration make code longer
(7) Infinite recursion can crash the system	7) Infinite looping uses CPU cycles repeatedly
8) Recursion is slower process	8) Iteration is faster

Dated :
Page :

* Types of Recursion :

(1) Direct Recursion :

A function is called direct Recursion if it calls the same function again

Structure of direct recursion :

```
fun () {
    // some code
        fun();
    // some code
    }
```

(2) Indirect Recursion :

A function is called indirect recursion if it calls another function and then fun2 call fun directly or indirectly

```
fun() {
    // some code
        fun2();
    // some code
    }
```

```
fun2() {
    // some code
        fun();
    // some code
    }
```

(3) Tail recursion :

A recursive function is said to be tail recursion if the recursion call is the last thing done by the function. There is no need to keep record of the previous state.

Dated :
Page :

(4) Non tail Recursion:
A recursive function is said to be non-tail recursive if the recursive call is not the last thing done by the function. After returning back, there is some something left to evaluate.

5. What is the difference b/w malloc() and calloc()

malloc()	calloc()
1) Allocated block doesn't initialized	1) All elements are initialized by zero
2) It is one argument function	2) It is 2 argument function
3) It will provide higher performance compare to calloc() because it doesn't take extra time for initilizes	3) It is a bit slower process compare to malloc() because it will take extra amount of time for memory initialization.

6. What is the difference b/w structure and union

Structure	Union
1) A structure is define with keyword struct	A union is defined with keyword union
2) The compiler allocate the memory for each member of a variable which is associated with a structure	The compiler allocates the memory by considering the size largest data member

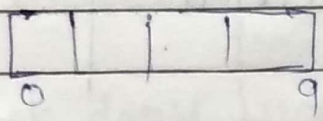
Dated :
Page :

3)	Each structure member is stored in different memory location	memory location is shared by union data member
4)	Altering the value of a member will not affect others member's individual member can be accessed	Only one data member can be accessed

Dated :
Page :

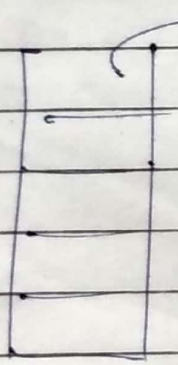
* Classification:
There are two types of data structure
i) Linear DS
ii) Non-linear DS.

(i) Linear DS: This data structure includes a arranging the element in linear fashion

Ex: Array 

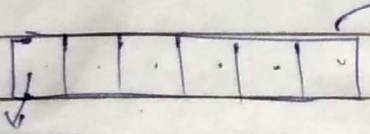
Syntax: length = upperbound - Lowerbound + 1

Declear:
int a[i]

(ii) Stack:


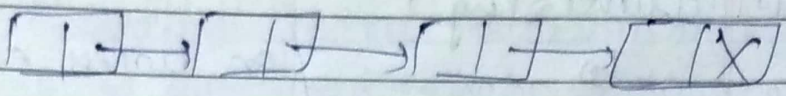
for Insertion = Push
for deletion = POP

if (Top == -1) empty.

(iii) Queue:


Dated :
Page :

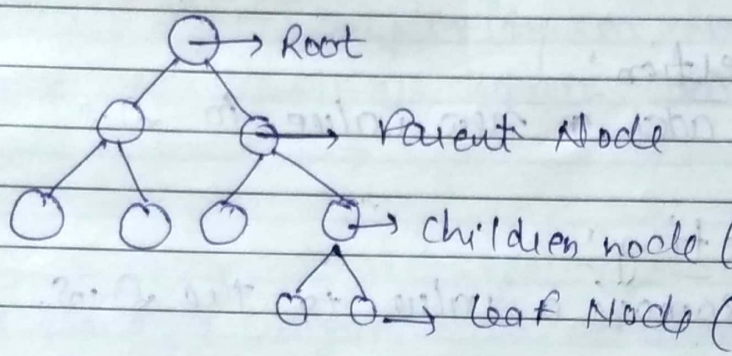
(iv) Linked List :-



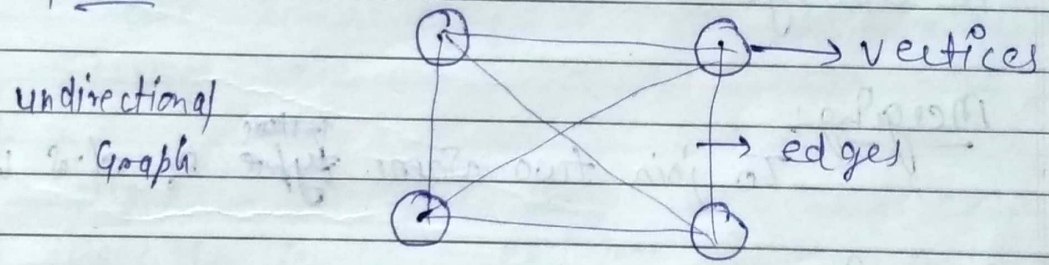
* (2) Non-linear ds :-

This d.s involves representing the elements in Graphical order

Ex :- Trees, Graphs.



Graph :-



* Data structure operation :-

operation means Processing the data in the data structure. The following are some imp operations

(1) Traversing :-
(2) Searching
(3) Insertion
(4) Deletion
(5) Sorting
(6) merging

Dated :
Page :

(1) Traversing:
To or Process each data exactly once in the D.S

(2) Searching:
To search for a Particular value in the D.S for the given value.

(3) Insertion:
To add a new value to D.S

(4) Deleting:
to Remove a value to the D.S.

(5) Sorting:
To arrange the value in the Particular order

(6) Merging:
To join two same ~~type~~ of d.s value

* Algorithm Analysis:

Complexicity theory, which provide theoritical expression for the required source of an algorithm to solve a specific computation problem.

most of the algorithm are ~~dg~~ designed to works inputs of arbitrary length.

P.S | S.S
Better Time Comp. ↓

Dated :
Page :

Analysis of Algorithm is the determination of the amount of time and space resources required to execute it.

Usually, the efficiency of running time of an algorithm is stated as a function relating the input n. to the no. of steps known as time complexity or volume of memory is known as space complecity.

*

Since multiple algorithm are available for solving the same problem

For Ex:- a sorting problem has many algorithm like insertion, selection, Quick sort and many more

Algorithm analysis helps us to determine which Algorithm is most efficient in terms of time and space consuming

Given to a algorithm for a task how to find out which one is better.

one navige
implement both the algorithm and Run two Programm on a computer for the diff. Input and see which one take less time.

O= w:B

O=

Ω=

Dated :

Page :

There are many problems with this approach for analysis of algorithm.

i) It might be possible that for some inputs first algorithm performs better than the second performs.

ii) It might also be possible that for some inputs first algorithm perform better of one machine and second works better on other machine for some other input

* Asymptotic Analysis is the big idea that handle above in analysing algorithm

In Asymptotic Analysis, we evaluate the Performance of Algorithm in terms of input size taken by an algorithm entries increases with the input size

Insertion Sort :

1) Linear time - O(1)
O(n)

2) Quadratic time - $O(n^2)$,

Dated :
Page :

* Assignment - 3 :

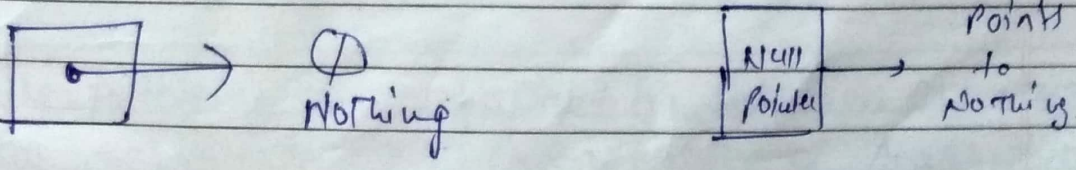
Q. Define Pointers. Discuss its various types with the help of Diagram

Ans: Dangling pointers
Void pointers
Null pointers
Void pointers
This pointer

Ans: Pointers: Pointer is a variable that stores the address of another variable
- Pointer saves memory space
- Execution time with pointer is faster because data is manipulated with the address i.e direct access to memory location
- The memory is accessed efficiently with the pointer is dynamically memory is allocated & deallocated

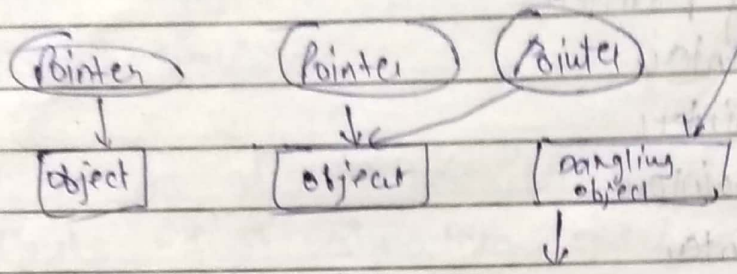
(1) Null Pointer:
Null pointer is a pointer which is pointing to nothing.
Null pointer points the base address of segment

Pointer which is initialized with NULL value is considered as null pointer



Dated :
Page :

(ii) Dangling Pointer :
Pointer pointing to a Non-existing memory location is called dangling pointer.



This dangling Pointer points to a de-allocated memory chunk that used to hold the original object

(3) Wild pointer :
A pointer in C that has been initialized till its first use is known as the wild pointer
A wild pointer points to some random memory location.

Dated :
Page :

* As Generally we perform the following type of Analysis
(1) worst case :
The max. no. steps taken on any instance of size A.

(2) Best case :
The min. no. of steps taken on any instance of size A.

(3) Average case :
An Avg. no. of steps taken on any instance of size A.

(4) Amortized :
The main idea of A sequence of operation applied to input of Science & average over time.

The main idea of Asymptotic Notation is to have a major of efficiency of Algorithm That does not depend on machine specific Constant does not Required Algorithm to be implimented and time taken by the program to be Compare

worst case → Average → Avg. worst case

Asymptotic Notations are mathematical tools to represent time complexity of algorithms for Asymptotic notation Analysis.

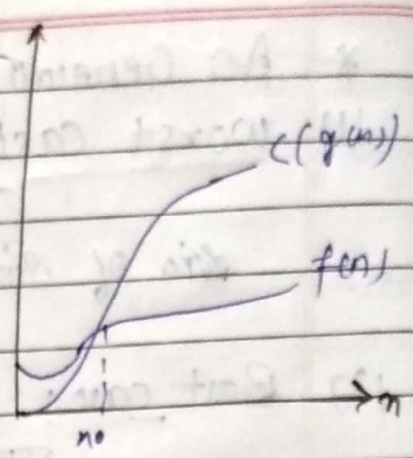
use hashtable, dictionary set & splay tree

The following three Asymptotic Notation are mostly to represent time complexity of Algorithm.

Dated :
Page :

* Big oh: [O]:

It shows the relationship b/w growth of time and input then in order to find the worst case of upper bound of the fn we use Big oh notation.



n = no. of i/p
t = time

Let us consider a function c(g(n)) growing in such a way that it is greater than the f(n) with limit n(0)

if $f(n) \leq c(g(n))$

$c > 0$, $n_0 \geq 1$ if Best case

$n \geq n_0$

$$f(n) = O(g(n))$$

It covers linear type as well as quadratic type

The Big oh notation is useful then we only have upper bound or time complexity of an algorithm

Dated :
Page :

Ex: $f(n) = 3n+2$ $g(n) = n$

$f(n) = O(g(n))$

$c > 0, \; n_0 \geq 1$

$f(n) \leq c(g(n))$ $\quad n \geq n_0$

$3n+2 \leq cn$

let $c = 4$

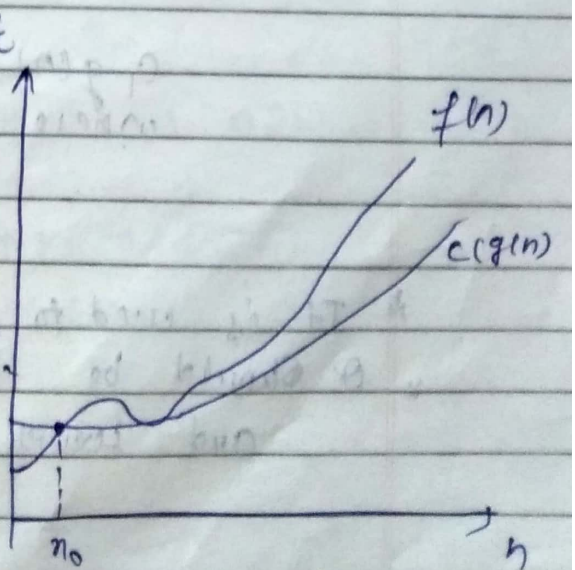
$3n+2 \leq 4n$

$n \geq 2$

So we found that for $f(n) = 3n+2$, $g(n) = n$ is the value which is always bound $3n+2$. it means if we apply any value of $g(n)$ i.e $n^2, n^3, n^4 \ldots 2^n$. will always bound $f(n) = 3n+2$ but we should choose the value which have tiest upper bound i.e Least greater Bound which is an ~~on~~ in this case.

* Big omega [Ω] :

Just as Big oh notation provide an asymptotic upper Bound on an function, omega notation provides an asymptotic lower bound



W3Schools.com

$3n+2$ $n \geq 1$

Dated :

Page :

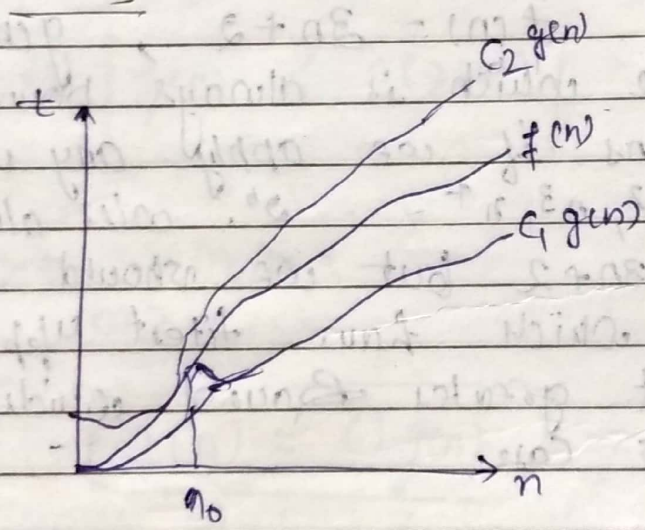
Ω Notation can be useful when we have lower bound on time complexicity of an algorithm of.

Since the Best Case of an algorithm is generally not use to, the Ω notation is the least used Notation among all three.

$$f(n) \geq c(g(n))$$

$$c > 0, \quad n_0 > 1, \quad n \geq n_0$$

* Theta :- [Θ]



$$C_1 g(n) \leq f(n) \leq C_2 g(n)$$

where $C_1, C_2 > 0$, $n_0 \geq 1$

$n \geq n_0$

* It is used to analyse average case
* Θ should be bounded by n both upper bound and Lower bound

Dated :

Page :

$f(n) = 2n^3 + 3n^2 + 5n + 10$

dominating term

$\theta(n^3)$

Que $f(n) = 3n + 2 \qquad g(n) = n$

find O

Let $c = 4$

$3n + 2 \leq 4n$

$-n \leq +2$

$\boxed{n \geq 2}$

* Practical significance of O, θ, Ω :

(i) O tells that in any case your time will not exceede the upper limit : it give the worst case time and space complexicity.

2) Ω (Big omega) tells it any case the Best time you can achieve . it give the Best Case.

3) θ tell the average . case

for ($i = 0$, $i <= 100$, $i++$) → $O(n)$

for ($i = 0$ $i^2 <= 100$, $i++$) → $O(\sqrt{n})$

Dated :

Page :

* Searching :

- Sequential (Linear) Search (for Random data order)
- Binary Search (when sorted)

* 1) Information with is one of most imp. application of computer

2) Frequently, it become necessary to search a list of records to identify a particular record

3) Usually each record contain of field whose value is unique, to distinguish among the records are known keys.

4) A particular record can be identify when the keep value of that record is equal to the given value. this operation is known as searching.

5) The searching algorithm are used to search or find one or more then one element, data set. These types of algorithm are used to find elements, form a specific data structure

Dated: _____
Page: _____

* Types of Searching:
The searching problem fall in two cases → Internal, External
if there are many Records
perhaps each one record be quite large
then it will be necessary to store the records in files on disk or tape, external to the primary memory. This case is called External searching.

then in the other case the records to be search are stored entirely in the primary memory There be only consider Internal searching

* Searching Techniques:

1) Sequential Searching:
The simplest way to do a search is to begin at one end of the List and scan down it until the target key is found or the other end is reached

- This search is applicable to a table that contain Random elements either as an array or as a Linked List

- This method of searching is also known as Linear searching

Best case ⇒ O(1)

Dated :

Page :

22/8/19

- The time taken on the no. of comparision made in searching a record in a search table determines the efficiency of the ~~se~~ search technique.

* Efficiency of LS :

- If the desired record is present in the first position of table then only one comparision is made. If the desire record in last one then n comparision have to be make.

- If the record is present somewhere in the search table, on an average the no of comparisions will be $(n+1)/2$

→ The worst case efficiency of this technique is O(n) stands for the order of execution.

* Algorithm :

Given : a is a linear array with n elements, x is a given item information

1. Set loc to 1
2. if loc > n then go to step 7
3. if A[loc] = x, then goto step 6.
4. set loc to loc + 1.
5. Go to step 2.
6. Print element x found at index loc and go to step 8.
7. Print element x is not found
8. Exit.

Dated :
Page :

* Binary Search :

(i) Sequential search is simple method for searching an element in an array having Random Element.

(ii) This is efficient for a small list of element But highly Inefficient for Longer List.

(iii) In the worst case we will have to make and comparision so as to search for the Last element to in the list.

(iv) Binary search is very efficient search technique which works for shorted list.

(v) In this method we make a Comparision b/w key and the middle element of the array

(vi) Since the array is shorted, this comparision result either in a match b/w key and the middle element of array on identifing Left half of the array or the Right half of the array to which desire element may belong.

(vii) In this case when the Current element is not equal to the middle element of the array, the procedure is Repeated on the half in which the desire element is likely to be present, Proceding in this way, either the element is found or the final division leads to a half consisting of no element.

Dated :
Page :

(viii) There may be ambiguity over the determination of the middle element of an array then the no. of elements is even. In this case, there are two element in the middle. However, an ~~ambiguity~~ arbitrary choice of any one of these as the middle element will serve the purpose.

(ix) The time complexity of Binary searching $O(\log_2 n)$

* Algorithm :-

1. Initialize segment variables
   Set beg = LB, end = UB and
   mid = int ((beg + end))/2

2. Repeat step 3 and 4 while
   beg <= end and a[mid] != item.

3. If item < a[mid], then
   Set end = mid - 1
   else
   set beg = mid + 1
   [end of structure]

4. set mid = int ((beg + end))/2

[End of step 2 loop]

Dated :
Page :

5. If a[mid] = item then
   Set Loc = mid
   else
   set Loc = ~~mid~~ NULL
   [End of if structure]

6. Exit.

SBG STUDY