১। .....পরীক্ষা.....সাল

২। বিষয়/কোর্সের নাম:.....

৩। সেমিস্টার/বর্ষ :.....Data structure.....

৪। প্রশ্নপত্র কোড :.....

৫। পরীক্ষার তারিখ : .....সকাল/বিকাল

(এ স্থান হতে উত্তর লেখা আরম্ভ করতে হবে)

Data type and data structures data structure operation.

The data structure is the way of organize the data in memory. There are many ways of organizing the data in the memory as we have already seen one of the data structure in c langua array is the collection of memory elements

The data structure is not any programming language like c, c++, java etc. It is a set of algorithms that we use in any programming language to structure the data in the memory

To structure data in memory n number of algorithm were proposed and all these algorithms are known as Abstract data types. These abstract data types are the set of rules.

Algorithm
↓
Abstract data types
↓
set of rules.

- Types of data structures.
  - Primitive data structure.
  - Non-primitive data structure.

* Primitive data structure:-

Primitive data structure are primitive data type. Primitive data types specify the size and type of variable values. The int, char, float, double, pointer, byte etc are primitive data structure.

* Non-primitive data structure:-
Non-primitive data structure is a data structure that allows you to store multiple data type values.
example, Array, Linked list, Stack etc.

The non-primitive data structure is divided into two types:
- Linear data structure.
- Non-linear data structure.

- Linear data structure.
  when one element is connected with 'n' number of elements known as Non linear data structure. The best example is trees and graph.

- Linear data structure:
  The arrangement of data in a sequential manner is known as linear data structure. Array, linked list, Stacks and Queues.

Data structure can also be classified

- Static data structure:-

It is a type of data structure where the size is allocated at the compile time. Therefore, the maximum size is fixed.

- Dynamic data structure.

It is a type of data structure where the size is allocated at run time. Therefore, the maximum size is flexible.

- Major Operation:-

1. Traversing: Accessing each record exactly once so that certain items in the record may be processed. This is also called visiting.
2. Seaching: Finding the location of the record with a given key value, or finding the locations of all records which satisfy one condition.

Inserting: Adding a new record to the structure.

Sorting: we can short the element according to acending or decending order.

Updation: we can also update the eleme we can replace the element with another element.

Deletion: we can remove the elemet from the structure using deletion operation.

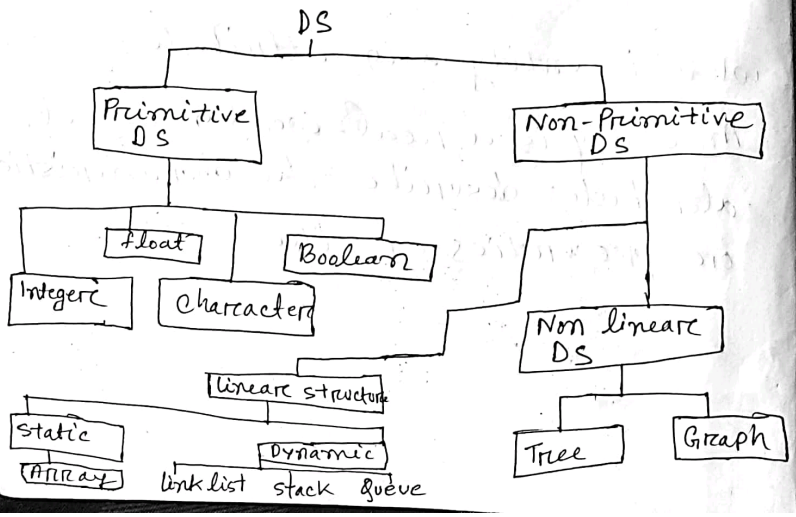
what is entity and attribute

An entity is a real-world object. attributes describe the characteristi or properties of the entity.

1. Field: A single elementary unit of information symbolizing the Attribute of an Entity is known as Field.

2. Record- the collection of different data items of field is known as record.

3. File: A collection of different Records of one entity type is known as file.



Dynamic data structure is

Link List: A Linked list is another example of linear ds used to store a collection of data elements dynamically. Data elements in this data structure are represented by the Nodes connected using links or pointers. Each ~~field~~ Node contain two fields. The information field consists of the actual data and the pointer fields consists the address of the subsequent nodes in the list. Linked lists can be classified into different types:

(a) Singly Linked List: A singly linked list the most common type of linked list. Each node has data and a pointer field containing an address to the next node.

(b) A information group two pointer fields, ~~doubly~~ are called Doubly linked list.

(c) Circular linked list: The circula
linked list is similare to the singly
linked list. The only key differen
is that the last node contain t
address of first node. forming a
circulare loop in the circular
linked list.

3. Stacks: LIFO - (Last In, First out)

A stack is a lineare data structure
follows the LIFO (Last In, First out)
principle that allows operation, insor
and deletation form one end of th
stack
the primary operations in the stack a
as follows:.

(a) Push: Operation to insert new elem
in the stack is termed as push o

(b) Pop: operation to remove or del
element from the ~~array~~
stack.

B 05103063

জাতীয় বিশ্ববিদ্যালয়

ক্রমিক নং ঃ .....

অতিরিক্ত উত্তরপত্র

১। ..... পরীক্ষা ..... সাল

২। বিষয় ঃ.....

৩। বিষয়ের শিরোনাম ঃ.....

৪। বিষয় কোড ঃ.....

৫। পরীক্ষার তারিখ ঃ.....

৬। ইনভিজিলেটরের স্বাক্ষর ও তারিখ ঃ .....

(এ স্থান হতে উত্তর লেখা আরম্ভ করতে হবে)

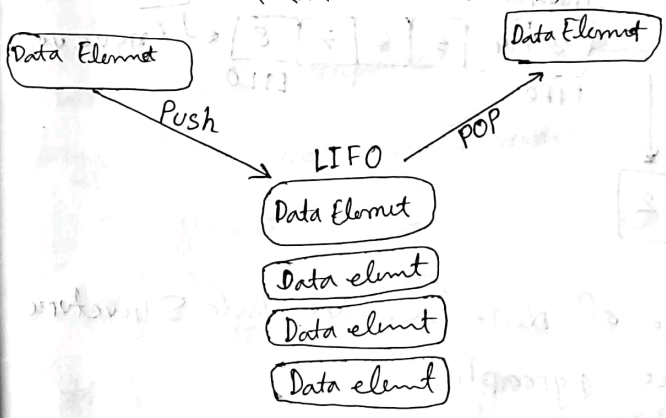


Figure A stack

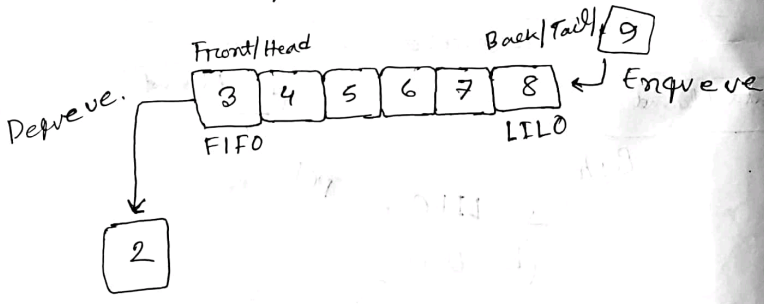
4. Queues: FIFO (First In First out)

A Queues is a linear data structure similare to a stack with some limit.

The primary operations of the Queue:

(a) Enqueue: The insertion or Addition of some element in data structure.

(b) Dequeue: Deleting or removing data element from the Queue is termed Dequeue.



Type of Non-linear Data Structure

1. Trees, graph.

Time complexity:-

Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.

For example

```
for (i=0; i<n; i++)
{
  printf("Hellow world");
}
```

Here, time-complexity is n

Algorithm Add two number. A, B and s

1. Input A and B
2. sum = A + B
3. Output sum.

The two numbers one addition operation and one assignment oper it is constant. so Time complexit

$T(n) = O(1)$.

Space complexity.

The space complexity of an algorithm quantifies the amount of space taken by an algorithm to run as a function of the length of the input. The amount of memory required by the algorithm to solve given problem is called space complexity of algorithm.

To estimate the memory requirement we need to focus on two parts:

(1) A fixed part: It is independent of the input size. It includes memory for instructions (code), constants, variables

(2) A variable part: It dependent on the input size. It includes memory for recursion stack, referenced variables.

example of time complexity the space complexity is

$S(n) = O(1)$.

LINEAR ARRAYS

A linear array is a list of a finite number of n ~~element~~ of homogeneous data elements.

(a) The elements of the array are referenced respectively by an index set consisting of n consecutive numbers

(b) The elements of the array are stored respectively in successive memory locations.

length = UB - LB + 1

Representation of linear array.

LOC(LA[K]) = Base(LA) + w(K - lower bo

where w is the number of words pe memory cell for the array LA.
K is the given any subscript.

30, 36, 25, 24, 35

We apply the bubble sort

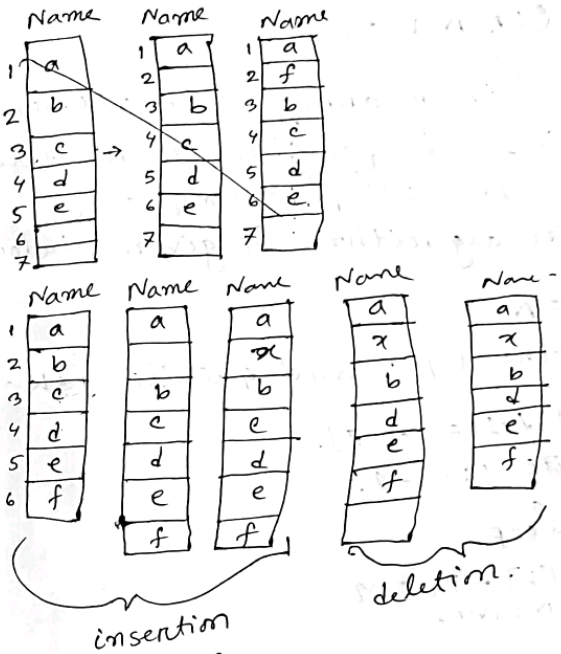
*

Insertion

let A be a collection of data element in a computer. Inserting refers to the operation of adding another element to the collection A.

Inserting element at the end of a linear array can be easily done provided the memory space allocated for the array is large enough to to accommodate the additional element. On the other hand, suppose we need to insert element in the middle of the array. Then on the average half of the elements must be moved downward to new location and

Deleting refers to the operation removing elements from A.



Insertion algorithm.

Insert(LA, N, K, ITEM)

Here LA is a linear array with the size N. If the position is k^{th} and inserting element is ITEM. The the insertion algorithm is

1. set J := N.
2. Repeat the process step 3 and 4 while J >= K

Insert (LA, N, K, ITEM)

LA is a linear array with the element N and k is the position. Insert the ITEM to the k^{th} position. The insertion algorithm is given below.

1. set J := N
2. Repeat the step 3 and 4 while J ≥ K:
3. set LA[J+1] := LA[J].
4. set J := J-1.
- End the loop.
5. set LA[K] = ITEM.
6. Exit. N = N+1.

Delation (Deleting an element from a linear array).

Delet (LA, N, K, ITEM)

LA is a Linear array with N elem
K is the position, where item is
avaible. Delete the ITEM from
K^{th} position.

B 0514786 18

জাতীয় বিশ্ববিদ্যালয়

ক্রমিক নং ঃ .....

অতিরিক্ত উত্তরপত্র

১। ..... পরীক্ষা ..... সাল

২। বিষয় ঃ .....

৩। বিষয়ের শিরোনাম ঃ .....

৪। বিষয় কোড ঃ .....

৫। পরীক্ষার তারিখ ঃ .....

৬। ইনভিজিলেটরের স্বাক্ষর ও তারিখ ঃ .....

(এ স্থান হতে উত্তর লেখা আরম্ভ করতে হবে)

1. set ITEM := LA[K].
2. Repeat ~~the steps 3 and 4~~ for J := K to N-1.

LA[J] = LA[J+1].
End the loop.

3. N = N-1.
4. Exit.

Bubble sort.

12, 15, 13, 17, 16, 15

We apply the bubble sort to the array A. we sort the element assending order. we discuss the process using pass separately

Pass 1.

(a) compare 12 < 15, the list is not altered

(12), (15), 13, 17, 16, 15

(b) compare 15 > 13. interchange 13, 15.

12, (13), (15), 17, 16, 15

(c) compare 15 < 17, the list is not alter.

12, 13, 15, (17), (16), 15

(d) 17 > 16 interchange 16 and 17

12, 13, 15, (16), (17), 15

(e) compare 17 > 15. interchage, 15, 17

12, 13, 15, 16, (15), (17)

Now 17 is fixed.

Pass - 2 with element N-1.

(a) (12), (13), 15, 16, 15, 17

(b) 12, (13), (15), 16, 15, 17

(c) 12, 13, (15), (16), 15, 17

(d) 12, 13, 15, (15), (16), 17

Pass is going on N-1 times.

then sort all the elements and we get.

12, 13, 15, 15, 16, 17

The bubble sorting algorithm is

BUBBLE (DATA, N)

Here DATA is an array with the element N. The algorithm sorts the elements in DATA.

1. Repeat step 2 and 3 for K = 1 to N-1
2. set PTR := 1.
3. while Repeat while PTR ≤ N-K:
   (a) If DATA[PTR] > DATA[PTR+1] interchange DATA[PTR] and DATA[PTR+1] set.
   (b) PTR := PTR+1.

   End the inner loop.

End the outter loop.

4. Exit.

Binary Search

Binary search is a search algorithm that works by repeatedly dividing the search interval in half. It is very efficient algorithm for sorted array. The binary search algorithm is time complexity is $O(n)$ or $\log(n)$.

$O(\log n)$.

The binary search.

11, 12, 13, 14, 15, 16, 17, 18, 19, 20

BEG = 1, END = 10 and $MID = int\left(\frac{1+10}{2}\right)$

$= 5$

DATA[MID] = 15 is not

ITEM (16) > DATA[MID] (15)

BEG = MID + 1.

BEG = 05 + 1 = 06

$MID = int\left(\frac{6+10}{2}\right) = 8$

DATA[MID] = 18

ITEM < 16 < 18

END = MID − 1 = 8 − 1 = 7

$MID = int(((BEG + END)/2)$

$= int((6+7)/2) = 6$

DATA[MID] = 16

16 is found at location 6

The binary searching algorithm.

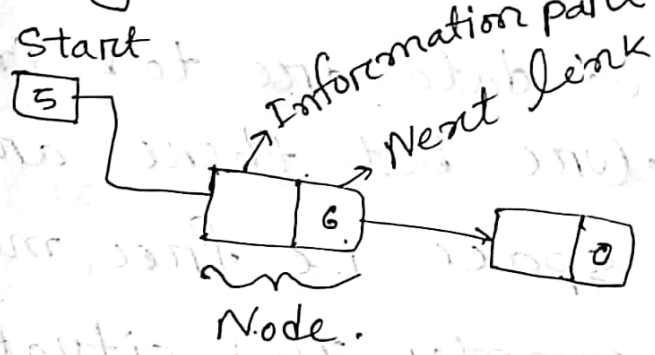
BINARY (DATA, LB, UB, ITEM, LOC)

1. Set BEG = LB, END = UB and MID = INT((BEG+END)/2
2. Repeat steps 3 and 4 while BGE ≤ END. and DATA[MID] ≠ ITEM.
3. If ITEM < DATA[MID], then:
   END := MID − 1.
   else:
   BEG := MID + 1.
4. Set $MID := int\left(\frac{BEG + END}{2}\right)$.
5. if DATA[MID] = ITEM
   set LOC := MID.
   else.
   LOC := NULL.
6. Exit.

$f(n) = O(\log n)$

Linked list.

A link list is a linear ~~array~~ collection of data elements are not stored at continuous memory location. The elements in a link list are linked using pointer.



What is Garbage collection?

Garbage collection is an automated dynamic memory management that identifies the dead memory block and reallocates storage for reuse.

It follows two steps

1. Garbage collection sequentially visit all the nodes in memory and mark all nodes which are being used in program.

2. Then it will collect all unmark nodes and place them in free storage area.

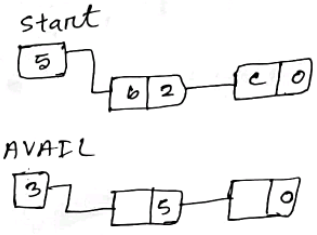
Overflow:-
Some-times a new data are to be inserte into a data structure but there are no ~~ab~~ available space i.e free memory storage list is empty. This situation is usually called overflow.

Under-flow:-
Underflow refers to the situation where one wants to delete data from a data structure that is empty.

Insertion algorithm:

~~1. [OVERFLOW]. If AV START,~~
Insert(INFO, LINK, AVAIL, ITEM)
~~Insert~~ ITEM insert at first position to the link list at START position. From AVAIL list the empity list is collected.
The insertion algorithm is given below.

1. [OVERFLOW]. If AVAIL = NULL. Write OVERFLOW, and Exit.
2. NEW := AVAIL and AVAIL = LINK[AVAIL].
3. Set INFO[NEW] := ITEM.
4. Set LINK[NEW] := START.
5. Set START := NEW.
6. Exit.



INSLOC(INFO, LINK, START, AVAIL, LOC, ITEM)

This algorithm inserts ITEM so that ITEM follows the node with location LOC or inserts ITEM as first node when LOC := NULL.

1. [OVERFLOW]. IF AVAIL := NULL Write overflow, and Exit.
2. Set NEW := AVAIL and AVAIL := LINK[AVAIL]
3. set INFO[NEW] := ITEM.
4. if LOC := NULL, then:
   set LINK[NEW] = START and START := NEW.
   else:
   LINK[LOC] = LINK[LOC], LINK[LOC] := NEW.
5. Exit.

Deleting algorithm

Algorithm which delete nodes from linked lists come up in various situations. The first one deletes the node following a given node and the second one deletes the node with a given ITEM of information.

DEL (INFO, LINK, START, AVAIL, LOC, LOCP)

The algorithm deletes the node N with location LOC. LOCP is the location of the node which precedes N or, when N is the first node, LOCP = NULL.

1. If LOCP := NULL, then:
   set START := LINK[START].
   Else:
   set LINK[LOCP] := LINK[LOC].
2. set LINK[LOC] := AVAIL and AVAIL = LOC
3. Exit.