Data Structures

Definitions

- data- means value or set of values.
- Data item is a single unit of value.A collection of data may be organizd into fields, records or file.
- Entity is something that has attributes or properties which maybe assigned values. Similar entities maybe grouped together to form an entity set.

Field is a single elementary unit of information representing an attribute of an entity.

Note: A collection of data maybe organized as records or file.

Record is the collection of field values of a given entity.

File is a collection of records of the entities in a given entity set. A value in a certain field which may uniquely determine the record in the file is called **Primary key.**

DATA STRUCTURE

It is a logical or mathematical model of a particular organization of data. It is a collection of related data and set of values for organizing and accessing it.

Note: The choice of a particular data model depends on the following:

1. It must be rich in structure to mirror the actual relationships of the data in the real world.
2. It must be simple enough that one can effectively process the data when necessary.

ALGORITHM

is a procedure in terms of the action to be executed and the order in which these actions are to be executed.

According to Niklaus Wirth,

Data Structure + Algorithms = Program

Complexity of Algorithms:

1. Time complexity- the no of steps executed by an algorithm.
2. Memory Complexity- the amount of memory needed in the execution of an algorithm.

Factors affecting the running time of a program:

1. The input to the program.
2. The quality of code generated by the compiler to create the object code.
3. The nature and speed of the instructions on the machine used to execute the program.
4. The time complexity of the algorithm underlying the program.

QUERY: Which of these factors a programmer can control?

Basic data structure operations:

1. Traversal- accessing each element exactly once.
2. Search - find the location of an element in the list.
3. Insert - add a new element to the structure.
4. Delete - remove an element from the structure.
5. Sort - arrange the elements logically.
6. Merging - combine the elements from different structures into a single structure.
7. Update - visit and apply changes to the structure.

Types of Data Structures

1. Arrays- group of contiguous memory locations that all have the same name and the same type. Its elements are stored consecutively.
2. Linked list- a linear collection of homogeneous data elements which are not necessary stored consecutively. Its elements are called NODES. Each node has 2 parts : INFO part and linkfield or nextfield pointer which contains the address of its next element. There is also a special value Start/Head that contains the address of the ist element in the list.

3. Stack- A linear collection of homogeneous elements wherein an element maybe added or removed from only one end called TOP. It is also called a LIFO structure. Push to insert and POP to delete.
4. Queue- A linear structure in which insertion and deletion are done at different ends of the list. This is called a FIFO structure.

5. Tree - a nonlinear structure which consists of a finite set of elements called nodes. If the tree in nonempty, it has a root but every other node can be reached from it by following a unique sequence of connective arcs.
6. Graph consists of a set of nodes(vertices) and a set of arcs(edges). Each arc in a graph is specified by a pair of nodes.

ABSTRACT DATA TYPE(ADT)

- it is an externally defined data that holds some kind of data
- Is a built in operation that can be performed on it and by it.
- Users of an ADT do not need to have any detailed info about the external representation of the data storage or implementation of the operation.
- The implementation maybe array or pointer-based.

ALGORITHM(TRAVERSING AN ARRAY)

A is a linear array with base address LB and last index UB. This algorithm traverses A applying an operation PROCESS to each element of A.

Steps:

```
1.  [Initialize Counter] Set K= LB.
2.  Repeat steps 3 and 4 while $K \leq UB$
3.  [Visit element] Apply PROCESS to
                        A[K].
4.  [Increment Counter] Set K =K+1
     [End of Step 2 loop]
5.   Exit.
```

ALGORITHM(Binary Search)

Steps:

```
1. [Initialize segment variables]Set First= LB, Last= UB and Mid= (First+Last)/2
2. Repeat 3steps 3 and 4 while First ≤Last and A[Mid] != ITEM
3. If ITEM<A[Mid] Set Last= Mid – 1
   else Set First =Mid+ 1[end of if structure]
4. Set Mid= (First+Last)/2
   [End of step2 loop]
5. If A[Mid]= ITEM set Loc =Mid
   else Set Loc = Null [End of if structure]
6. Exit
```

ALGORITHM(Linear Search)

Steps:

1. Assume the target has not been found
2. Start with the initial array element
3. Repeat while the target is not found and there are more array elements
4. If the current element matches the target
   Set a Flag to indicate that the target has been found
   else
   Advance to the next array element
5. If the target was found
       return the target index as the search result
   else
       return -1 as the search result.

A **Recursive Binary Search Algorithm.**

```
procedure binary search(x, i, j)
  m: =[(i + j)/2]
  if x =am then
     location: = m
  else if (x<am and i<m) then
     binary search(x, i, m-1)
  else if (x>am and j>m)then
     binary search(x, m+1, j)
  else location : = 0
```

ALGORITHM(Insert into an Array)

This algorithm inserts an ITEM into array A with N elements at position K where $K \leq N$.

Steps:

```
1.  [Initialize Counter] Set J=N
2.  Repeat steps 3 and 4 while J≥K
3.      [Move Jth element downward]Set A[J+1]=A[J]
4.      [Decrease Counter] Set J=J-1
5.  [End of step 2 loop]
6.  [Insert element] Set A[K]= ITEM
7.  [Reset N] Set N=N+1
8.  Exit
```

ALGORITHM(Delete an Item)

This algorithm deletes ITEM from array A with N elements, ITEM is at position K, $K \leq N$.

Steps:

```
1.    Set ITEM= A[K]
2.    Repeat for J= K to N-1
       [Move J+1st  element upward]
         Set A[J]= A[J+1]
      [End of Step2 loop]
3.    [Reset the no. of elements]
      Set N= N-1
```