Data Structures and Algorithms

Week 01 and 02

Data Structures

- Specific organization of data (Field, Record, File)
- Logical or Mathematical organization of data.
- A data structure
- is a (often non-obvious) way to organize information to enable efficient computation over that information

A data structure supports certain operations, each with a:

– Meaning: what does the operation do/return

– Performance: how efficient is the operation (5 5 5 5 5 = 5x5)

3 5 9 2 6 1 3 insert 3, 1 2 3 5 6 9

How to maintain the Efficiency of the operation/ process:

- Logical or Mathematical description of structures
- Implementation of structure on Computers.
- Quantitative Analysis.
    - Amount of memory required/ needed to store structure (Arrays, Pointers, Variable, Stack, Queue)
    - Time required to process the structure.

Examples:

– List with operations insert and delete

– Stack with operations push and pop

Efficiency and Effectiveness

Trade-offs

A data structure strives to provide many useful, efficient operations But there are unavoidable trade-offs:

– Time vs. space

– One operation more efficient if another less efficient

– Generality vs. simplicity vs. performance

Terminology

• ***Abstract Data Type (ADT)***

– Mathematical description of a "thing" with set of operations.

– Not concerned with implementation details

1 2 3 4 5 inset item 6

9 8 7 6 1 2 3 4 5 Stack (LIFO) 1 2 3 4 5 6 7 8 9 Queue (FIFO, FCFS)

Int a[10]

Tree, Graphs

• ***Algorithm***

– A high level, language-independent description of a step-by step process

• ***Data structure***

– A specific organization of data and family of algorithms for implementing an ADT

Solution is efficient if it is solved in available resources.

Available resources: Memory, Processor, **Time**

Hardware and Software two ways for any computational problem

Through Software--→ Data Structures, Algorithms.

Data Structure basics

Space for each data item

Time to perform basic operations

Characteristics of DS (Data Structure) - Time Complexity and Space Complexity

Why we need DS? Processor Speed (H/W issue), Data Searching, Multiple Request (web applications)

Phases (System Life Cycle):

Problem----identify problem
Requirements------- set specification, define purpose
Analysis----- Break the problem into manageable chunks, two approaches

- Bottom up Approach and Top Down Approach

Design-----ADT, Algorithm,
Coding-----representation of data
Verification/ Test---------Correctness proof.

Week # 03:

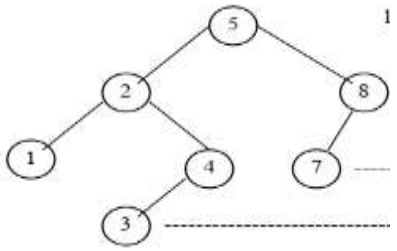
Data Structure Operations:

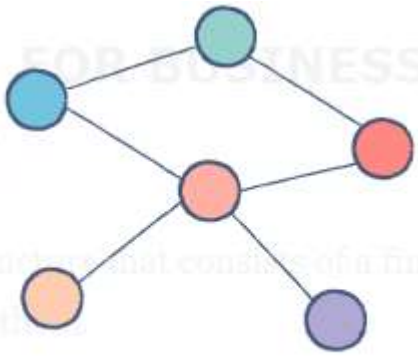
i. Traversing – to access each record/ field exactly once
To visit a record / field.
ii. Search – to find the location of the required record/ field.
iii. Insertion – Adding a new record / field into a structure
iv. Deletion – Deleting a field/ record
v. Sorting – to arrange data/ record/ filed in some logical order
vi. Merge – Combine to different record/ fields and make a single file

10	8	11	20	17	9	7	1	23	*13*	*5*
0	1	2	3	4	5	6	7	8	9	10

Linear DS: Arrays Link List, Stack, Queue

Nonlinear DS : Tree, Graph





Undirected

Stack:

It is an ADT and linear DS, works on the LIFO (Last in First out) principle
Insertion and deletion is from taken place at only one end
Stack is an ordered list.
Arrival order of elements: 5 10 3 8 15
Now delete element 15

8 (Top)
3
10
5

Queue:
It is an ADT and linear DS, works on the FIFO (First in First out) principle
Queue is an ordered list

10 (front)

3
8
15 (rear)

Insertion takes place at one end, called ‘Rear’ and deletion takes place from the other end, called ‘Front’.
Arrival order of elements: 5 10 3 8 15

“ Representation of Queue is more difficult than Stack”

Algorithm:

- Finite set of instructions that accomplishes a particular task
- Well defined set of steps for solving a particular problem
- Each algo have a particular Data Structure.

Complexity of an algorithm is the function which gives the running time and space

Characteristics of an Algorithm:

- Input/ Output – inputs and outputs should be clear.
- Finiteness- set of instruction.
- Independent – from any programming language
- Effectiveness – not unnecessary steps
- Unambiguous -all steps should be clear.
- Feasibility – available resources

Algorithm Example (Table Generation)

[this algorithm has the int variables N, L ANS]
1: Read N L (give inputs to program)
2: Set K as S or 0 (set value of K, as starting value)
3: Repeat Step 4 for K<=L (Loop)
4: While ANS = N * K (Condition)
    Set K: K + 1, or K=K+ 1
5: Print ANS or Result is ANS
6: Exit.
N = 5, L =5

Algorithm Example (find the largest number in an array and its location)

1: Set K: = 1, LOC:=1, MAX = DATA[1]
2: K:= K+1 (k=1; k>n; k+ +)
3: If K>N, then (N is total number of elements of DATA[])
    Write LOC and MAX and exit
4: If MAX <DATA[K], then
    LOC: = K, MAX = DATA[K]
5: repeat step 2.

3 2 1 5 4 8 0 6 7 9

"*A set of steps to accomplish or complete a task that is **described precisely** enough that a computer can run it"*

These algorithms run on computers or computational devices. For example, GPS in smartphones.

GPS uses shortest path algorithm. Online shopping uses cryptography which uses RSA algorithm.

Expectation from an algorithm

- Correctness: Algorithms must produce correct result, and checks how often it gives incorrect results (Accuracy Percentage)
- Approximation algorithm: Exact solution is not found, but near optimal solution can be found out. (Applied to optimization problem.)
- Less resource usage: Algorithms should use less resources (time and space).

Performance:

How long the algorithm takes : will be represented as a function of the size of the input.

f(n)→how long it takes if 'n' is the size of input.

How fast the function that characterizes the running time grows with the input size.

"Rate of growth of running time", the algorithm with less rate of growth of running time is considered better.

A: 3 2 1 5 4 8 0 6 7 9
B: 0 1 2 3 4 5 6 7 8 9……..99

Time: the number of key operations on n size data

n, log n, n^2, n^3, n log n

f(n), f(log n), f (n log n), f(n^3)

Strings:

Pattern Matching Algo-

Given string: dawood, has 6 characters

Req Pattern : food, character are 4

'T' be the text/ string, 'P' is the required pattern

T = 20, P length = 4

No of Max Cycles = T-P+1, max cycles occurred when the pattern is not matched.

6-4+1 = 3

No of Max Cycles= 20-4+1 = 17

T = xy xy xx ya (not considering the blank space), P = xxyx

Question: Find the cycles and total number of comparisons.

T = $(x)^{20}$, P = xx xy, find number of cycles and comparison in each cycle

No of Max Cycles = 20 – 4 +1 =17

T = $(x)^{20}$ = xx xx xx xx xx xx xx xx xx xx

Total comparison in each cycle = 4

Total comparison in all cycles , 17 * 4 = 68

In the above example, pattern is not found, it is the worst case scenario since maximum comparisons and max cycles.

Worst Case, Best Case, Average Case

The data size of the above example, n = P + T => T = n-P

Worst Case; C(n) = P (T-P+1) => 4 * 17 = 68

C(n) = P (n -2P +1) ->[Eq A], for max value of C(n) occurs when , P = (n+1)/ 4

C(n) = [(n+1)/ 4] [n – (n+1)/4 +1]

[after simplifying the above equation]

C(n) = $[(n+1)^2]$ / 8,

C(n) = $O(n^2)$ [*O* -> Bog Oh notation]

Q: Solve for the number of comparison and cycles

P = a a a, T = ab aa bb aa ab ba bb ab ba ba

Insertion Sort

(Excel file)

Selection ---→ Comparison ----→ Shift----→ Insertion

Link List:

It is a linear Data Structure or One way List

Linear order is maintained by the Pointers

Circular and doubly link list (excel file)

Stack

It is an example of Abstract Data Type

Works on LIFO principle
linear Data Structure

Basic Operations of stack, Push (insertion) and Pop (deletion)

==========

Expression: an expression is consist/ made of operands, operators and delimiters

Each operator has the priority/ precedence in expression == x = 4 * (4 +1)

Delimiters (braces) are used to increase the priority of operators.

Notations:

Infix: A+B

Prefix: +AB

Also called Polish Notation

A way to write an expression without parenthesis (braces) in which operators are place before operands

It is fundamental property of the polish notation, the order of operations are determined completely by the positions of operators and operands

A=2 B=3 C=5

(A+B)*C A+B*C

(2+3)*5	2+3*5
25	2+15 =17
*+ABC	+*CBA

A+B+C

++ABC

Postfix: AB+