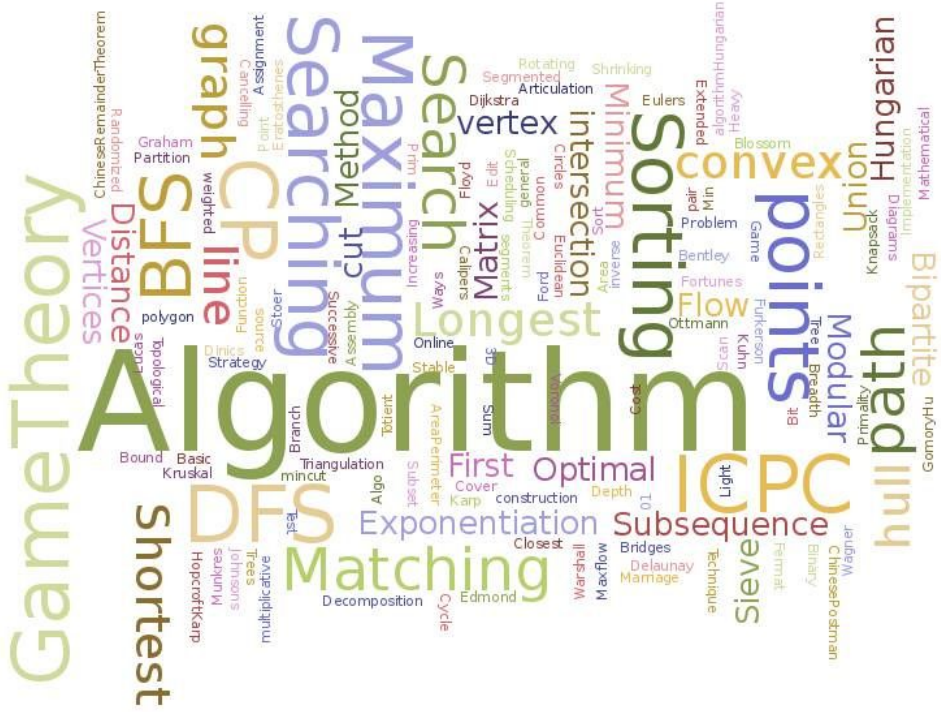

IIT BOMBAY

SUMMER OF SCIENCE

MAY 2019 - JULY 2019

NAME - PARTH . S . LATURIA

ROLL NO. - 180050071

MENTOR - THARUN TEJA ATLA

CONTENTS:

1. FOUNDATIONS

1.1 INTRODUCTION

1. An algorithm is any well-defined computational procedure that takes some set of values as input and produces some set of values as output.
2. A Data Structure is an efficient way to store and organise data in order to facilitate access and modifications.
3. Parallelism is a feature that allows the user to run multithreaded algorithms on a single machine simultaneously.
4. Algorithms are mainly used to optimize the running time and the memory space occupied by any program.
5. We prove the correctness of an algorithm by loop invariants(Initialization, maintenance and termination) and checking of pseudocode.
6. We analyse the algorithms by computing their running time,order of growth and doing their worst-case and average-case analysis.
7. Insertion sort is a common technique of sorting that involves an incremental approach.
8. Most of the algorithms that are recursive in nature typically follow the divide-and-conquer approach.
9. Merge sort involves divide-and-conquer approach in the form of recurrence relation.

1.2 GROWTH OF FUNCTIONS

1. Asymptotic notations are used to describe the order of growth of the functions and their respective running time.
2. There are 3 types of asymptotic notations: upper-bound, lower-bound and tight-bound.
3. For any integer a and any positive integer n, the value a mod n is the reminder of the quotient a/n.

1.3 DIVIDE-AND-CONQUER ALGORITHMS

1. A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

2. One of the common problems - “The maximum-subarray problem” can be solved in optimal time using divide-and-conquer approach.
3. We define an auxiliary function FIND-MAX-CROSSING-SUBARRAY and use it in the main function FIND-MAXIMUM-SUBARRAY in the above problem.
4. Strassen's algorithm that performs the matrix multiplication in less than O(n^3) time is also based on divide-and-conquer approach.
5. This topic also involves using substitution method, recursion tree method and master method for solving recurrences.
6. The substitution method involves guessing the form of the solution and then using mathematical induction to find constants.
7. In recursion tree method, each node represents the cost of a single subproblem somewhere in the set of recursive function invocations.
8. The master theorem finds the running time of a program based on the different possibilities of its functional time that are obtained on comparing it with related typical values.
9. This chapter also gives the proof of the master theorem: first assuming that the running time of function is defined only for exact powers and then extending it for generalization.

1.4 PROBABILISTIC ANALYSIS

1. Probabilistic analysis is the use of probability in the analysis of problems.
2. It is used to analyse the running time of an algorithm.
3. This chapter introduces the famous hiring problem that aims to find the cost that is needed to be paid in order to hire the best candidate in a sequence of candidates with different efficiency levels using prespecified selection rules.
4. Randomized algorithms are the algorithms whose behaviour is determined not only by its input but also by values produced by a random-number generator.

5. Indicator random variables are a type of symbols that provide a convenient method for converting between probabilities and expectations.
6. The birthday paradox problem : How many people must be there in a room before there is a 50% chance that two of them were born on the same day of the year?
7. Streaks problem: What is the longest streak of consecutive heads that you expect to see?
8. Both these problems are solved using probabilistic analysis.

2. SORTING AND STATISTICS

2.1 HEAPSORT

1. The heap data structure is an array object that can be viewed as a nearly complete binary tree.
2. There are two types of heap-property: max-heap property and min-heap property.
3. The max-heap property(which is generally used by default) states that the largest element in the heap is stored at the root and the subtree rooted at a node contains values no larger than that contained in the node itself (and vice-a-versa for min-heap).
4. The Heapsort Algorithm uses O(n log(n)) time to sort the given heap and uses the following functions:
5. BUILD-MAX-HEAP: produces a max-heap from an unordered input array
6. MAX-HEAPIFY: key to maintaining the max-heap property.
7. Priority Queue is a data structure for maintaining a set S of elements each associated with an associated value called a key. They can be implemented using :
8. MAX-HEAP-INSERT, HEAP-EXTRACT-MAX, HEAP-INCREASE-KEY and HEAP-MAXIMUM (these are for max priority queue and vice-a-versa for min-priority queue).

2.2 QUICKSORT

1. This sorting technique applies divide-and-conquer algorithm for sorting.
2. It has an average run-time of O(n log(n)) and the worst run time of O(n^2).
3. A randomized version of quicksort can be developed using random sampling of initial input.
4. Both the quicksort techniques involve a function to partition the array/sequence and find the pivot in the sequence recursively.

2.3 LINEAR TIME SORTING

1. A decision tree is a full binary tree that represents the comparisons between elements that are performed by a particular sorting algorithm operating on an input of a given size.
2. Heap sort and merge sort are asymptotically optimal comparison sorts.
3. Counting sort: it assumes that each of the input elements is an integer in the range 0 to k, for some integer k and then places an element x directly into its position in the output array by determining the number of elements less than x.
4. Radix sort: It is an algorithm used by the card-sorting machines you now find only in computer museums. It solves this problem by sorting starting from least significant digit to most significant digit.
5. Bucket Sort: It assumes that the input is drawn from a uniform distribution and has an average-case running time of O(n). It divides the interval [0,1) into n equal-sized buckets and then distributes the n input numbers into the buckets using some method.

2.4 MEDIANS

1. The ith order statistic of a set of n elements is the ith smallest element.
2. A median is the halfway point of the set.
3. The selection problem is : Given a set of n elements, say A and an integer i,output an element x such that x is larger than exactly i-1 other elements of the set A.
4. The RANDOMIZED-SELECT algorithm does the selection in expected linear time using RANDOMIZED-PARTITION algorithm.

3. DATA STRUCTURES

3.1 ELEMENTARY DATA STRUCTURES

1. A dynamic set is a set in which each element is represented by an object whose attributes can be examined and manipulated if we have a pointer to the object.
2. Stack is a type of data structure which follows the LAST-IN-FIRST-OUT principle.
3. Queue is a data structure which follows the FIRST-IN-FIRST-OUT principle.
4. Overflows and underflows can occur in a stack.
5. Insert and delete operations on a queue are called ENQUEUE and DEQUEUE.
6. A linked list is a data structure in which the objects are arranged in a linear order and the order is determined by a pointer in each object.
7. The operations like searching, inserting and deleting of elements can be done on every kind of data structure.
8. A sentinel is a dummy object that allows us to simplify boundary conditions.
9. All these data structures can be implemented using pointers and objects also.
10. Rooted trees can also be represented by linked list structure.

3.2 HASH TABLES

1. Hashing is an extremely effective and practical technique to store a set of data and supports various kinds of hash tables.
2. Direct-addressing is a simple technique that works well when the global set(universe) of keys(elements) is reasonably small.
3. Hash Table is a kind of data structure that stores a set of elements with the use of a hash function h to compute the slot for a key k as h(k).
4. The collisions in hash-table(two keys hashing to the same slot) can be resolved by chaining.

5. Some of the good hashing functions are: hashing by division,hashing by multiplication and universal hashing.
6. Open addressing is a technique in which all the elements occupt the hash table itself, i.e. each table entry contains either an element of the dynamic set or NIL.
7. This technique involves Linear probing, Quadratic probing and double hashing.

3.3 BINARY SEARCH TREES

1. It is a special kind of binary tree in which every element in the left subtree is less than or equal to the root and every element in the right subtree is greater than or equal to the root.
2. There are 3 kinds of tree walk that print the elements of the tree in a certain specific order:
3. Inorder tree walk -- left subtree->root->right subtree
4. Preorder tree walk -- root->left subtree->right subtree
5. Postorder tree walk -- left subtree->right subtree->root
6. The binary search tree supports functions like searching ,insertion,deletion,finding successor and predecessor of a given element,etc.
7. A randomly built binary search tree on n keys is one that arises from inserting the keys in random order into an initially empty key.

3.4 RED-BLACK TREES

1. It is a binary tree with one extra bit of storage per node : its colour which can be red or black.
2. Lemma : A red-black tree with n internal nodes has the height at most 2* log(n+1).
3. The rotation operation present on these search trees is one use to preserve its binary-search property.
4. A special kind of red-black trees i.e. AVL search trees are those that are height balanced i.e. for each node x, the heights of the left and right subtrees of x differ by atmost 1.

3.5 AUGMENTING DATA STRUCTURES

1. An order statistic tree, T is simply a red-black tree with additional information stored in each node x of the size of the subtree rooted at x.
2. The rank of an element is its position in linear order of the set.
3. We can retrieve an element of a given rank or we can also find the rank of a given element.
4. The purpose of augmenting a data structure is to support additional functionality in algorithm design.
5. A closed interval is an ordered pair of real numbers [t1,t2] with t1<=t2.
6. An interval tree is a red-black tree that maintains a dynamic set of elements,with each element x containing an interval x.int .

4. ADVANCED DESIGN AND ANALYSIS TECHNIQUES

4.1 DYNAMIC PROGRAMMING

1. Dynamic programming solves problems by combining the solutions to subproblems. They are generally applied to optimization problems.
2. Some of the problems that are solved by dynamic programming are:
3. Rod cutting problem: Given a rod of length n inches and a table of prices pi for i = 1,2,3,...,n, determine the maximum revenue rn obtainable by cutting the rod and selling the pieces.
4. Matrix chain multiplication: Given a sequence of n matrices A1, A2,....An , we wish to compute the product of these matrices.
5. Longest common subsequence: Given two sequences X and Y, we wish to find the maximum-length common subsequence of X and Y.
6. The basic elements and steps of dynamic programming are: characterizing structure of optimal solution, defining the value of an optimal solution, computing its value in a bottom-up fashion(memoization) and constructing an optimal solution from computed information.

4.2 GREEDY ALGORITHMS

1. The greedy algorithm is a technique that constructs the solution by making the choice that looks the best at the moment.Some of the problems that can be optimally solved by these algorithms are:
2. Activity-selection problem: Given a set S of n proposed activities that wish to use a resource, which can serve only one activity at a time, we wish to select a maximum-size subset of mutually compatible activities.
3. Greedy algorithm consists of both recursive and iterative methods.
4. Huffman code is an optimal prefix code constructed by greedy algorithm that stores the data in a compressed and efficient manner.
5. A matroid is an ordered pair M=(S,I) relating to graph theory.
6. Theorem : If G=(V,E) is an undirected graph,then Mg = (Sg,Ig) is a matroid.
7. All maximal independent subsets in a matroid have the same size.

4.3 AMORTIZED ANALYSIS

1. In an amortized analysis, we averaged the time required the time required to perform a sequence of data-structure operations over all the operations performed.
2. In aggregate analysis, we show that for all n, a sequence of n operations takes worst-case time T(n) in total.
3. Some of the examples that can be analysed using aggregate analysis are : Stack operations, incrementing a binary counter,etc.
4. Some of the methods that use amortized analysis are :
5. accounting method: Here we assign differing charges to different operations, with some operations charged more or less than they actually cost and do the further optimization steps.
6. Potential method: It represents the prepaid work as potential energy which can be released to pay for future operations.
7. Dynamic Table: the load factor alpha(T) of a nonempty table T is the number of items stored in the table divided by the size of the table.

5. ADVANCED DATA STRUCTURES

5.1 B-TREES

1. B-trees are balanced search trees designed to work well on disks or other direct-access secondary storage devices.
2. The primary memory and secondary memory of a computer mainly consist of silicon chips and magnetic disks.
3. B-tree supports operations like b-tree-search, b-tree-create, b-tree-insert and b-tree-delete.

5.2 FIBONACCI HEAPS

1. A mergeable heap is any data structure that supports the following five operations: make-heap, insert, minimum, extract-min and union.
2. A Fibonacci heap is a collection of rooted trees that are min-heap ordered i.e. each tree obeys the min-heap property.

5.3 vEB TREES

1. Some of the approaches for storing a dynamic set is direct addressing, superimposing a binary tree structure or superimposing a tree of constant height.
2. Proto vEB structure: it contains an attribute u giving its universe size . If u=2, then it is base size and it contains array A[0,10] of 2 bits, Else u=2^(2^k) for some integer k>=1 so that u>=4.
3. Some of the operations that can be done on proto vEB structure are: Determining whether a value is in the set, finding the minimum element of the set,finding the successor,inserting an element or deleting an element.
4. The vEB tree modifies the proto-vEB structure in a way that unless the universe size u (vEB(u)) equals the base size 2, the attribute summary points to a vEB(sqrt(u)) and the array A[0,1,... sqrt(u) -1] points to sqrt(u) * vEB(sqrt(u)).
5. All the operations possible on proto vEB structure are also possible on vEB structure.

5.4 DATA STRUCTURES FOR DISJOINT SETS

1. A disjoint set data structure maintains a collection S = {S1,S2,..Sk} of disjoint dynamic sets.
2. Each set is identified by some specific member of the set called a **representative.** The following operations should be supported on the set: make-set, union, find-set.
3. Linked-list representation is one of the most convenient ways to represent the disjoint data sets.
4. In a disjoint set-forest, each member points only to its parent and the root of each tree contains the representative and is its own parent.
5. We can achieve an asymptotically optimal disjoint-set data structure by using two heuristics- “union by rank” and “path compression”.
6. The amortized cost of each MAKE-SET operation is O(1).
7. The amortized cost of each LINK operation is O(alpha (n)).
8. The amortized cost of each FIND-SET operation is O(alpha (n)).
9. A sequence of m MAKE-SET, UNION and FIND-SET operations , n of which are MAKE-SET operations, can be performed on a disjoint-set forest with union by rank and path compression in worst case time O(m*alpha(n)).

6. GRAPH ALGORITHMS

6.1 ELEMENTARY GRAPH ALGORITHMS

1. There are 2 standard ways to represent a graph G=(V,E): as a collection of adjacency lists or as an adjacency matrix.
2. Adjacency lists are used to represent sparse graphs: those for which |E| is much less than |V|^2.
3. Adjacency matrix is preferred when the graph is dense i.e. |E| is close to |V|^2.
4. If edges have an attribute f, then we denote this attribute for the edge(u,v) by (u,v).f .
5. There are 2 types of simple algorithms for searching a graph:
   Breadth- first search: It assumes that the input graph is represented using adjacency list. This algorithm discovers all vertices at a distance k from the source before discovering any vertex at a distance k+1. Hence for any vertex v reachable from s, the simple path in the breadth-first tree from s to v corresponds to a shortest path from s to v in the graph G.
   Depth-first search: It assumes that the input graph is represented using adjacency matrix. Depth-first search explores edges out of the most recently discovered vertex v that has still unexplored edges leaving it.In this type of search, the predecessor subgraph forms a forest of trees. The discovering and finishing times in this search have parenthesis structure.
6. There are 4 kinds of edges in graphs: tree-edge, back-edge, forward-edge and cross-edge.
7. In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.
8. A topological sort of a dag G=(V,E) is a linear ordering of all its vertices such that if G contains an edge (u,v) then u appears before v in the ordering.
9. A strongly connected components of a directed graph G=(V,E) is a maximal set of vertices C <= V such that for every pair of vertices u and v in C, we have u -> v and v -> u.

6.2 MINIMUM SPANNING TREES

1. The minimum-spanning tree problem : Given an undirected graph G=(V,E) and weight w(u,v) for each edge (u,v) <= E, we wish to find an acyclic subset T<=E that connects all of the vertices and whose total weight w (T) is minimized.
2. A cut (S, V-S) of an undirected graph G=(V,E) is a partition of V.
3. There are 2 algorithms for solving this problem:
4. Kruskal's algorithm: here, the set A is a forest whose vertices are all those of the given graph.
5. Prim's algorithm: here, the set A forms a single tree. The safe edge added to A is always a least-weight edge connecting the tree to a vertex not in the tree.

6.3 SHORTEST PATHS

1. In shortest paths,we are given a weighted directed graph g=(V,E) with a weight function w:E -> R.
2. A critical path is the longest path through the dag, corresponding to the longest time to perform any sequence of jobs.
3. Some of the algorithms to solve this problem are: bellman-ford algorithm, dijkstra algorithm, floyd-warshall algorithm and johnson's algorithm.

6.4 MAXIMUM FLOW PROBLEM

1. A flow network G=(V,E) is a directed graph in which each edge (u,v) belongs to E has a non-negative capacity c(u,v) >=0.
2. A flow in G is a real-valued function f: V x V -> R that satisfies some capacity constraint and some flow constraint.
3. Given a flow network G = (V,E) and a flow f, the residual network of G induced by f is Gf = (V,Ef) where:
   Ef = {(u,v) belongs to V x V : cf(u,v) > 0}
4. If f is a flow in G and f' is a flow in the corresponding residual network Gf, we define f |^| f'(u,v) :
   - f(u,v) + f'(u,v) - f'(v,u) if (u,v) belongs to E
   - 0 otherwise
5. Pushing flow on the reverse edge in a residual network is known as cancellation.
6. Given a flow network G = (V,E) and a flow f, an augmenting path p is a simple path from s to t in the residual network Gf.
7. We call the maximum amount by which we can increase the flow on each edge in an augmenting path p the residual capacity of p, given by: cf(p) = min(cf(u,v) : (u,v) is on p}.
8. A minimum cut of a network is a cut whose capacity is minimum over all cuts of network.
9. Given an undirected graph G=(V,E),a matching is a subset of edges M belongs to E such that for all vertices v belongs to V , at most one edge in M is incident on v.
10. After a non-saturating push from u to v, the vertex u is no longer overflowing.

7. SELECTED MATHEMATICAL TOPICS

7.1 MULTITHREADED ALGORITHMS

1. Each thread maintains an associated program counter and can execute code independently of other threads.
2. Dynamic multithreading allows programmers to specify parallelism in applications without worrying about communication protocols, load balancing and other vagaries of static-thread programming.
3. We can describe a multithreaded algorithm by adding to our pseudocode just 3 keywords: parallel, spawn and sync.
4. Nested parallelism occurs when the keyword spawn precedes a procedure call.
5. The concurrency keywords express the logical parallelism of the computation.
6. The different kinds of edges of a computation dag are: continuation edge, spawn edge, call edge and return edge.
7. The work of a multithreaded computation is the total time to execute the entire computation on one processor.
8. The span is the longest time to execute the strands along any path in the dag.
9. The speedup of a computation on P processors is how many times faster the computation is on P processors than on 1 processor.
10. A multithreaded algorithm is deterministic if it always does the same thing on the same input, no matter how the instructions are scheduled on the multicore computer.
11. A determinacy race occurs when two logically parallel instructions access the same memory location and atleast one of the instructions perform a write.
12. Some of the examples of multithreading algorithms are multithreading matrix multiplication, multithreading strassen's algorithm and multithreading merge sort.

7.2 MATRIX OPERATIONS

1. If the number of equations is less than the number of unknowns, then the system is undetermined and has either infinitely many solutions or no solution.
2. If the system is overdetermined, then there may not exist any solution.
3. We can solve the system of linear equations by forward and back substitutions.
4. Matrix multiplication and matrix inversion have asymptotically the same running time.
5. Any positive definite matrix is nonsingular.

7.3 LINEAR PROGRAMMING

1. A linear programming problem is the problem of either minimizing or maximizing a linear function subject to a finite set of linear constraints.
2. There are 2 forms of linear programming : standard and slack.
3. Simplex algorithm takes as input a linear program and returns an optimal solution.
4. A linear program in standard form is optimization of a linear function subject to linear inequalities.
5. A linear program in slack form is optimization of a linear function subject to linear equalities.
6. Two maximization linear programs L and L' are equivalent if for each feasible solution x to L with objective value z, there is a corresponding feasible solution x to L' with objective value z and vice versa.
7. Slack means the difference between the left hand side and the right hand side of the given linear equation.
8. Shortest paths,maximum flow, minimum-cost flow and multicommodity problems can be modelled and solved using linear programming.

9. Linear programming duality enables us to prove that a solution is indeed optimal.
10. **Fundamental theorem of linear programming:** Any linear program L, given in standard form either has optimal solution with finite objective value, is infeasible or is unbounded.

7.4 POLYNOMIALS

1. A polynomial in the variable x over an algebraic field F represents a function A(x) as a formal sum :
   A(x) = a0*x^0 + a1*x^1 + … + a(n-1)*x^(n-1).
2. A coefficient representation of a polynomial A(x) of degree bound n is a vector of n elements.
3. A point-value representation of a polynomial A(x) of degree-bound n is a set of n point-value pairs.
4. A complex nth root of unity is a complex number w such that w^n = 1.

7.5 NUMBER-THEORETIC ALGORITHMS

1. An algorithm with integer inputs a1,a2,...,ak is a polynomial time algorithm if it runs in time polynomial in lg(a1),lg(a2),...,lg(ak) i.e. polynomial in the lengths of its binary-encoded inputs.
2. For any integer a and any positive integer n, there exist unique integers q and r such that 0<=r<n and a=q*n + r.
3. The greatest common divisor of two integers a and b is the largest of the common divisor of a and b.
4. Two integers a and b are relatively prime if their only common divisor is 1.
5. For all primes p and all integers a and b, if p|ab, then p|a or p|b or both.
6. For any non-negative integer a and any positive integer b, gcd(a,b) = gcd(b,a mod b)
7. A group (S,#) is a set S together with a binary operation @ defined on S for which the following properties hold: closure, identity, associativity and inverse.

8. If a group (S,#) satisfies the commutative law a#b = b#a, then it is an abelian group.
9. The system (Zn,+n) and (Zn,*n) are finite abelian groups.
10. If n is an odd composite number, then the number of witnesses to the compositeness of n is atleast (n-1)/2.

7.6 STRING MATCHING PROBLEM

1. The string matching problem is the problem of finding all valid shifts with which a given pattern P occurs in a given text T.
2. The concatenation of two strings x and y, denoted xy has length |x| + |y| and consists of characters from x followed by the characters from y.
3. Some of the algorithms for solving the problem are: naive string-matching algorithm, rabin-karp algorithm and knuth-morris pratt algorithm.
4. A finite automaton M is a 5-tuple(Q,q0,A,#,d) where:
    - Q is a finite set of states.
    - Q0 belongs to Q is the start state.
    - A belongs to Q is a distinguished set of accepting states.
    - # is a finite input alphabet.
    - d is a function from Q x # into Q, called transition function of M.