String Matching, Approximation Algorithms and Theory of NP Completeness

Unit: 5

Design & Analysis of Algorithms ACSE0401

B.Tech 5th Sem

Parul Goel

Assistant Professor
CSE Department

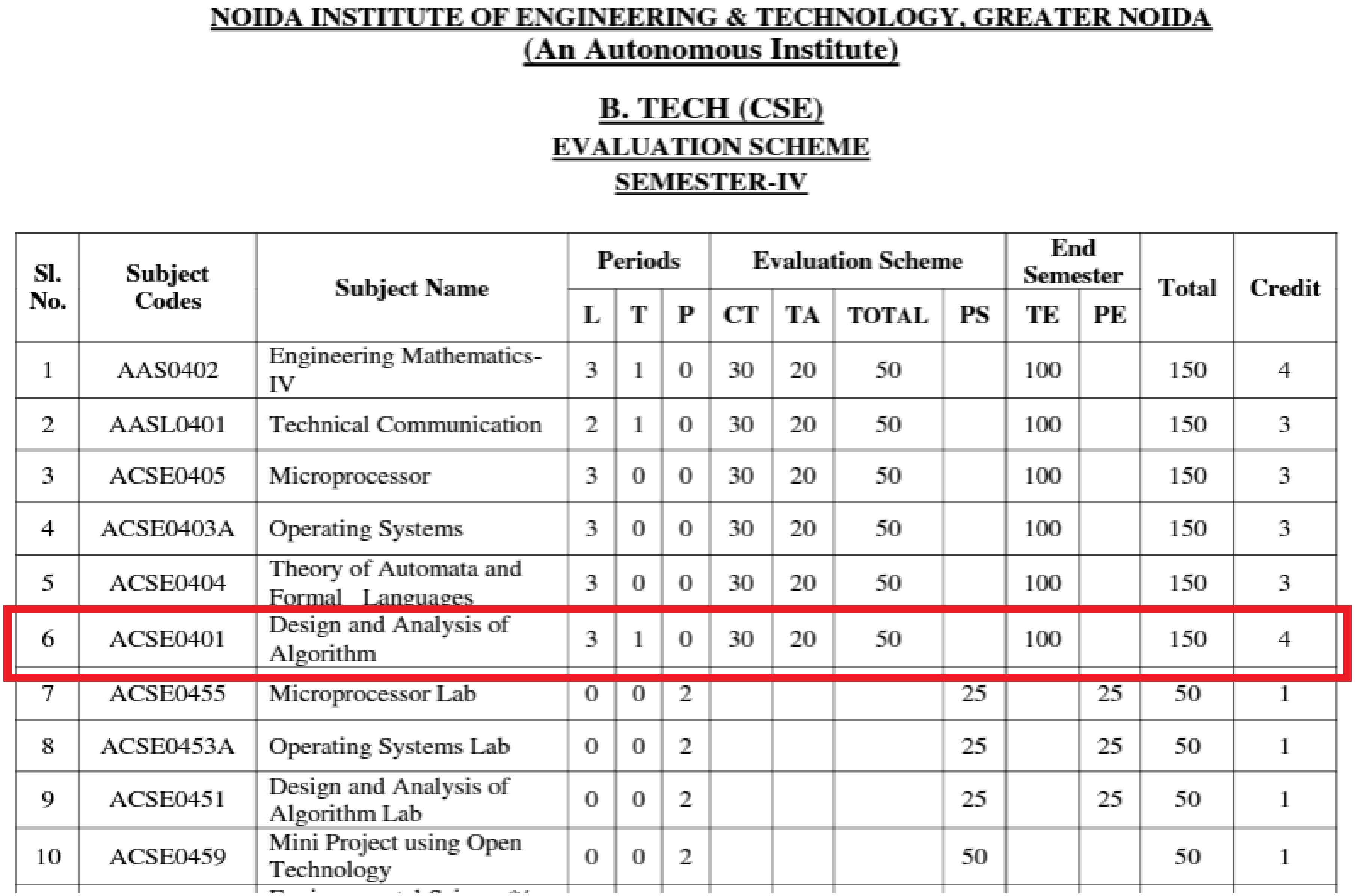
Brief Introduction About Me

Ms. Parul Goel
Designation : Assistant Professor CSE Department
NIET Greater NOIDA
Qualification:

- B.Tech.(CSE)
- M. Tech.(CSE)

Teaching experience : 03

Evaluation Scheme

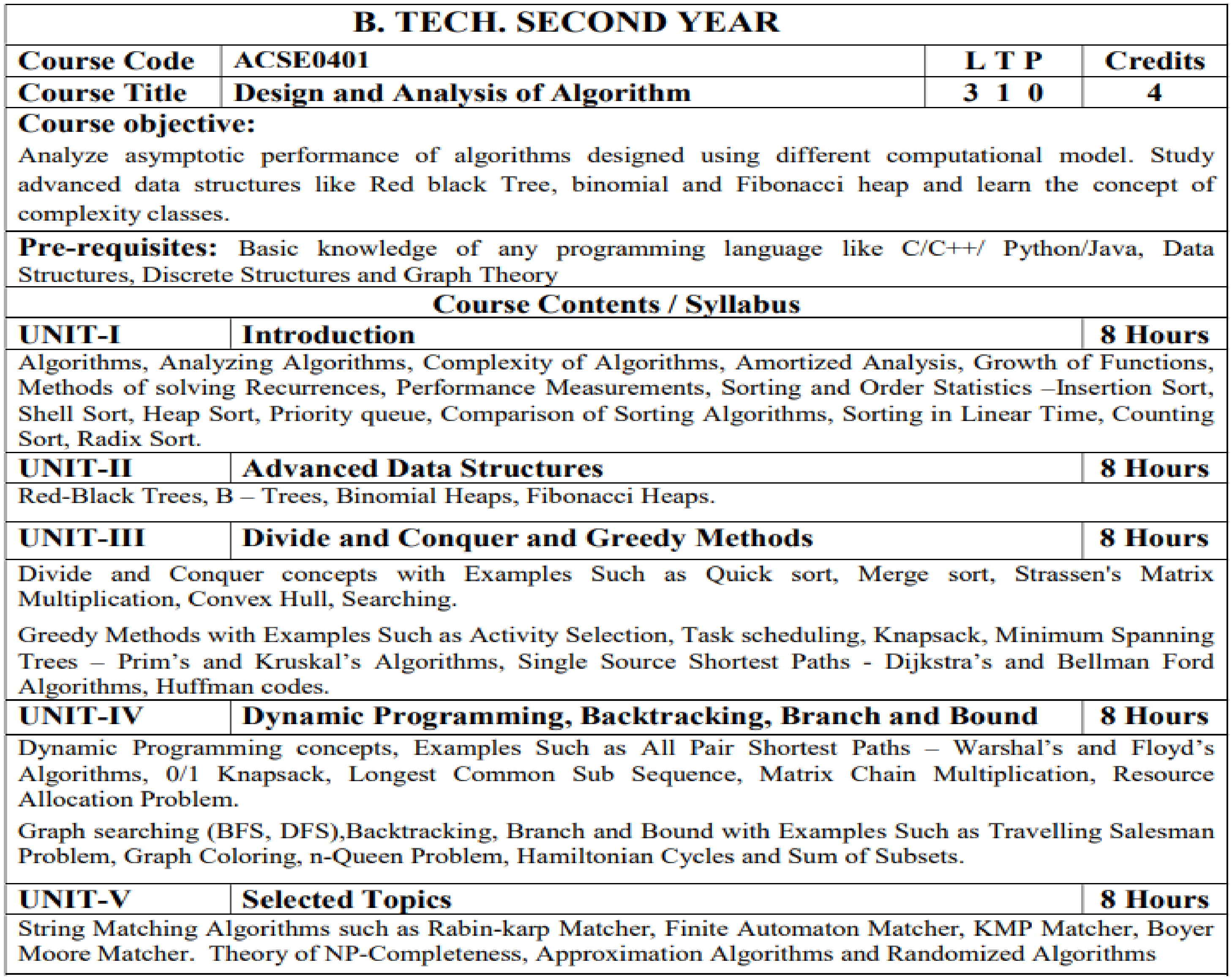


NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute)

B. TECH (CSE)
EVALUATION SCHEME
SEMESTER-IV

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0402	Engineering Mathematics-IV	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	ACSE0405	Microprocessor	3	0	0	30	20	50		100		150	3
4	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
5	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
6	ACSE0401	Design and Analysis of Algorithm	3	1	0	30	20	50		100		150	4
7	ACSE0455	Microprocessor Lab	0	0	2				25		25	50	1
8	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
9	ACSE0451	Design and Analysis of Algorithm Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1

Syllabus



B. TECH. SECOND YEAR			
Course Code	**ACSE0401**	**L T P**	**Credits**
Course Title	**Design and Analysis of Algorithm**	**3 1 0**	**4**
Course objective: Analyze asymptotic performance of algorithms designed using different computational model. Study advanced data structures like Red black Tree, binomial and Fibonacci heap and learn the concept of complexity classes.			
Pre-requisites: Basic knowledge of any programming language like C/C++/ Python/Java, Data Structures, Discrete Structures and Graph Theory			
Course Contents / Syllabus			
UNIT-I	**Introduction**		**8 Hours**
Algorithms, Analyzing Algorithms, Complexity of Algorithms, Amortized Analysis, Growth of Functions, Methods of solving Recurrences, Performance Measurements, Sorting and Order Statistics –Insertion Sort, Shell Sort, Heap Sort, Priority queue, Comparison of Sorting Algorithms, Sorting in Linear Time, Counting Sort, Radix Sort.			
UNIT-II	**Advanced Data Structures**		**8 Hours**
Red-Black Trees, B – Trees, Binomial Heaps, Fibonacci Heaps.			
UNIT-III	**Divide and Conquer and Greedy Methods**		**8 Hours**
Divide and Conquer concepts with Examples Such as Quick sort, Merge sort, Strassen's Matrix Multiplication, Convex Hull, Searching. Greedy Methods with Examples Such as Activity Selection, Task scheduling, Knapsack, Minimum Spanning Trees – Prim's and Kruskal's Algorithms, Single Source Shortest Paths - Dijkstra's and Bellman Ford Algorithms, Huffman codes.			
UNIT-IV	**Dynamic Programming, Backtracking, Branch and Bound**		**8 Hours**
Dynamic Programming concepts, Examples Such as All Pair Shortest Paths – Warshal's and Floyd's Algorithms, 0/1 Knapsack, Longest Common Sub Sequence, Matrix Chain Multiplication, Resource Allocation Problem. Graph searching (BFS, DFS),Backtracking, Branch and Bound with Examples Such as Travelling Salesman Problem, Graph Coloring, n-Queen Problem, Hamiltonian Cycles and Sum of Subsets.			
UNIT-V	**Selected Topics**		**8 Hours**
String Matching Algorithms such as Rabin-karp Matcher, Finite Automaton Matcher, KMP Matcher, Boyer Moore Matcher. Theory of NP-Completeness, Approximation Algorithms and Randomized Algorithms			

Branch wise Application

- In Data mining
- Image Processing
- Digital Signature.
- DNA Matching.

Course Objective

- Upon completion of this course, students will be able to do the following:
- Analyze the asymptotic performance of algorithms.
- Write rigorous correctness proofs for algorithms.
- Demonstrate a familiarity with major algorithms and data structures.
- Apply important algorithmic design paradigms and methods of analysis.
- Synthesize efficient algorithms in common engineering design situations.

Course Outcome

At the end of the semester, the student will be able:

	Description	Bloom's Taxonomy
CO 1	To have knowledge of basic principles of algorithm design and Analysis, asymptotic notations and growth of functions for time and space complexity analysis and applying the same in different sorting algorithms	Knowledge, analysis And design
CO 2	To apply different problem-solving approaches for advanced data structures	Knowledge, analysis And apply
CO 3	To apply divide and conquer method for solving merge sort, quick sort, matrix multiplication and Greedy Algorithm for solving different Graph Problem.	Knowledge, analysis and Apply
CO 4	To analyze and apply different optimization techniques like dynamic programming, backtracking and Branch & Bound to solve the complex problems	Knowledge, Analysis And Apply
CO 5	To understand the advanced concepts like NP Completeness and Fast Fourier Transform, to analyze and apply String Matching, Approximation and Randomized Algorithms to solve the complex problems	Knowledge, Analysis and Apply

Program Outcome

At the end of the semester, the student will be able:

POs	Engineering Graduates will be able to
PO1	Engineering Knowledge
PO2	Problem Analysis
PO3	Design & Development of solutions
PO4	Conduct Investigation of complex problems
PO5	Modern Tool Usage
PO6	The Engineer and Society
PO7	Environment and sustainability
PO8	Ethics
PO9	Individual & Team work
PO10	Communication
PO11	Project management and Finance
PO12	Life Long Learning

CO-PO and PSO Mapping

Design and Analysis of Algorithm (kCS-502)

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0401.1	3	3	3	3	2	-	-	-	2	2	-	3
ACSE0401.2	3	3	3	3	2	2	-	1	1	1	-	3
ACSE0401.3	3	3	2	3	3	2	-	2	1	1	2	3
ACSE0401.4	3	3	3	3	2	2	-	2	2	1	3	3
ACSE0401.5	2	2	2	2	2	2	-	2	1	1	1	2
Average	2.8	2.8	2.6	2.8	2.2	1.6	-	1.8	1.4	1.2	1.2	2.8

Program Educational Objectives(PEOs)

PEO1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.

PEO2: To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face global challenges.

PEO3: To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.

PEO4: To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, enterpreneur and bureaucrat for betterment of society

Result Analysis

Subject Result: 96.63 %

Department Result: 96.63 %

Faculty-Wise Result:

Mr. Harsh Vardhan Mishra (B): 93.94%
Mr. Harsh Vardhan Mishra(C): 97.01%
Mr. Twinkle Tyagi(A): 96.978%
Mr. Twinkle Tyagi(D): 98.53%

End Semester Question Paper Template

B TECH

(SEM-V) THEORY EXAMINATION 20__-20__

COMPILER DESIGN

Time: 3 Hours **Total Marks: 100**

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1

Q.No.	**Question**	**Marks**	**CO**
1		2	
2		2	
.		.	
10		2	

End Semester Question Paper Templates

SECTION B

2. Attempt any three of the following: 3 x 10 = 30

Q.No.	Question	Mark s	CO
1		10	
2		10	
.		.	
5		10	

3. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

NIET Greater Noida GET FUTURE READY Estd 2001 AN AUTONOMOUS INSTITUTE

End Semester Question Paper Templates

4. Attempt any one part of the following: 1 x 10 = 10

Q.No.	**Question**	**Marks**	**CO**
1		10	
2		10	

5. Attempt any one part of the following: 1 x 10 = 10

Q.No.	**Question**	**Marks**	**CO**
1		10	
2		10	

6. Attempt any one part of the following: 1 x 10 = 10

Q.No.	**Question**	**Marks**	**CO**
1		10	
2		10	

Prerequisite and Recap

- Prerequisite
- Basic concept of c programming language.
- Concept of stack, queue and link list.

- **Recap**
- Flow Chart
- Algorithm

Unit Content

- Algebraic Computation
- Fast Fourier Transform
- · String Matching
- Theory of NP-completeness
- Approximation algorithms
- Randomized algorithms.

Unit Objective

- This objective this unit is to make students understand about
- Different classes of problems , P Class, NP class, NPC.
- Different string matching algorithms.
- The Approximation algorithm and Randomized algorithm

Topic Objective

- This objective this topic is to make students understand about
- Computational Mathematics.
- Algebraic Structures
- Representation
- FFT & DFT

Algebraic Computation(CO5)

- In **computational** mathematics, computer **algebra**, also called symbolic **computation** or **algebraic computation**, is a scientific area that refers to the study and development of algorithms and software for manipulating mathematical expressions and other mathematical objects.

- Although , computer algebra should be a subfield of scientific computing is generally based on numerical computation with inexact floating point number, while representation computation emphasize exact computation with terminology contain variables that have no given value and are manipulated as symbols, hence the representation computation

Algebraic Structures

- Basic requirements
  - precise representation of algebraic structures
  - precise arithmetic with algebraic structures
  - other analytical operations with these structures (e.g., differentiation , integration)

Algebraic Computation(CO5)

- To work on a computer with algebraic structures, we need to represent them by some sort of data structures
- Representation is very important because often the effectiveness of an algorithm will depend on the representation that is used.
- Can be represented by
  – Representation of integers
  – Representation of polynomials
  – Representation of expressions

Algebraic Computation(CO5)

Fast Fourier transformation

- The discovery of the Fast Fourier transformation (FFT) is attributed to Cooley and Tukey, who published an algorithm in 1965.

- A fast Fourier transform (FFT) is an algorithm that computes the Discrete Fourier Transform (DFT) of a sequence, or its inverse (IDFT).

- Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

- The DFT is obtained by decomposing a sequence of values into components of different frequencies.

Algebraic Computation(CO5)

Fast Fourier transformation

- This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical.
- An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors.
- It manages to reduce the complexity of computing the DFT from $O(N^2)$, which arises if one simply applies the definition of DFT , to $O(N\log N)$, where N is the size of data.
- The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions.

Discrete Fourier transformation

- The **discrete fourier transform (DFT)** of the polynomial A(x) (or equivalently the vector of coefficients $(a_0,a_1,\ldots,a_{n-1})$ is defined as the values of the polynomial at the points $x=w_{n,k}$ i.e. it is the vector:

$$\begin{aligned}\mathrm{DFT}(a_0,a_1,\ldots,a_{n-1}) &= (y_0,y_1,\ldots,y_{n-1})\\ &= (A(w_{n,0}),A(w_{n,1}),\ldots,A(w_{n,n-1})\\ &= (A(w^0_n),A(w^1_n),\ldots,A(w^{n-1}_n))\end{aligned}$$

- Thus, if a direct DFT computes the values of the polynomial at the points at the n-th roots, the inverse DFT can restore the coefficients of the polynomial using those values.

Algebraic Computation(CO5)

- The **fast Fourier transform** is a method that allows computing the DFT in O(nlogn) time.

- The basic idea of the FFT is to apply divide and conquer.

- We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT for each of them, and combine the results to compute the DFT of the complete polynomial.

Algebraic Computation(CO5)

Inverse DFT

- The inverse DFT of values of the polynomial $(y_0,y_1,\dots,y_{n-1})$ are the coefficients of the polynomial $(a_0,a_1,\dots,a_{n-1})$.

  InverseDFT$(y_0,y_1,\dots,y_{n-1}) = (a_0,a_1,\dots,a_{n-1})$

- Thus, if a direct DFT computes the values of the polynomial at the points at the n-th roots, the inverse DFT can restore the coefficients of the polynomial using those values.

- The computation of the inverse DFT is almost the same as the calculation of the direct DFT, and it also can be performed in O(nlogn).

String Matching(CO5)
Objective

This objective this topic is to make students understand about

- Concepts of String Matching
- Naïve String Matching
- Rabin Karp Algorithm
- Knuth Morris Pratt
- String Matching with automata
- Boyer Moore Algorithm

Prerequisite and Recap

Prerequisite

- Algorithms
- Finite automata

Recap

- Algebraic Computation

String Matching (CO5)

- String Matching Algorithm is also called "String Searching Algorithm" are an important class of string algorithm that try to find a place where one or several strings(also called patterns) are found within a larger string or text.

- Given a text array, T [1.....n], of n character and a pattern array, P [1......m], of m characters. The problems are to find an integer s, called **valid shift** where $0 \leq s < n-m$ and T [s+1......s+m] = P [1......m].

- In other words, to find even if P in T, i.e., where P is a substring of T.

- The item of P and T are character drawn from some finite alphabet such as {0, 1} or {A, B .....Z, a, b..... z}.

String Matching (CO5)

- Given a string T [1......n], the **substrings** are represented as T [i......j] for some 0≤i ≤ j≤n-1, the string formed by the characters in T from index i to index j, inclusive. This process that a string is a substring of itself (take i = 0 and j =m).

- The **proper substring** of string T [1......n] is T [1......j] for some 0<i ≤ j≤n-1. That is, we must have either i>0 or j < m-1.

String Matching(CO5)

- Using these descriptions, we can say given any string T [1......n], the substrings are

T [i.....j] = T [i] T [i +1] T [i+2]......T [j] **for** some 0≤i ≤ j≤n-1.

And proper substrings are

T [i.....j] = T [i] T [i +1] T [i+2]......T [j] **for** some 0≤i ≤ j≤n-1.

- Note: If i>j, then T [i.....j] is equal to the empty string or null, which has length zero.

String Matching(CO5)

Algorithms used for String Matching:

There are different types of method is used to finding the string

- The Naive String Matching Algorithm
- The Rabin-Karp-Algorithm
- Finite Automata
- The Knuth-Morris-Pratt Algorithm
- The Boyer-Moore Algorithm

String Matching(CO5)

Naïve String Matching Algorithm

- The naïve approach tests all the possible placement of Pattern P [1.......m] relative to text T [1......n]. We try shift s = 0, 1.......n-m, successively and for each shift s. Compare T [s+1.......s+m] to P [1......m].

- The naïve algorithm finds all valid shifts using a loop that checks the condition P [1.......m] = T [s+1.......s+m] for each of the n - m +1 possible value of s.

String Matching(CO5)

NAIVE-STRING-MATCHER (T, P)

- 1. n ← length [T]
- 2. m ← length [P]
- 3. for s ← 0 to n -m
- 4. do if P [1.....m] = T [s + 1....s + m]
- 5. then print "Pattern occurs with shift" s

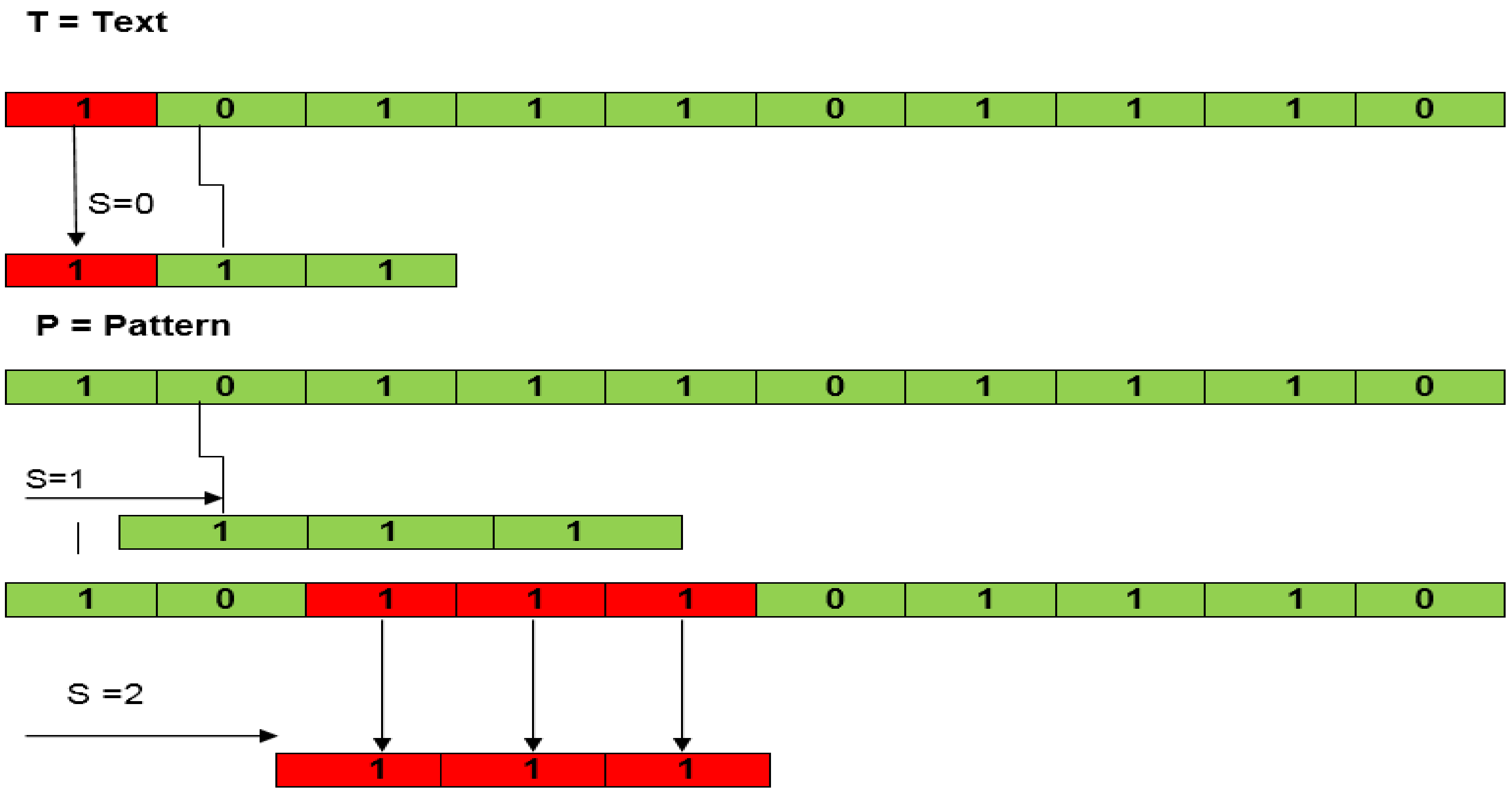
Analysis: This for loop from 3 to 5 executes for n-m + 1(we need at least m characters at the end) times and in iteration we are doing m comparisons. So the total complexity is O (n-m+1)

String Matching(CO5)

- Example:

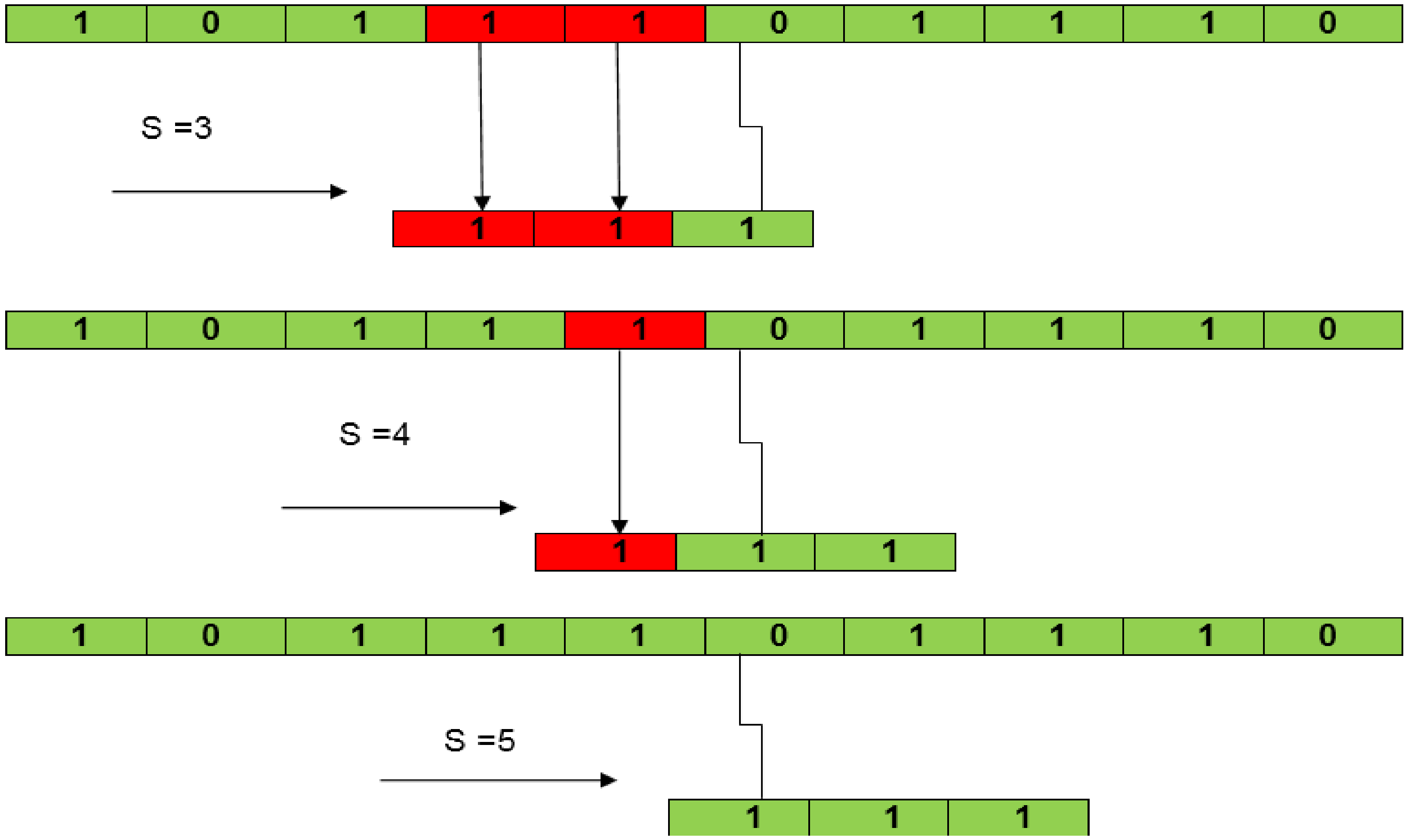
  Suppose T = 1011101110 , P = 111 . Find all the Valid Shift

Solution:

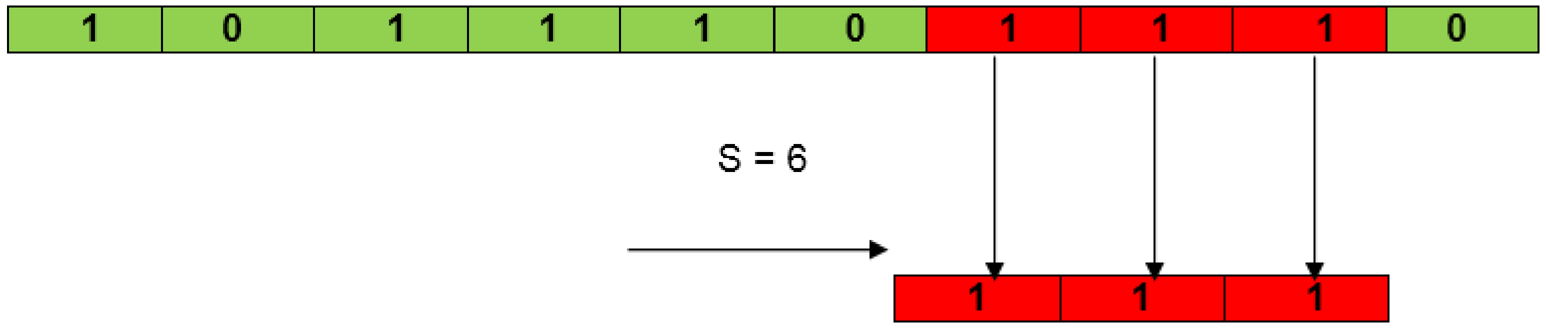
- 

String Matching(CO5)

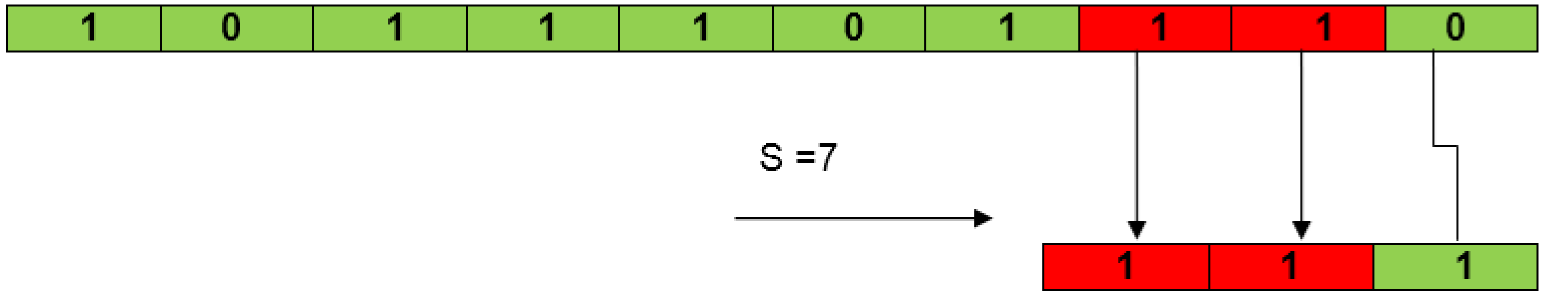
So, S=2 is a Valid Shift



String Matching Algorithm (CO5)



So, S=6 is a Valid Shift



String Matching(CO5)

Rabin- Karp- Algorithm

- The Rabin-Karp string matching algorithm calculates a hash value for the pattern, as well as for each M-character subsequences of text to be compared.

- If the hash values are unequal, the algorithm will determine the hash value for next M-character sequence.

- If the hash values are equal, the algorithm will analyze the pattern and the M-character sequence.

- In this way, there is only one comparison per text subsequence, and character matching is only required when the hash values match.

String Matching(CO5)

Rabin Karp Matcher(T, P, d, q)

```
1.  n ← length[T]
2.  m ← length[P]
3.  h ← $d^{m-1}$ mod q
4.  p ← 0
5.  $t_0$ ← 0
6.  For i ← 1 to m                          (preprocessing)
7.          do p ← (dp + P[i])modq
8.              $t_0$ ← (d $t_0$ + T[i])modq
9.  For s ← 0 to n-m                        (matching)
10.       do if p = $t_s$
11.               then if P[1…..m] = T[s +1,……..s+m]
12.                       then “Pattern occurs with shift” s
13. If s < n-m
14.     then $t_{s+1}$ ← (d($t_s$ – T[s+1]h) + T[s+m+1]) mod q
```

String Matching(CO5)

Rabin- Karp- Algorithm

Complexity: The running time of RABIN-KARP-MATCHER in the worst case scenario O ((n-m+1) m but it has a good average case running time. If the expected number of strong shifts is small O (1) and prime q is chosen to be quite large, then the Rabin-Karp algorithm can be expected to run in time O (n+m) plus the time to require to process spurious hits.

Example: For string matching, working module q = 11, how many spurious hits does the Rabin-Karp matcher encounters in Text T = 31415926535.......

T = 31415926535.......

P = 26

Here T.Length =11 so Q = 11

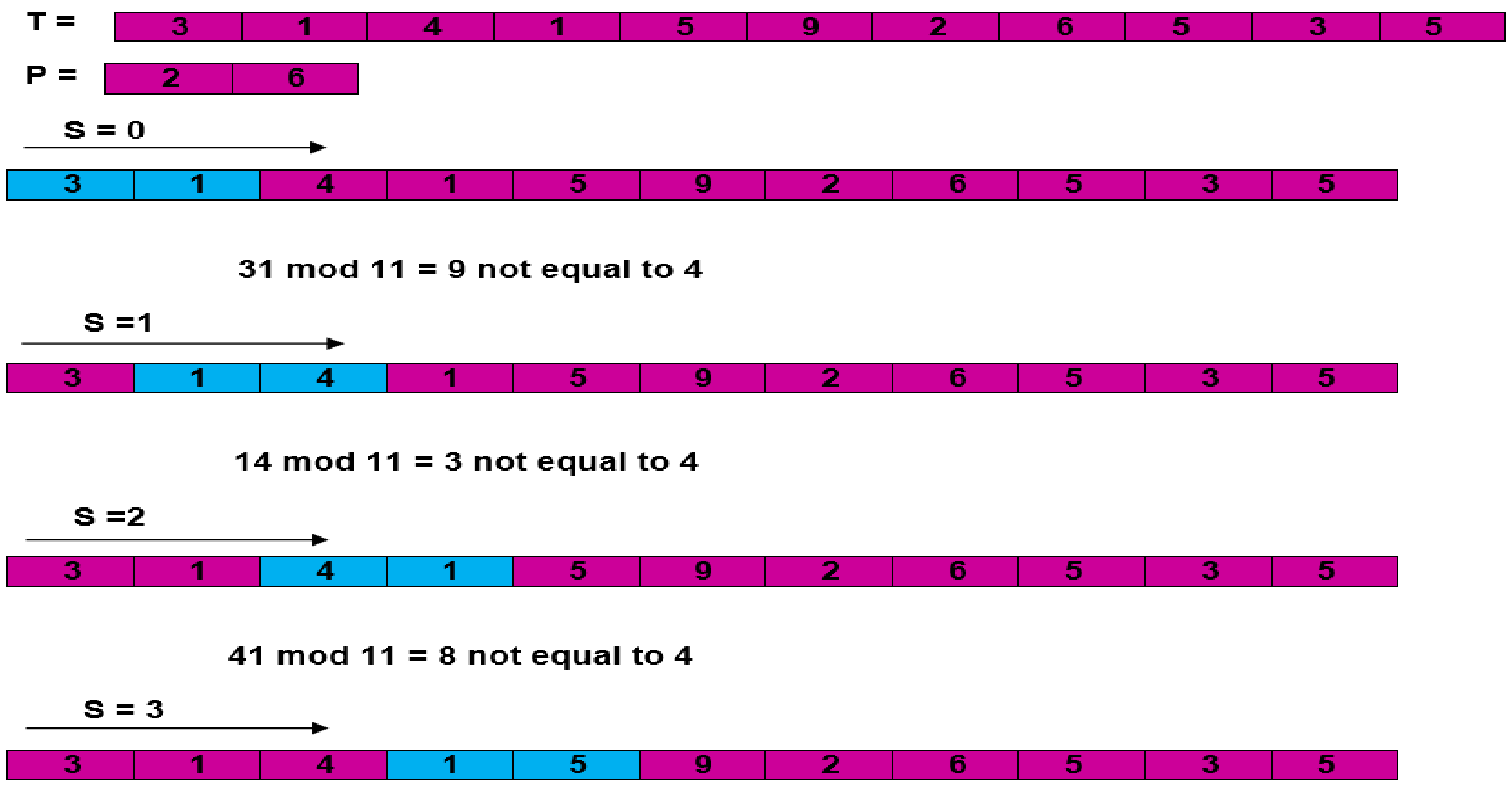
And P mod Q = 26 mod 11 = 4

Now find the exact match of P mod Q...

String Matching(CO5)

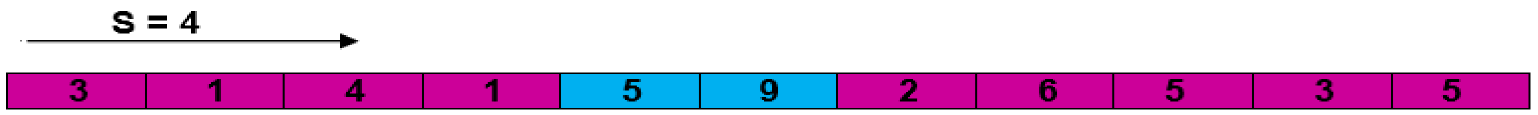
Rabin- Karp- Algorithm

Solution:

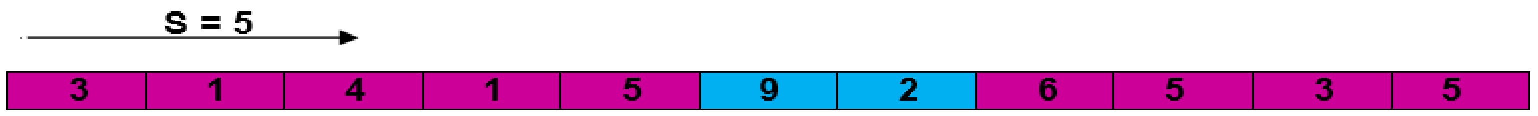


Rabin- Karp- Algorithm

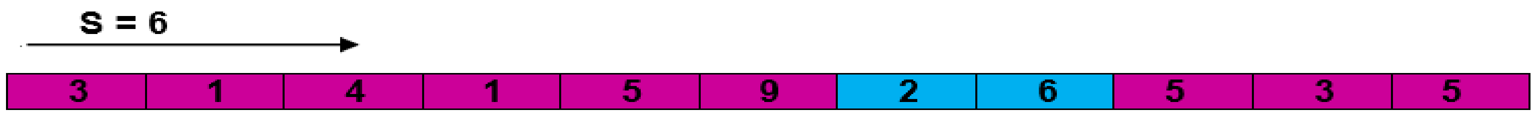
15 mod 11 = 4 equal to 4 SPURIOUS HIT



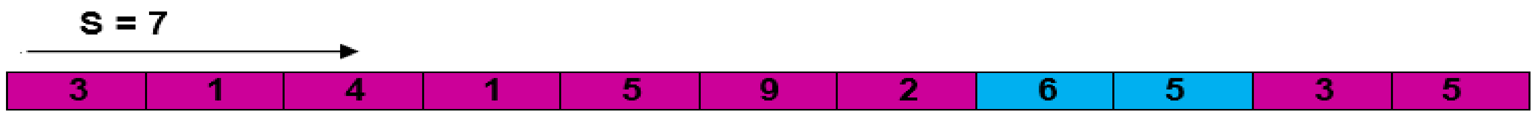
59 mod 11 = 4 equal to 4 SPURIOUS HIT



92 mod 11 = 4 equal to 4 SPURIOUS HIT



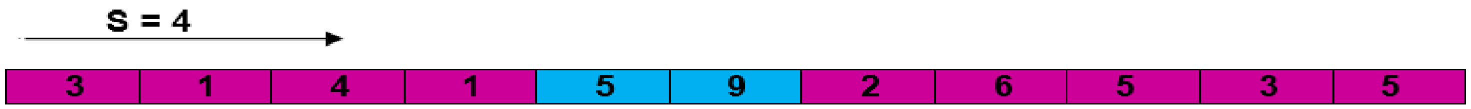
26 mod 11 = 4 EXACT MATCH



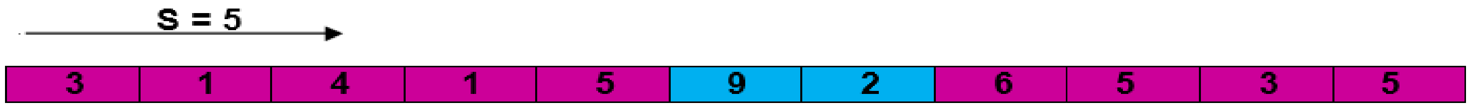
String Matching(CO5)

Rabin- Karp- Algorithm

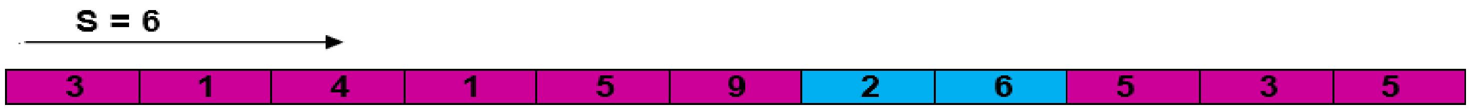
15 mod 11 = 4 equal to 4 SPURIOUS HIT



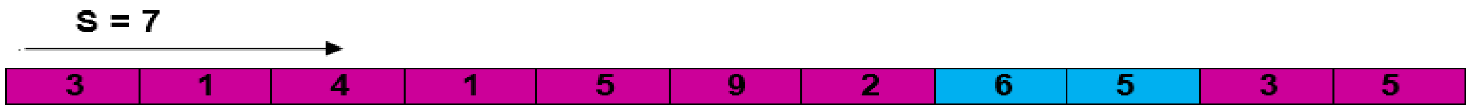
59 mod 11 = 4 equal to 4 SPURIOUS HIT



92 mod 11 = 4 equal to 4 SPURIOUS HIT

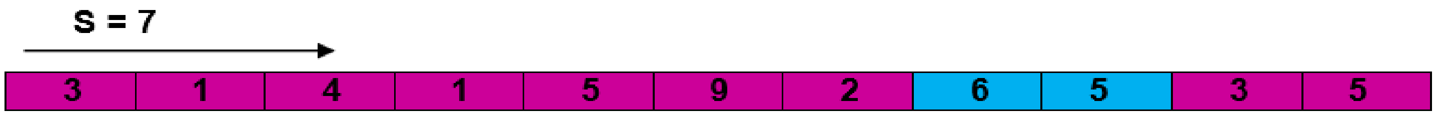


26 mod 11 = 4 EXACT MATCH

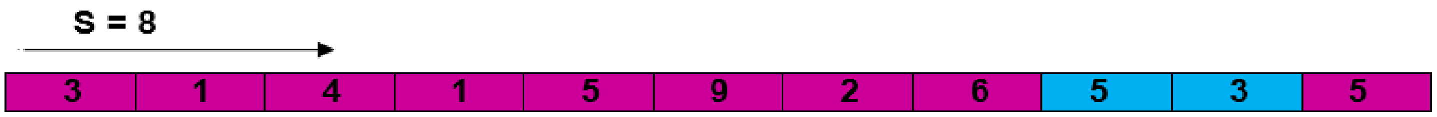


String Matching(CO5)

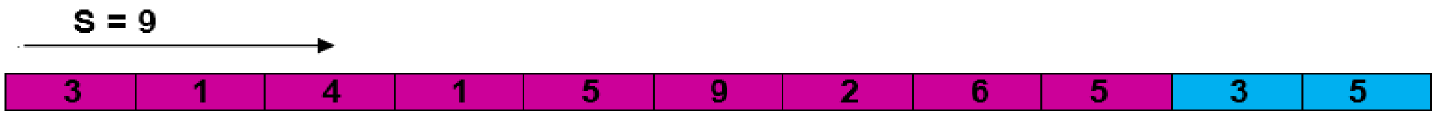
Rabin- Karp- Algorithm



65 mod 11 = 10 not equal to 4



53 mod 11 = 9 not equal to 4



35 mod 11 = 2 not equal to 4

The Pattern occurs with shift 6.

String Matching(CO5)

Rabin- Karp- Algorithm

- The string-matching automaton is a very useful tool which is used in string matching algorithm.
- It examines every character in the text exactly once and reports all the valid shifts in O (n) time.
- The goal of string matching is to find the location of specific text pattern within the larger body of text (a sentence, a paragraph, a book, etc.)

String Matching(CO5)

String Matching With Finite Automata

A finite automaton **M** is a 5-tuple **(Q, q_0,A,$\sum_\delta$)**, where

- Q is a finite set of **states**,
- $q_0 \in Q$ is the **start state**,
- $A \subseteq Q$ is a notable set of **accepting states**,
- $\sum$ is a **finite input alphabet**,
- δ is a function from **Q x $\sum$** into **Q** called the **transition function** of **M**.

- The finite automaton starts in state **q_0** and reads the characters of its input string one at a time.
- If the automaton is in state q and reads input character a, it moves from state q to state δ (q, a).
- Whenever its current state q is a member of A, the machine M has accepted the string read so far. An input that is not allowed is **rejected**.

String Matching(CO5)

Knuth-Morris and Pratt Algorithm

- Knuth-Morris and Pratt introduce a linear time algorithm for the string matching problem.

- A matching time of O (n) is achieved by avoiding comparison with an element of 'S' that have previously been involved in comparison with some element of the pattern 'p' to be matched. i.e., backtracking on the string 'S' never occurs.

String Matching(CO5)

Components of KMP Algorithm:

1. **The Prefix Function (Π):** The Prefix Function, Π for a pattern encapsulates knowledge about how the pattern matches against the shift of itself. This information can be used to avoid a useless shift of the pattern 'p.' In other words, this enables avoiding backtracking of the string 'S.'

2. **The KMP Matcher:** With string 'S,' pattern 'p' and prefix function 'Π' as inputs, find the occurrence of 'p' in 'S' and returns the number of shifts of 'p' after which occurrences are found.

String Matching(CO5)

The Prefix Function (Π)

Following pseudo code compute the prefix function, Π:

```
Compute-Prefix-function(P)
1.  n← length[P]
2.  Π[1]← 0
3.  K← 0
4.  for q← 2 to m
5.        do while k > 0 and P[k+1] ≠ P[q]
6.             do k← Π[k]
7.        if P[k+1] = P[q]
8.              then k← k+1
9.        Π[q]← k
10. Return Π
```

In the above pseudo code for calculating the prefix function, the for loop from step 4 to step 10 runs 'm' times. Step1 to Step3 take constant time. Hence the running time of computing prefix function is O (m).

String Matching(CO5)

Knuth-Morris and Pratt Algorithm

Example: Compute Π for the pattern 'p' below:

P :						
a	b	a	b	a	c	a

Solution:

Initially: m = length [p] = 7 Π [1] = 0 k = 0

String Matching(CO5)

Knuth- Morris and Pratt(KMP) algorithm

Step 1: q = 2, k = 0

Π [2] = 0

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0					

Step 2: q = 3, k = 0

Π [3] = 1

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1				

Step3: q =4, k =1

Π [4] = 2

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	A
π	0	0	1	2			

String Matching(CO5)

Knuth-Morris and Pratt Algorithm

Step4: q = 5, k =2

Π [5] = 3

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1	2	3		

Step5: q = 6, k = 3

Π [6] = 0

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1	2	3	0	

Step6: q = 7, k = 1

Π [7] = 1

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1	2	3	0	1

String Matching(CO5)

Boyer and Moore Algorithm

- Robert Boyer and J Strother Moore established it in 1977. The B-M String search algorithm is a particularly efficient algorithm and has served as a standard benchmark for string search algorithm ever since.

- The B-M algorithm takes a 'backward' approach: the pattern string (P) is aligned with the start of the text string (T), and then compares the characters of a pattern from right to left, beginning with rightmost character.

- If a character is compared that is not within the pattern, no match can be found by analyzing any further aspects at this position so the pattern can be changed entirely past the mismatching character.

String Matching(CO5)

Boyer and Moore Algorithm

- For deciding the possible shifts, B-M algorithm uses two preprocessing strategies simultaneously. Whenever a mismatch occurs, the algorithm calculates a variation using both approaches and selects the more significant shift thus, if make use of the most effective strategy for each case.

- The two strategies are called heuristics of B - M as they are used to reduce the search. They are
- Bad Character Heuristics
- Good Suffix Heuristics

String Matching(CO5)

Boyer and Moore Algorithm

Bad Character Heuristics

This Heuristics has two implications:

- Suppose there is a character in a text in which does not occur in a pattern at all. When a mismatch happens at this character (called as bad character), the whole pattern can be changed, begin matching form substring next to this 'bad character.’
- On the other hand, it might be that a bad character is present in the pattern, in this case, align the nature of the pattern with a bad character in the text.
- Thus in any case shift may be higher than one.

Boyer and Moore Algorithm

Good Suffix Heuristics:

- A good suffix is a suffix that has matched successfully.
- After a mismatch which has a negative shift in bad character heuristics, look if a substring of pattern matched till bad character has a good suffix in it, if it is so then we have an onward jump equal to the length of suffix found.

NP Completeness(CO5)
Objective

This objective this topic is to make students understand about

- Different Class of problems
- P Class
- NP Class
- NPC
- Hard Problems
- Approximation and Randomized algorithm

Prerequisite and Recap

Prerequisite

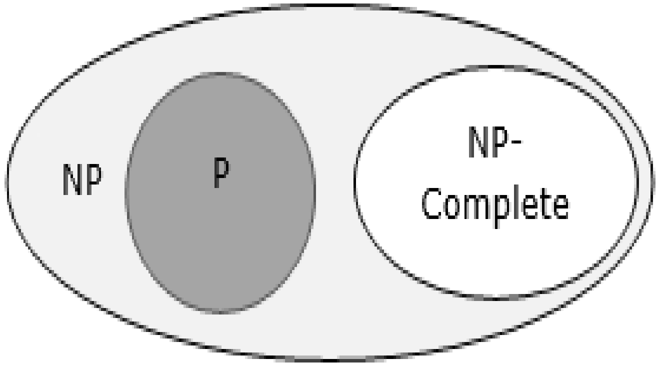
- Different Problems like graph colouring
- Travelling Salesman Problem

Recap

- String Matching Algorithm
- Algorithms for solving real world problems

NP Completeness(CO5)

- A problem is in the class NPC if it is in NP and is as **hard** as any problem in NP. A problem is **NP-hard** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself.



- If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable. These problems are called **NP-complete**. The phenomenon of NP-completeness is important for both theoretical and practical reasons.

NP Completeness(CO5)

A language **B** is ***NP-complete*** if it satisfies two conditions

1. **B** is in NP
2. Every **A** in NP is polynomial time reducible to **B**.

- If a language satisfies the second property, but not necessarily the first one, the language **B** is known as **NP-Hard**. Informally, a search problem **B** is **NP-Hard** if there exists some **NP-Complete** problem **A** that Turing reduces to **B**.

- The problem in NP-Hard cannot be solved in polynomial time, until **P = NP**. If a problem is proved to be NPC, there is no need to waste time on trying to find an efficient algorithm for it. Instead, we can focus on design approximation algorithm.

NP Completeness(CO5)

- **NP-Complete Problems**

Following are some NP-Complete problems, for which no polynomial time algorithm is known.

- Determining whether a graph has a Hamiltonian cycle
- Determining whether a Boolean formula is satisfiable, etc.

- **NP-Hard Problems**

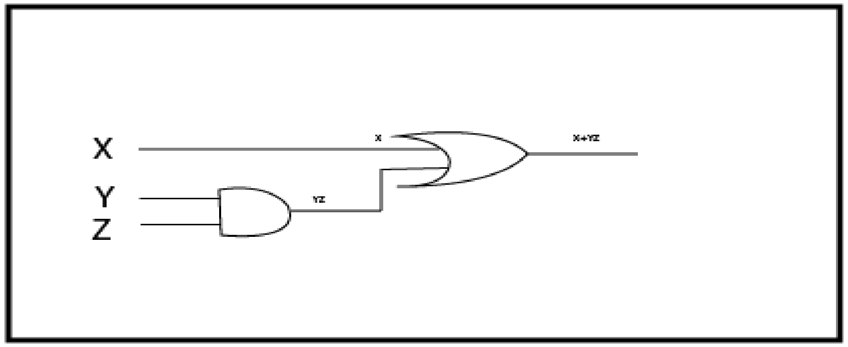
The following problems are NP-Hard

- The circuit-satisfiability problem
- Set Cover
- Vertex Cover
- Travelling Salesman Problem

NP Completeness(CO5)

Circuit Satisfiability

According to given decision-based NP problem, we can design the CIRCUIT and verify a given mentioned output also within the P time. The CIRCUIT is provided below:-



Although we can design a circuit and verified the mentioned output within Polynomial time but remember we can never predict the number of gates which produces the high output against the set of inputs/high inputs within a polynomial time. So we verified the production and conversion had been done within polynomial time. So it is NPC.

NP Completeness(CO5)

Set Cover Problem

- Given a universe U of n elements, a collection of subsets of U say S = {S_1, S_2…,S_m} where every subset S_i has an associated cost.
- Find a minimum cost subcollection of S that covers all elements of U.
- There is no polynomial time solution available for this problem as the problem is a known NP-Hard problem

NP Completeness(CO5)

Example:

U = {1,2,3,4,5},   S = {S_1,S_2,S_3}

S_1 = {4,1,3},   Cost(S_1) = 5
S_2 = {2,5},   Cost(S_2) = 10
S_3 = {1,4,3,2}, Cost(S_3) = 3

Output: Minimum cost of set cover is 13 and set cover is {S2, S3}
There are two possible set covers {S_1, S_2} with cost 15 and {S_2, S_3} with cost 13

NP Completeness(CO5)

Vertex Cover problem

The minimum **vertex cover problem** is the optimization **problem** of finding a smallest **vertex cover** in a given graph. The **vertex cover problem** is an NP-complete **problem.**
A vertex-cover of an undirected graph $\boldsymbol{G = (V, E)}$ is a subset of vertices $\boldsymbol{V' \subseteq V}$ such that if edge $\boldsymbol{(u, v)}$ is an edge of $\boldsymbol{G}$, then either $\boldsymbol{u}$ in $\boldsymbol{V}$ or $\boldsymbol{v}$ in $\boldsymbol{V'}$ or both.

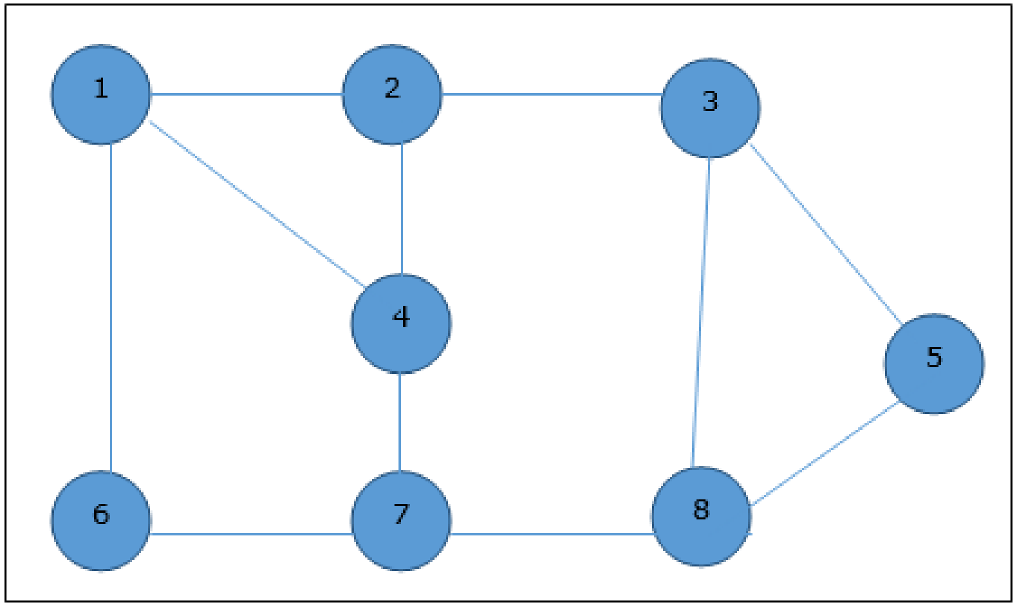
```
APPROX-VERTEX_COVER (G: Graph) c ← { } E' ← E[G]
1. while E' is not empty do
2.  Let (u, v) be an arbitrary edge of E' c ← c U {u, v}
3.  Remove from E' every edge incident on either u or v
4. return c
```

NP Completeness(CO5)

Example:

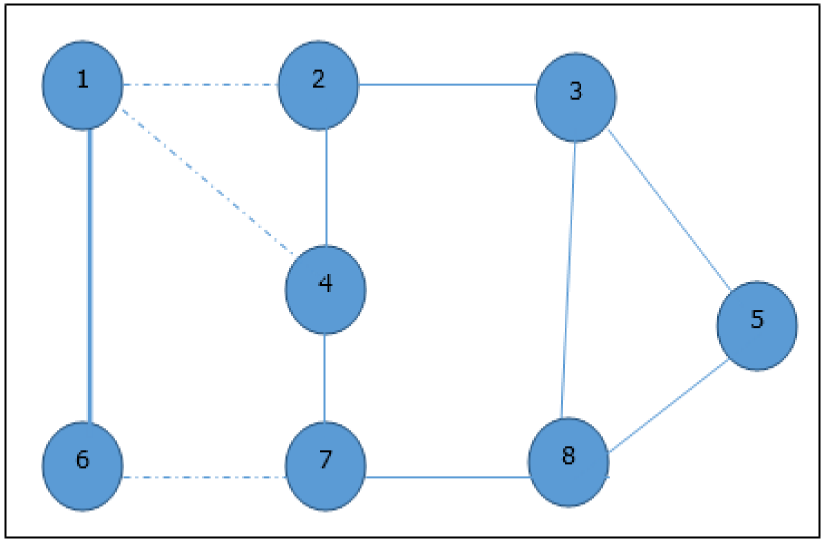
The set of edges of the given graph is –

{(1,6),(1,2),(1,4),(2,3),(2,4),(6,7),(4,7),(7,8),(3,8),(3,5),(8,5)}

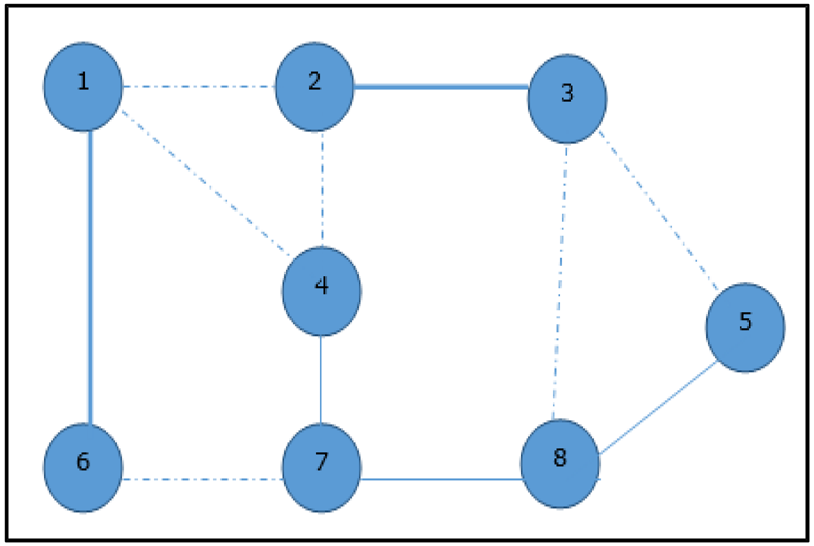


NP Completeness(CO5)

Now, we start by selecting an arbitrary edge (1,6). We eliminate all the edges, which are either incident to vertex 1 or 6 and we add edge (1,6) to cover.

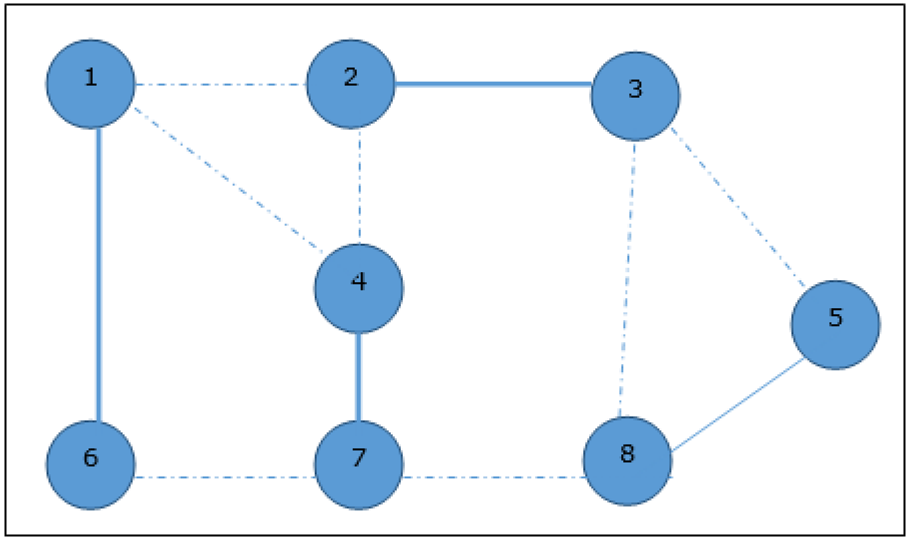


In the next step, we have chosen another edge (2,3) at random

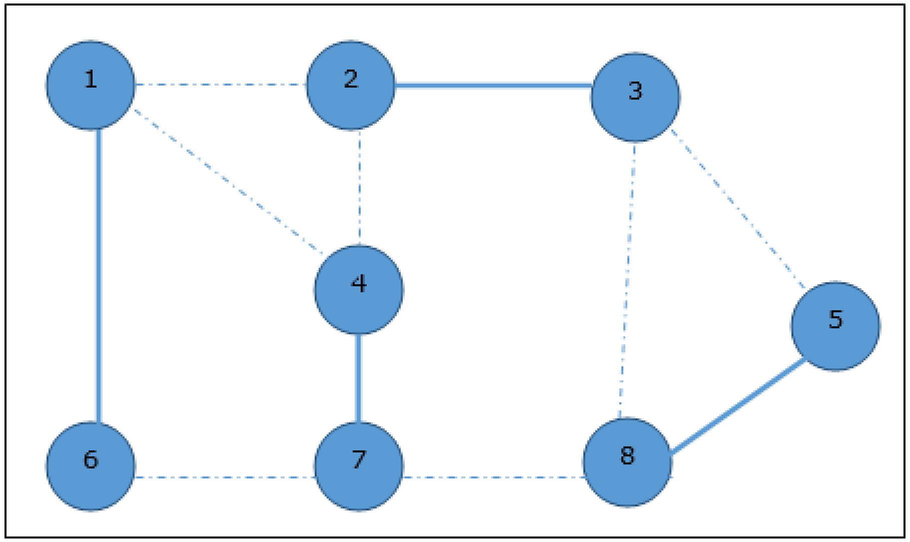


NP Completeness(CO5)

Now we select another edge (4,7).



We select another edge (8,5).



Hence, the vertex cover of this graph is {1,2,4,5}.

Analysis

It is easy to see that the running time of this algorithm is $\boldsymbol{O(V + E)}$, using adjacency list to represent $\boldsymbol{E}'$.

NP Completeness(CO5)

Travelling Salesman Problem

- The traveling salesman problem consists of a salesman and a set of cities.
- The salesman has to visit each one of the cities starting from a certain one and returning to the same city.
- The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip

NP Completeness(CO5)

Proof

1. To prove ***TSP is NP-Complete***, first we have to prove that ***TSP belongs to NP***.

- In TSP, we find a tour and check that the tour contains each vertex once.
- Then the total cost of the edges of the tour is calculated.
- Finally, we check if the cost is minimum.
- This can be completed in polynomial time.
- Thus ***TSP belongs to NP***.

NP Completeness(CO5)

2. Secondly, we have to prove that ***TSP is NP-hard***.

- To prove this, one way is to show that ***Hamiltonian cycle*** $\le_p$ ***TSP*** (as we know that the Hamiltonian cycle problem is NPcomplete).

Assume ***G = (V, E)*** to be an instance of Hamiltonian cycle.

Hence, an instance of TSP is constructed. We create the complete graph ***G′ = (V, E′)***, where

$$E' = \{(i,j) : i,j \in V \text{ and } i \neq j$$

$$E' = \{(i,j) : i,j \in V \text{ and } i \neq j$$

Thus, the cost function is defined as follows −

$$t(i,j) = 0 \text{ if } (i,j) \in E$$
$$= 1 \quad \text{otherwise}$$

NIET Greater Noida GET FUTURE READY

NP Completeness(CO5)

- Now, suppose that a Hamiltonian cycle $\boldsymbol{h}$ exists in $\boldsymbol{G}$. It is clear that the cost of each edge in $\boldsymbol{h}$ is $\mathbf{0}$ in $\boldsymbol{G}'$ as each edge belongs to $\boldsymbol{E}$. Therefore, $\boldsymbol{h}$ has a cost of $\mathbf{0}$ in $\boldsymbol{G}'$. Thus, if graph $\boldsymbol{G}$ has a Hamiltonian cycle, then graph $\boldsymbol{G}'$ has a tour of $\mathbf{0}$ cost.

- Conversely, we assume that $\boldsymbol{G}'$ has a tour $\boldsymbol{h}'$ of cost at most $\mathbf{0}$. The cost of edges in $\boldsymbol{E}'$ are $\mathbf{0}$ and $\mathbf{1}$ by definition. Hence, each edge must have a cost of $\mathbf{0}$ as the cost of $\boldsymbol{h}'$ is $\mathbf{0}$. We therefore conclude that $\boldsymbol{h}'$ contains only edges in $\boldsymbol{E}$.

- We have thus proven that $\boldsymbol{G}$ has a Hamiltonian cycle, if and only if $\boldsymbol{G}'$ has a tour of cost at most $\mathbf{0}$. TSP is NP-complete.

NP Completeness(CO5)

- An **Approximate Algorithm** is a way of approach NP-COMPLETENESS for the optimization problem.
- This technique does not guarantee the best solution.
- The goal of an **approximation algorithm** is to come as close as possible to the optimum value in a reasonable amount of time which is at the most polynomial time
- A **simple example** of an **approximation algorithm** is one for the Minimum Vertex Cover problem, where the goal is to choose the smallest set of vertices such that every edge in the input graph contains at least one chosen vertex.

NP Completeness(CO5)

- An **algorithm** that uses random numbers to decide what to do next anywhere in its logic is called **Randomized Algorithm**.. For example, in **Randomized** Quick Sort, we use random number to pick the next pivot (or we randomly shuffle the array). And in Karger's **algorithm**, we randomly pick an edge..

For example:

- Select a random number from stream, with O(1) space.
- Birthday Paradox
- Linearity of Expectation
- Random Number generator in arbitrary probability distribution fashion

Faculty Video Links, You tube & NPTEL Video Links and Online Courses Details

- Self Made Video Link:

- Youtube/other Video Links
- https://www.youtube.com/watch?v=e2cF8a5aAhE
- https://www.youtube.com/watch?v=jFW7fwa0Zm8
- https://www.youtube.com/watch?v=MEz1J9wY2iM

Daily Quiz

- What does NP stands for in complexity classes theory?
- The hardest of NP problems are _____________.
- Travelling sales man problem belongs to which of the class?
- A problem which is both ________ and __________ is said to be NP complete.
- In terms of NTIME, NP problems are the set of decision problems which can be solved using a non deterministic machine in ________ time.
- Problems that can be solved in polynomial time are known as?
- The sum and composition of two polynomials are always polynomials.(True/False).
- .... is the class of decision problems that can be solved by non-deterministic polynomial algorithms?

Weekly Assignment

1.Write Rabin-Karp algorithm. For string matching working modulo q = 11, how many spurious hits does the Rabin-Karp matcher encounter in the text T = 3141592653589793 when looking for pattern P = 26? [CO5]

2.Define NP, NP hard and NP complete. Give example of each. [CO5]

3. Explain Approximation and Randomized algorithms. [CO5]

4. Explain Rabin Karp algorithm. For the text 2359023141526739921 and for the pattern 31415 and working modulo q=13 how many valid match and spurious hits does the Rabin matcher encounter. [CO5]

5. Write short notes on [CO5]

- Algebraic Computation
- FFT

MCQs

1. Which of the following can be used to define NP complexity class?
   a) Verifier
   b) Polynomial time
   c) Both (a) and (b)
   d) None of the mentioned

2. Which of the following are not in NP?
   a) All problems in P
   b) Boolean Satisfiability problems
   c) Integer factorization problem
   d) None of the mentioned

MCQs

3. Which of the following contains NP?
   a) PSPACE
   b) EXPSPACE
   c) Both (a) and (b)
   d) None of the mentioned

4. State true or false?
   Statement: If a problem X is in NP and a polynomial time algorithm for X could also be used to solve problem Y in polynomial time, then Y is also in NP.
   a) true
   b) false

MCQs

5. Which of the following is incorrect for the given phrase
Phrase :'solvable by non deterministic algorithms in polynomial time'
a) NP Problems
b) During control flow, non deterministic algorithm may have more than one choice
c) If the choices that non deterministic algorithm makes are correct, the amount of time it takes is bounded by polynomial time.
d) None of the mentioned

6. Which of the following can be used to define NP complexity class?
a) Verifier
b) Polynomial time
c) Both (a) and (b)
d) None of the mentioned

MCQs

7. Which of the following does not belong to the closure properties of NP class?
   a) Union
   b) Concatenation
   c) Reversal
   d) Complement

8. The worst-case efficiency of solving a problem in polynomial time is?
   a) O(p(n))
   b) O(p(n log n))
   c) O(p(n^2))
   d) O(p(m log n))

MCQs

9. Halting problem is an example for?
   a) decidable problem
   b) undecidable problem
   c) complete problem
   d) trackable problem

10. A non-deterministic algorithm is said to be non-deterministic polynomial if the time-efficiency of its verification stage is polynomial.
    a) true
    b) false

Glossary Question

Q.1 _________worst-case efficiency of solving a problem in polynomial time.

a) O(p(n))

b) O(p(n log n))

c) $O(p(n^2))$

d) O(p(m log n))

Q.2 Problems that can be solved in polynomial time are known as_____________.

a) intractable

b) tractable

c) decision

d) complete

Q.3 __________ is the class of decision problems that can be solved by non-deterministic polynomial algorithms.

a) NP

b) P

c) Hard

d) Complete

Glossary Question

Q.4 Problems that cannot be solved by any algorithm are called _________.
a) tractable problems
b) intractable problems
c) undecidable problems
d) decidable problems

Q.5 Halting problem is an example for___________.
a) decidable problem
b) undecidable problem
c) complete problem
d) trackable problem.

Q.6 __________conditions have to be met if an NP- complete problem is polynomially reducible?
a) 1
b) 2
c) 3
d) 4

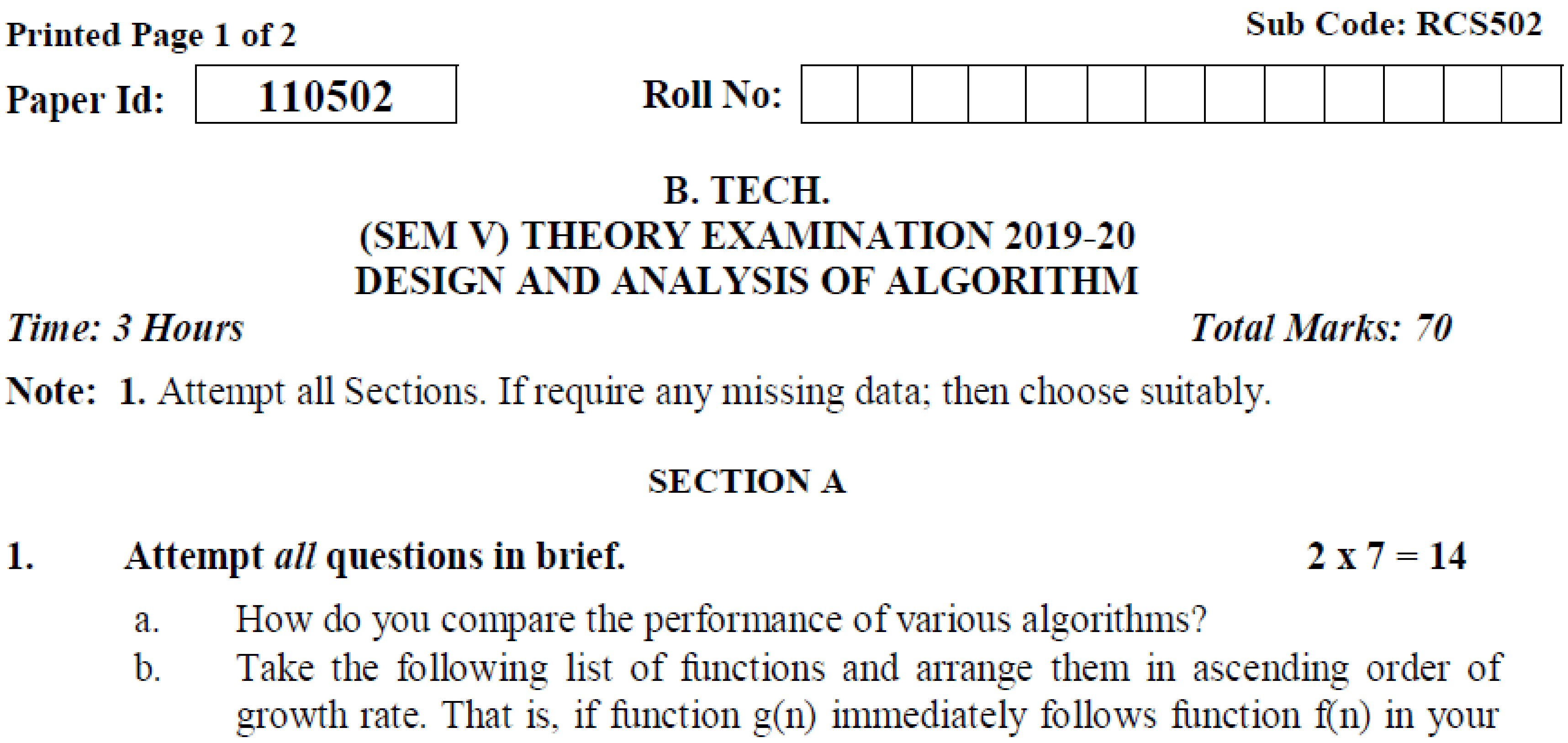
Glossary Question

Q.7 A CNF-satisfiability problem belong to______
a) NP class
b) P class
c) NP complete
d) NP hard

Q.8 The choice of polynomial class has led to the development of an extensive theory called ________
a) computational complexity
b) time complexity
c) problem complexity
d) decision complexity

Q.9 A randomized algorithm uses random bits as input in order to achieve a _____________ good performance over all possible choice of random bits.
a) worst case
b) best case
c) average case
d) none of the mentioned

Old Question Papers(2019-2020)



Printed Page 1 of 2 Sub Code: RCS502

Paper Id: 110502 Roll No:

B. TECH.
(SEM V) THEORY EXAMINATION 2019-20
DESIGN AND ANALYSIS OF ALGORITHM

Time: 3 Hours ***Total Marks: 70***

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

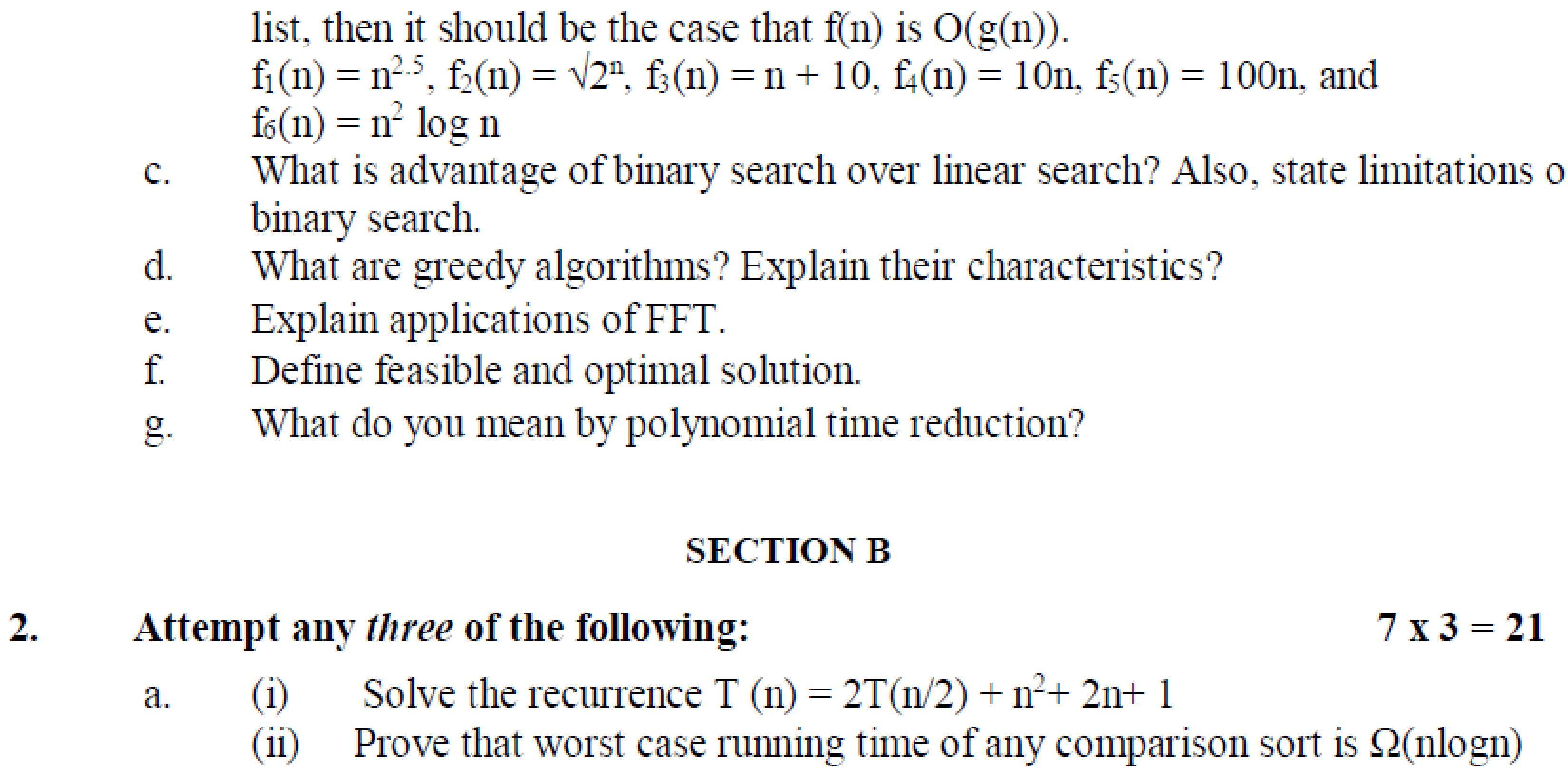
SECTION A

1. Attempt *all* questions in brief. **2 x 7 = 14**

a. How do you compare the performance of various algorithms?

b. Take the following list of functions and arrange them in ascending order of growth rate. That is, if function g(n) immediately follows function f(n) in your

Old Question Papers(2019-2020)



list, then it should be the case that f(n) is O(g(n)).
$f_1(n) = n^{2.5}$, $f_2(n) = \sqrt{2}^n$, $f_3(n) = n + 10$, $f_4(n) = 10n$, $f_5(n) = 100n$, and
$f_6(n) = n^2 \log n$

c. What is advantage of binary search over linear search? Also, state limitations of binary search.

d. What are greedy algorithms? Explain their characteristics?

e. Explain applications of FFT.

f. Define feasible and optimal solution.

g. What do you mean by polynomial time reduction?

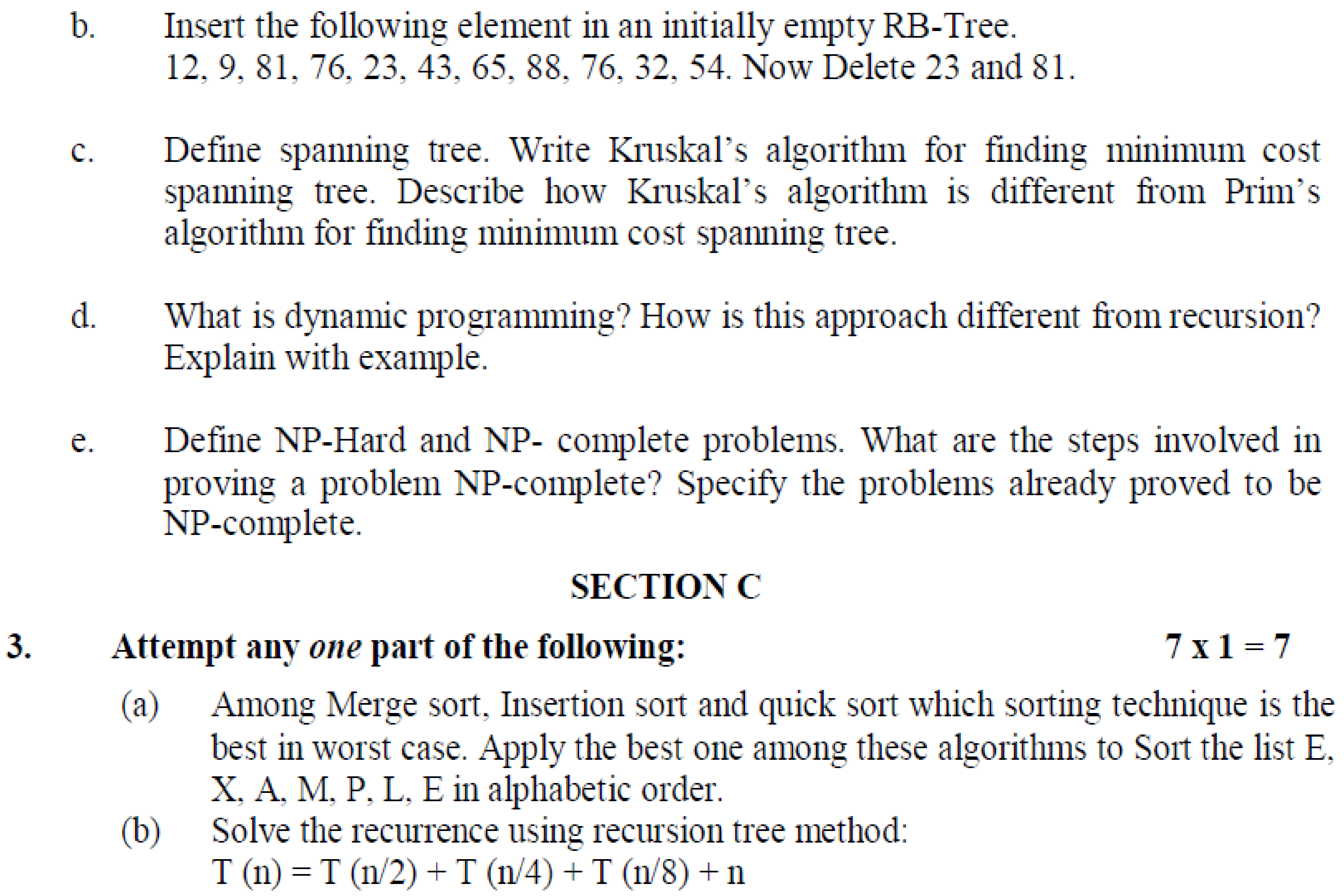
SECTION B

2. Attempt any *three* of the following: 7 x 3 = 21

a. (i) Solve the recurrence $T(n) = 2T(n/2) + n^2 + 2n + 1$

(ii) Prove that worst case running time of any comparison sort is $\Omega(n\log n)$

Old Question Papers(2019-2020)



b. Insert the following element in an initially empty RB-Tree.
12, 9, 81, 76, 23, 43, 65, 88, 76, 32, 54. Now Delete 23 and 81.

c. Define spanning tree. Write Kruskal's algorithm for finding minimum cost spanning tree. Describe how Kruskal's algorithm is different from Prim's algorithm for finding minimum cost spanning tree.

d. What is dynamic programming? How is this approach different from recursion? Explain with example.

e. Define NP-Hard and NP- complete problems. What are the steps involved in proving a problem NP-complete? Specify the problems already proved to be NP-complete.

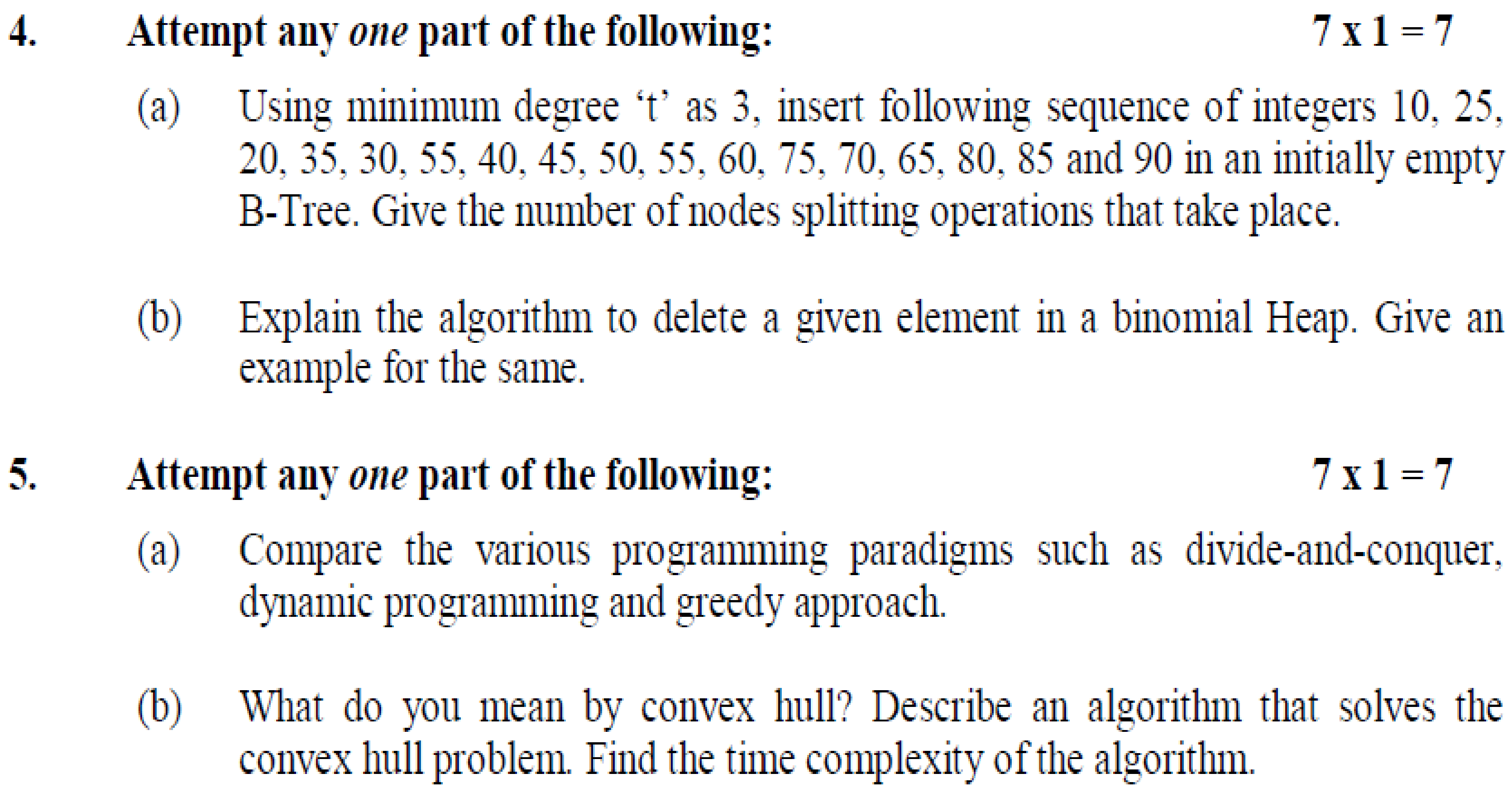
SECTION C

3. Attempt any *one* part of the following: 7 x 1 = 7

(a) Among Merge sort, Insertion sort and quick sort which sorting technique is the best in worst case. Apply the best one among these algorithms to Sort the list E, X, A, M, P, L, E in alphabetic order.

(b) Solve the recurrence using recursion tree method:
$T(n) = T(n/2) + T(n/4) + T(n/8) + n$

Old Question Papers(2019-2020)



4. **Attempt any *one* part of the following:** **7 x 1 = 7**

(a) Using minimum degree 't' as 3, insert following sequence of integers 10, 25, 20, 35, 30, 55, 40, 45, 50, 55, 60, 75, 70, 65, 80, 85 and 90 in an initially empty B-Tree. Give the number of nodes splitting operations that take place.

(b) Explain the algorithm to delete a given element in a binomial Heap. Give an example for the same.

5. **Attempt any *one* part of the following:** **7 x 1 = 7**

(a) Compare the various programming paradigms such as divide-and-conquer, dynamic programming and greedy approach.

(b) What do you mean by convex hull? Describe an algorithm that solves the convex hull problem. Find the time complexity of the algorithm.

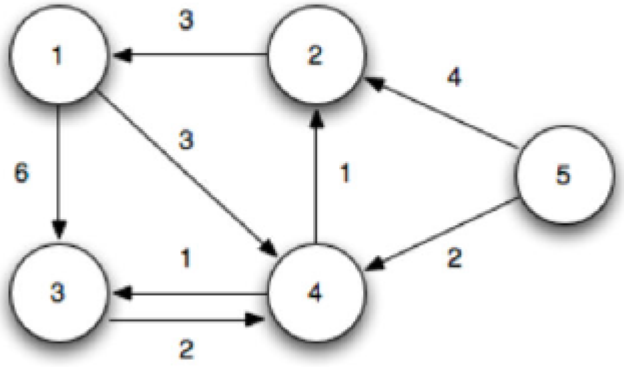
NIET Greater Noida GET FUTURE READY

Old Question Papers(2019-2020)

6. Attempt any *one* part of the following: 7 x 1 = 7

(a) Solve the following 0/1 knapsack problem using dynamic programming. P=[11,21,31,33] w=[2,11,22,15] c=40, n=4.

(b) Define Floyd Warshall Algorithm for all pair shortest path and apply the same on following graph:



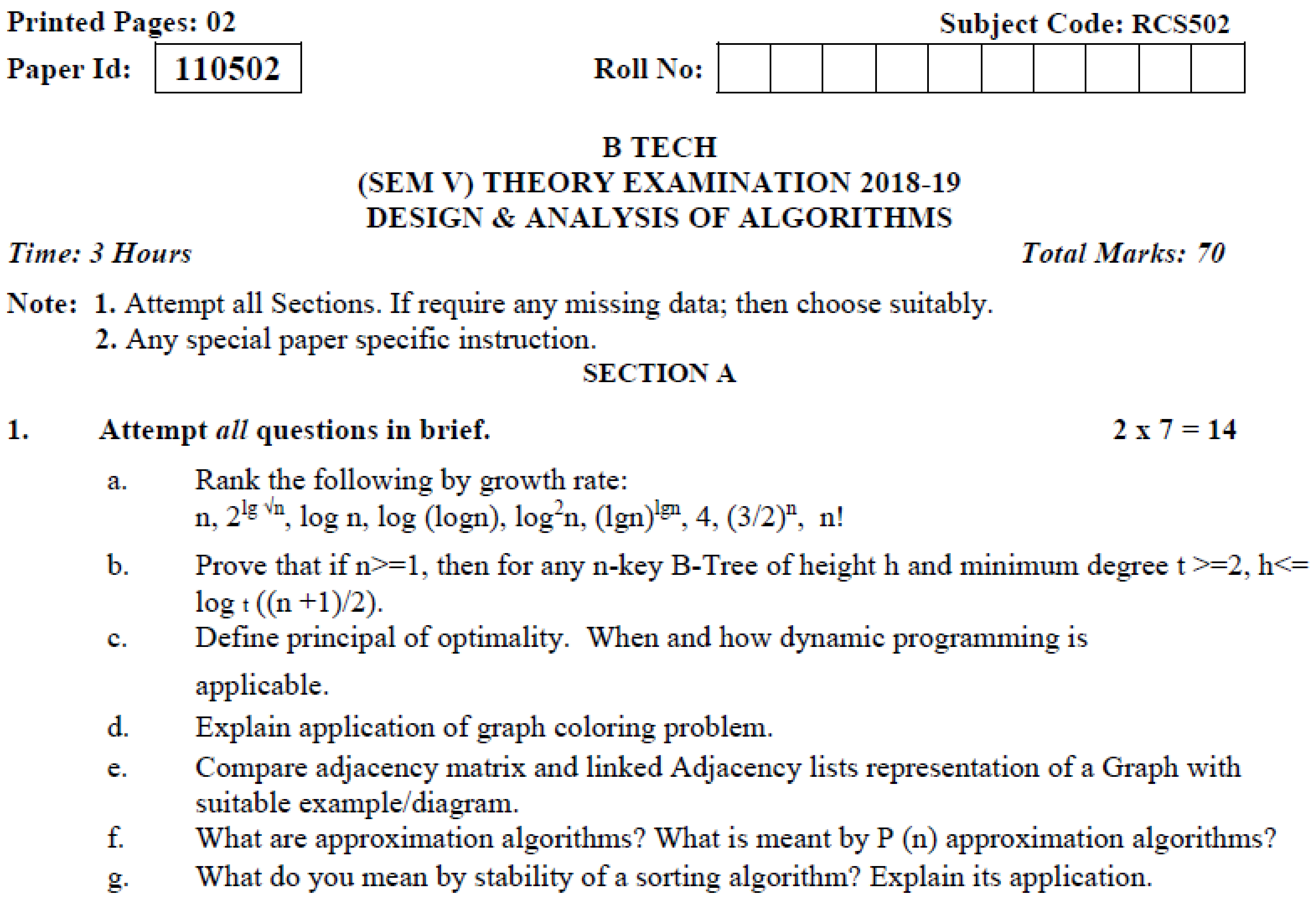
$$\begin{tikzcd}
1 \arrow[d, "6"'] \arrow[dr, "3"] & 2 \arrow[l, "3"'] & \\
3 \arrow[r, bend right, "2"'] & 4 \arrow[l, "1"'] \arrow[u, "1"'] & 5 \arrow[ul, "4"'] \arrow[l, "2"]
\end{tikzcd}$$

7. Attempt any *one* part of the following: 7 x 1 = 7

(a) Describe in detail Knuth-Morris-Pratt string matching algorithm. Compute the prefix function π for the pattern ababbabbabbababbabb when the alphabet is $\Sigma = \{a,b\}$.

(b) What is an approximation algorithm? What is meant by P (n) approximation algorithms? Discuss approximation algorithm for Travelling Salesman Problem.

Old Question Papers(2018-2019)



Printed Pages: 02 **Subject Code: RCS502**

Paper Id: 110502 **Roll No:**

B TECH
(SEM V) THEORY EXAMINATION 2018-19
DESIGN & ANALYSIS OF ALGORITHMS

Time: 3 Hours ***Total Marks: 70***

Note: **1.** Attempt all Sections. If require any missing data; then choose suitably.
2. Any special paper specific instruction.

SECTION A

1. **Attempt *all* questions in brief.** **2 x 7 = 14**

a. Rank the following by growth rate:
n, $2^{\lg \sqrt{n}}$, $\log n$, $\log(\log n)$, $\log^2 n$, $(\lg n)^{\lg n}$, 4, $(3/2)^n$, $n!$

b. Prove that if n>=1, then for any n-key B-Tree of height h and minimum degree t >=2, h<= $\log_t((n+1)/2)$.

c. Define principal of optimality. When and how dynamic programming is applicable.

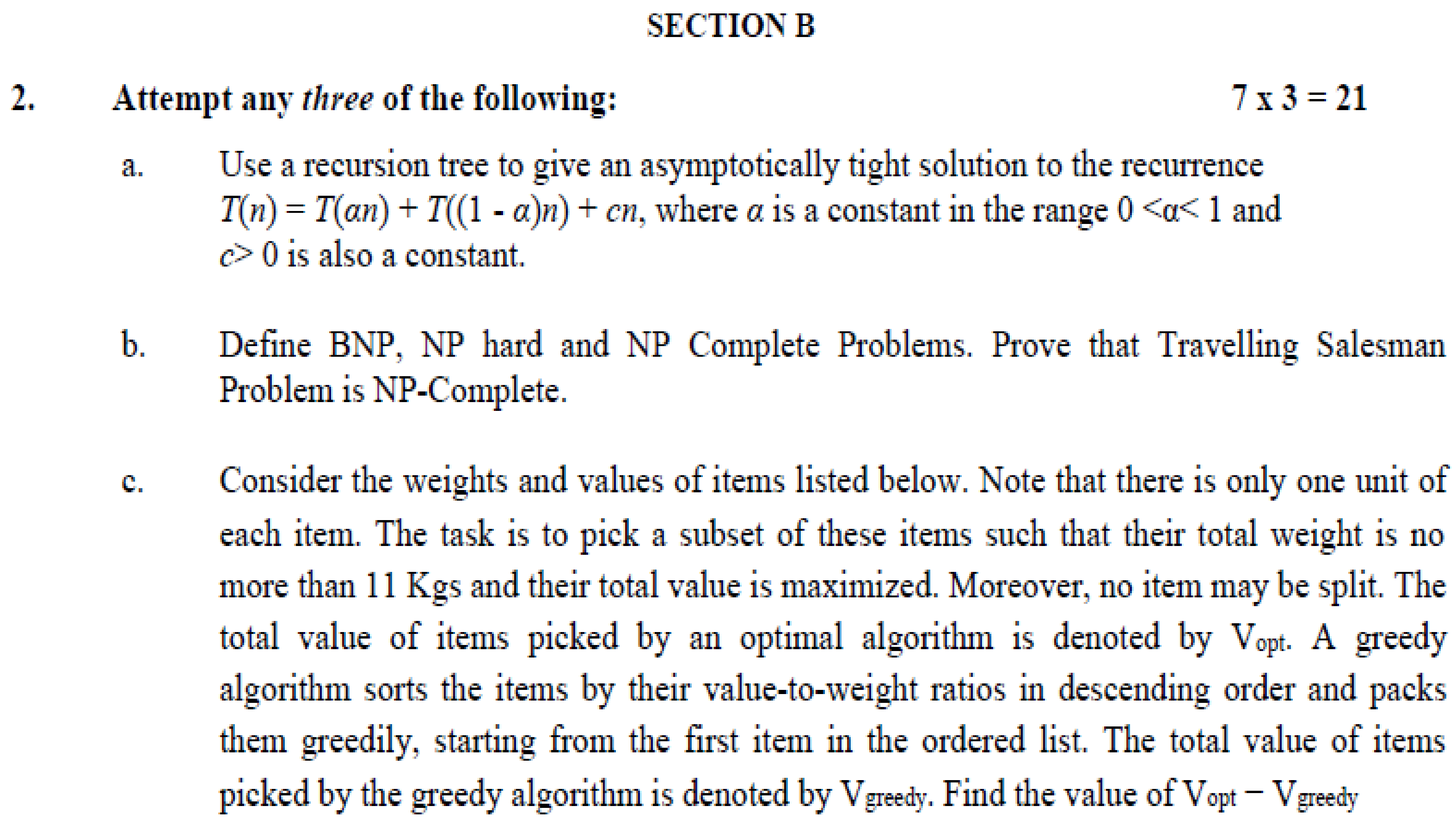
d. Explain application of graph coloring problem.

e. Compare adjacency matrix and linked Adjacency lists representation of a Graph with suitable example/diagram.

f. What are approximation algorithms? What is meant by P (n) approximation algorithms?

g. What do you mean by stability of a sorting algorithm? Explain its application.

Old Question Papers(2018-2019)



SECTION B

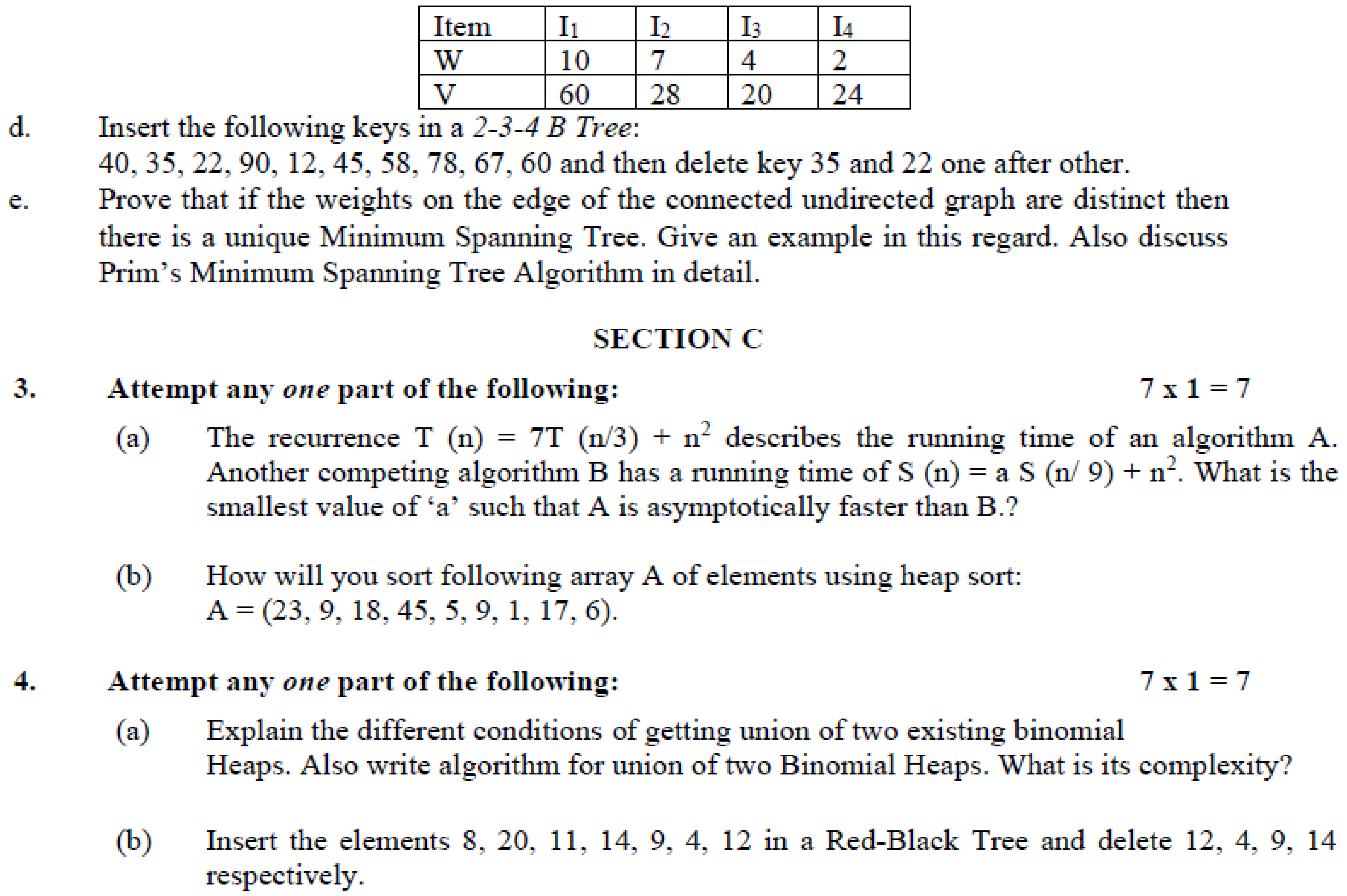
2. **Attempt any *three* of the following:** **7 x 3 = 21**

a. Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1 - \alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.

b. Define BNP, NP hard and NP Complete Problems. Prove that Travelling Salesman Problem is NP-Complete.

c. Consider the weights and values of items listed below. Note that there is only one unit of each item. The task is to pick a subset of these items such that their total weight is no more than 11 Kgs and their total value is maximized. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt}. A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy algorithm is denoted by V_{greedy}. Find the value of $V_{opt} - V_{greedy}$

Old Question Papers(2018-2019)



Item	I_1	I_2	I_3	I_4
W	10	7	4	2
V	60	28	20	24

d. Insert the following keys in a *2-3-4 B Tree*:
40, 35, 22, 90, 12, 45, 58, 78, 67, 60 and then delete key 35 and 22 one after other.

e. Prove that if the weights on the edge of the connected undirected graph are distinct then there is a unique Minimum Spanning Tree. Give an example in this regard. Also discuss Prim's Minimum Spanning Tree Algorithm in detail.

SECTION C

3. Attempt any *one* part of the following: 7 x 1 = 7

(a) The recurrence $T(n) = 7T(n/3) + n^2$ describes the running time of an algorithm A. Another competing algorithm B has a running time of $S(n) = a\,S(n/9) + n^2$. What is the smallest value of 'a' such that A is asymptotically faster than B.?

(b) How will you sort following array A of elements using heap sort:
A = (23, 9, 18, 45, 5, 9, 1, 17, 6).

4. Attempt any *one* part of the following: 7 x 1 = 7

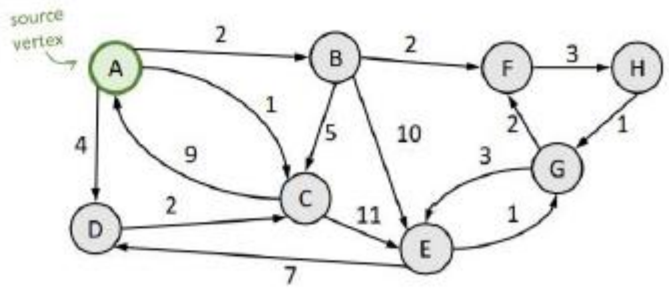
(a) Explain the different conditions of getting union of two existing binomial Heaps. Also write algorithm for union of two Binomial Heaps. What is its complexity?

(b) Insert the elements 8, 20, 11, 14, 9, 4, 12 in a Red-Black Tree and delete 12, 4, 9, 14 respectively.

Old Question Papers(2018-2019)

5. **Attempt any *one* part of the following:** **7 x 1 = 7**

(a) When do Dijkstra and the Bellman-Ford algorithm both fail to find a shortest path?Can Bellman ford detect all negative weight cycles in a graph?Apply Bellman Ford Algorithm on the following graph:



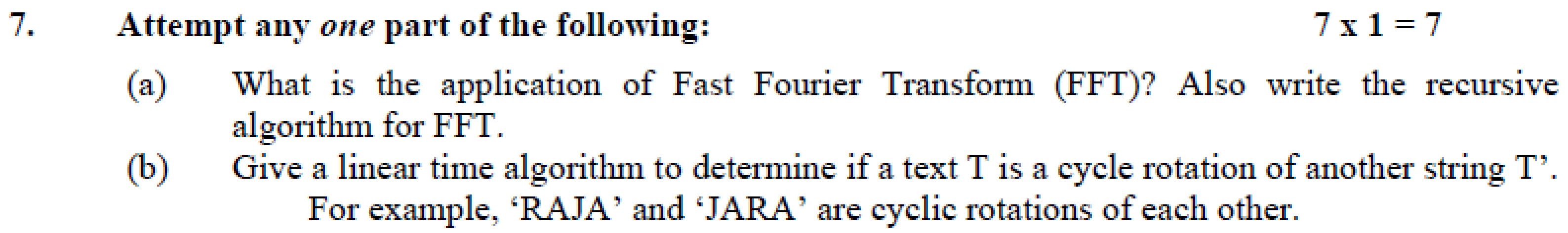
(b) Given an integer x and a positive number n, use divide & conquer approach to write a function that computes $x^n$with time complexity O (logn).

6. **Attempt any *one* part of the following:** **7 x 1 = 7**

(a) Solve the Subset sum problem using Backtracking, where n=4, m=18, w [4] = {5, 10, 8, 13}

(b) Give Floyd War shall algorithm to find the shortest path for all pairs of vertices in a graph. Give the complexity of the algorithm. Explain with example.

Old Question Papers(2018-2019)

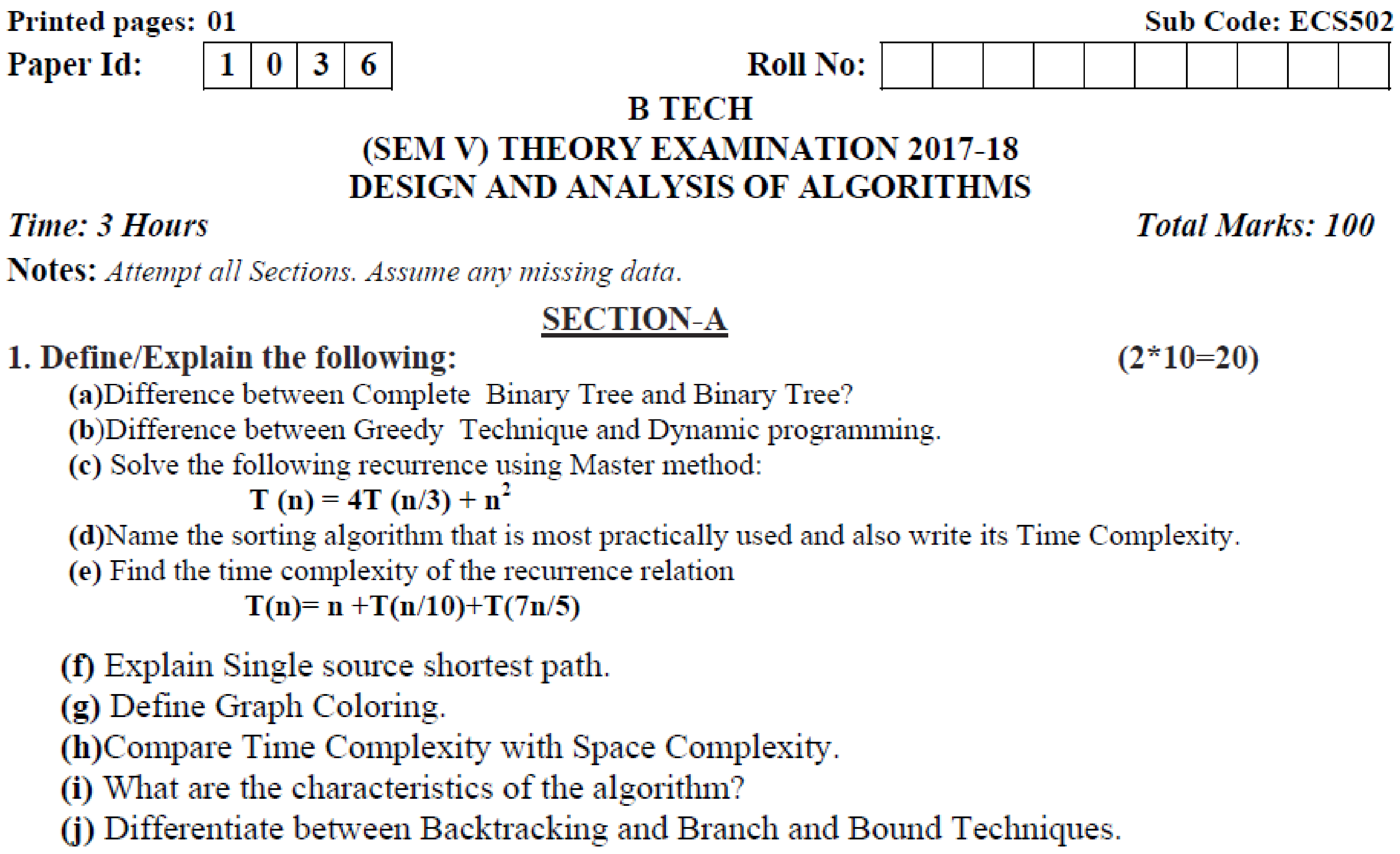


7. **Attempt any *one* part of the following:** **7 x 1 = 7**

(a) What is the application of Fast Fourier Transform (FFT)? Also write the recursive algorithm for FFT.

(b) Give a linear time algorithm to determine if a text T is a cycle rotation of another string T'. For example, 'RAJA' and 'JARA' are cyclic rotations of each other.

Old Question Papers(2017-2018)



Printed pages: 01 Sub Code: ECS502

Paper Id: 1 0 3 6 Roll No:

B TECH

(SEM V) THEORY EXAMINATION 2017-18

DESIGN AND ANALYSIS OF ALGORITHMS

Time: 3 Hours ***Total Marks: 100***

Notes: *Attempt all Sections. Assume any missing data.*

SECTION-A

1. Define/Explain the following: **(2*10=20)**

(a) Difference between Complete Binary Tree and Binary Tree?

(b) Difference between Greedy Technique and Dynamic programming.

(c) Solve the following recurrence using Master method:

$$T(n) = 4T(n/3) + n^2$$

(d) Name the sorting algorithm that is most practically used and also write its Time Complexity.

(e) Find the time complexity of the recurrence relation

$$T(n)= n +T(n/10)+T(7n/5)$$

(f) Explain Single source shortest path.

(g) Define Graph Coloring.

(h) Compare Time Complexity with Space Complexity.

(i) What are the characteristics of the algorithm?

(j) Differentiate between Backtracking and Branch and Bound Techniques.

NIET Greater Noida GET FUTURE READY

Old Question Papers(2017-2018)

SECTION-B

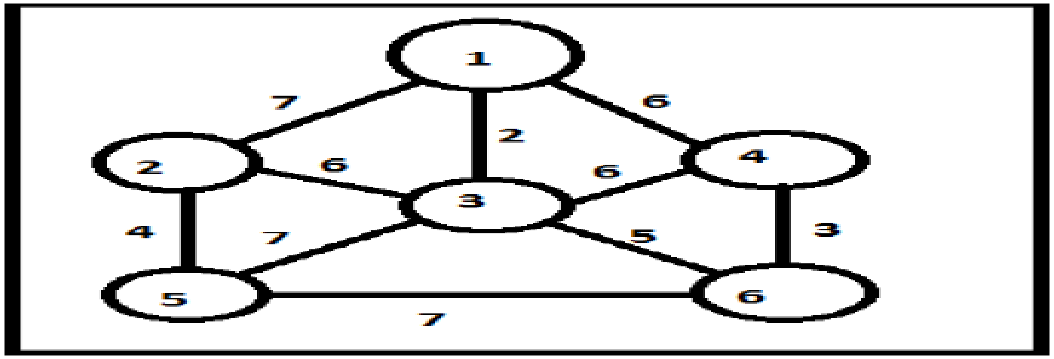
2. Attempt any three of the following: **(10×3=30)**

(a) Solve the following By Recursion Tree Method

T(n)=n+T(n/5)+T(4n/5)

(b) Insert the following information **F,S,Q,K,C,L,H,T,V,W,M,R,N,P,A,B,X,Y,D,Z,E,G,I.** Into an empty B-tree with degree t=3.

(c) What is Minimum Cost Spanning Tree? Explain Kruskal's Algorithm and Find MST of the Graph. Also write its Time-Complexity



(d) What is Red-Black tree? Write an algorithm to insert a node in an empty red-black tree explain with suitable example.

(e) Explain HEAP-SORT on the array. Illustrate the operation of HEAP-SORT on the array

A= {6, 14, 3, 25, 2, 10, 20, 7, 6}

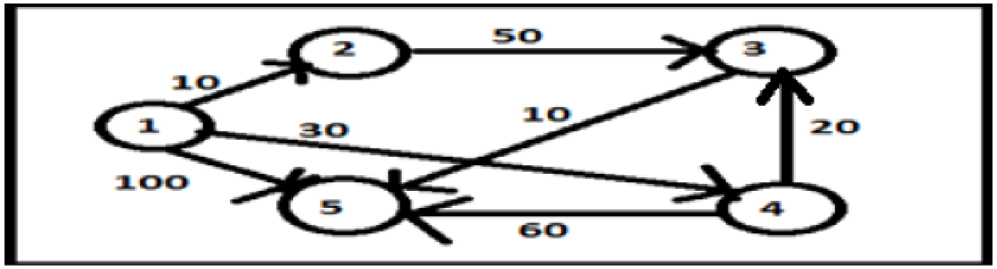
Old Question Papers(2017-2018)

SECTION C

3. Attempt any one part of the following: (10 x 1=10)

(a) Explain Convex –Hull problem.

(b) Find the shortest path in the below graph from the source vertex 1 to all other vertices by using Dijkstra's algorithm.

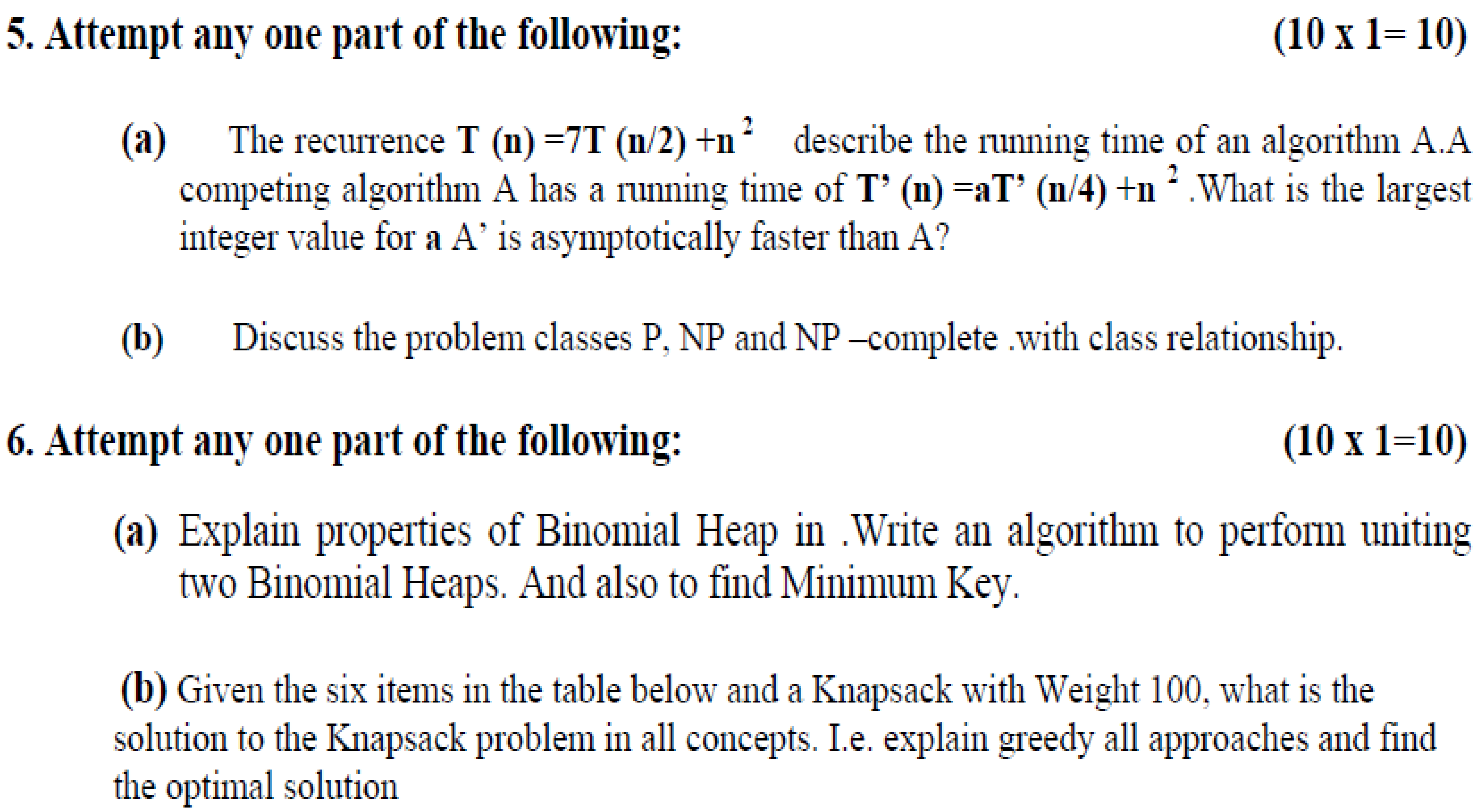


4. Attempt any one part of the following: (10 x 1=10)

(a) What is backtracking? Discuss sum of subset problem with the help of an example.

(b) Write down an algorithm to compute Longest Common Subsequence (LCS) of two given strings and analyze its time complexity.

Old Question Papers(2017-2018)



5. Attempt any one part of the following: **(10 x 1= 10)**

(a) The recurrence **$T(n) = 7T(n/2) + n^2$** describe the running time of an algorithm A.A competing algorithm A has a running time of **$T'(n) = aT'(n/4) + n^2$** .What is the largest integer value for **a** A' is asymptotically faster than A?

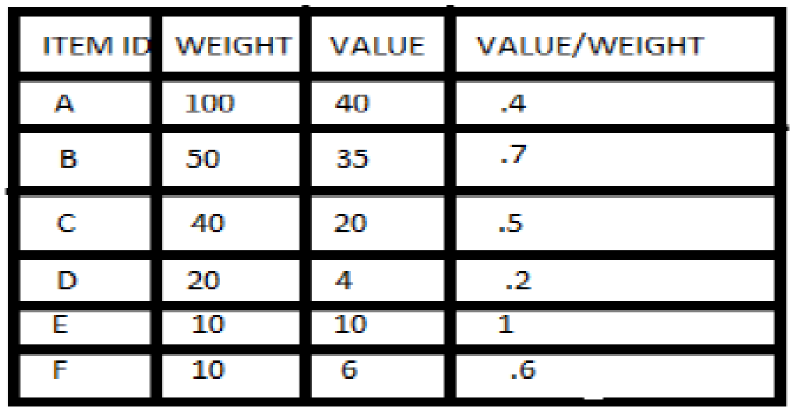
(b) Discuss the problem classes P, NP and NP –complete .with class relationship.

6. Attempt any one part of the following: **(10 x 1=10)**

(a) Explain properties of Binomial Heap in .Write an algorithm to perform uniting two Binomial Heaps. And also to find Minimum Key.

(b) Given the six items in the table below and a Knapsack with Weight 100, what is the solution to the Knapsack problem in all concepts. I.e. explain greedy all approaches and find the optimal solution

Old Question Papers(2017-2018)



ITEM ID	WEIGHT	VALUE	VALUE/WEIGHT
A	100	40	.4
B	50	35	.7
C	40	20	.5
D	20	4	.2
E	10	10	1
F	10	6	.6

7. Attempt any one part of the following: **(10 x 1=10)**

(a) Compute the prefix function π for the pattern P= a b a c a b using KNUTH-MORRIS –PRATT Algorithm. Also explain Naïve String Matching algorithm.

(b) Explain Approximation and Randomized algorithms.

Expected Questions for University Exam

1. What are the different applications of DFT and FFT? [CO5]
2. Discuss the string matching algorithms along with an example. [CO5]
3. How is approximation algorithm different from randomized algorithm? [CO5]
4. Explain Naive string matching algorithm. [CO5]
5. Discuss the problem classes, NP, P, NPC along with their relationship. [CO5]

Recap Of Unit

This unit gives the insights of different classes of problems , P Class, NP class, NPC. The different string matching algorithms have been discussed. The Approximation algorithm and Randomized algorithm have also been explained.

Unit 5

Thank You