FR. Conceicao Rodrigues College Of Engineering

Department of Computer Engineering

S.E. (Computer) (semester III)

(2018-2019)

Subject: Analysis of Algorithms

Subject Code: CSC 402

Course Outcomes and Assessment Plan

Syllabus:

Course Objectives:

- To provide mathematical approach for Analysis of Algorithms
- To solve problems using various strategies
- To analyze strategies for solving problems not solvable in polynomial time.

Course Outcomes:

At the end of the course student will be able to

1. Analyze the running time and space complexity of algorithms.
2. Describe, apply and analyze the complexity of divide and conquer strategy.
3. Describe, apply and analyze the complexity of greedy strategy.
4. Describe, apply and analyze the complexity of dynamic programming strategy.
5. Explain and apply backtracking, branch and bound and string matching techniques to deal with some hard problems.
6. Describe the classes P, NP, and NP-Complete and be able to prove that a certain problem is NP-Complete.

Module 1 Introduction to analysis of algorithm - 12 HRS

Performance analysis, space and time complexity, Growth of function – Big –Oh, Omega, Theta notation, Mathematical background for algorithm analysis, Analysis of selection sort, insertion sort. **Recurrences: -**The substitution method, Recursion tree method, Master method

Divide and Conquer Approach: General method, Analysis of Merge sort, Analysis of Quick sort, Analysis of Binary search, Finding minimum and maximum algorithm and analysis, Stassen's matrix multiplication

Module 2: Dynamic Programming Approach: 08 HRS

General Method, Multistage graphs, single source shortest path, all pair shortest path, Assembly-line scheduling, 0/1 knapsack, Travelling salesman problem, Longest common subsequence

Module 3: Greedy Method Approach: 06 HRS

General Method ,Single source shortest path, Knapsack problem, Job sequencing with deadlines Minimum cost spanning trees-Kruskal and prim's algorithm ,Optimal storage on tapes

Module 4: Backtracking and Branch-and-bound: 08 HRS

General Method, 8 queen problem(N-queen problem) ,Sum of subsets, Graph coloring ,15 puzzle problem, Travelling salesman problem.

Module 5 :String Matching Algorithms: 06 HRS

The naïve string matching Algorithms, The Rabin Karp algorithm, String matching with finite automata, The knuth-Morris-Pratt algorithm

Module 6 : Non-deterministic polynomial algorithms: 08 HRS

Polynomial time, Polynomial time verification NP Completeness and reducibility NP Completeness proofs Vertex Cover Problems Clique Problems

Text Books:

1. T.H.coreman , C.E. Leiserson,R.L. Rivest, and C. Stein, “Introduction to algorithms”, 2nd edition , PHI publication 2005.
2. 2. Ellis horowitz , Sartaj Sahni , S. Rajsekaran. “Fundamentals of computer algorithms” University Press

Reference Books:

1. Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, “Algorithms”, Tata McGraw- Hill Edition.
2. S. K. Basu, “Design Methods and Analysis of Algorithm”, PHI.
3. John Kleinberg, Eva Tardos, “Algorithm Design”, Pearson.
4. Michael T. Goodrich, Roberto Tamassia, “Algorithm Design”, Wiley Publication.

Course Outcomes:

Upon completion of this course students will be able to:

CSC 402.1 : Apply the methods for analyzing the complexity of the algorithms. **(Apply)**
CSC 402.2 : Analyze different techniques of algorithm design.(greedy,dynamic,divide and conquer, backtracking, branch and bound). **(Analyze)**
CSC 402.3 : Analyze different String matching techniques. **(Analyze)**
CSC 402.4 : Implement algorithms using different designing techniques. **(Apply)**

Mapping of CO and PO/PSO

Relationship of course outcomes with program outcomes: Indicate 1 (low importance), 2 (Moderate Importance) or 3 (High Importance) in respective mapping cell.

	PO1 (Engg Know)	PO2 (Ana)	PO3 (De sign)	PO4 (inve stiga)	PO5 (tools)	PO6 (eng g Soci)	PO7 (Env)	PO8 (Eth)	PO9 (ind Team)	PO10 (comm.)	PO1 1 (PM)	PO1 2 (life Long)
CSC402. 1	3	2										
CSC402. 2	3	3										
CSC402. 3	3	3										
CSC402. 4	3	3	3						1			
Course To PO	3	3	1						1			

CO	**PSO1**	**PSO2**
CSC402.1	3	2
CSC402.2	3	2
CSC402.3	3	2
CSC402.4	3	2
Course to PSO	3	2

Justification

PO1: CSC 402.1, CSC 402.2, CSC 402.3 and CSC402.4 maps to PO1 as engineering graduates apply the knowledge of mathematics and computer programming knowledge for providing solution to complex engineering problem.

PO2: CSC 402.1, CSC 402.2, CSC 402.3 ,CSC 402.4 maps to PO1 as engineering graduates identify and formulate a solution to a problem by analyzing efficiency of different algorithms using their time and space complexities ,selecting a design technique (greedy,dynamic,backtracking) as per the requirement of solution .

PO3: CSC 402.4 maps to PO3 because engineering graduates design a programmed solution to a problem using any high level programming language such as C,C++.

PO9: CSC402.4 maps to PO9 as students worked in a team for developing solution to real world problem by applying proper strategy

PSO1: CSC 402.1 to CSC402.4 maps to PSO1 because the graduates will be able to apply knowledge learnt in the subject to provide solution to real world problems.

PSO2: CSC 402.1 to CSC 402.4 maps to PSO2 as the students design and implement a programmed solution for a real world problem.

Assessment Tools:

Course Outcome	Assessment Tool Direct (weightage: 80%)	Assessment Tool Indirect (weightage= 20%)
CO1: Apply the methods for analyzing the complexity of the algorithms. (PO1)	Test 1 (20%) Postlab Assignment (10%) Assignment 1(20%) Quiz (10%) University Exam (30%) Gate questions(10%)	Course Exit Survey
CO2: Analyze different techniques of algorithm design.(greedy, dynamic, divide and conquer, backtracking, branch and bound). .	Test1+Test2 (20%) Postlab assignment(10%) Assignment 1(20%) Quiz (10%) University Exam(30%) Gate questions(10%)	

CO3: Analyze different String matching techniques.	Test 2(20%) Assignment 2(20%) Post lab assignment(20%) University Exam(30%) Gate questions(10%)	
CO4: Implement algorithms using different design strategies. (PO4)	Lab Work(50%) University Exam(20%) Assignment 2 marks(10%) Real world problem (20%)	

CO Assessment Tools:

CSC402.1: Direct Methods(80%): Unit Test 1 + PostLab + Assignment 1+Quiz+UniExam+Gate_Quest

CO1dm = 0.2T +0.1PLab+0.2Assignment+0.1Quiz + 0.3Uniexam+0.1Gate_Quest

InDirect Methods(20%): Course exit survey

CO1idm

CSC402.1 = 0.8*CO1dm + 0.2* CO1idm

CPC501.2:Direct Methods (80%):

Unit Test1&2+PostLab+Assignment+Quiz+UniExam+Gate_Quest

CO2dm = 0.2T +0.1PLab+ 0.2Assig+ 0.1Quiz+0.3Uniexam+0.1Gate_Quest

InDirect Methods(20%): Course exit survey

CO2idm

CSC402.2 = 0.8*CO2dm + 0.2* CO2idm

CPC501.3: Direct Methods (80%): Unit Test 2+PostLab+Assignment+Quiz+UniExam

CO3dm = 0.20T +0.2PLab+ 0.2Assig+ 0.3Uniexam+0.1Gate_Quest

InDirect Methods(20%): Course exit survey

CO3idm

CSC402.3 = 0.8*CO3dm + 0.2* CO3idm

CPC501.4: Direct Methods (80%): Lab assignments+Uniexam+Assig2+Real_world_problem

CO4dm=0.5LabAssignment+0.3UniExam+0.2Assign+0.2Real_world_Problem

InDirect Methods(20%): Course exit survey

CO4idm

CSC402.4 = 0.8*CO4dm + 0.2* CO4idm

Rubrics for Lab Experiments:

Sr. No	Performance Indicator	Excellent	Good	Below Average
1.	Coding Standards [4M]	The code adheres to all standards. The code is exceptionally well organized and very easy to follow. Comments are complete and useful; variables' purposes are clearly communicated by their names. [4 marks]	There may be some minor failures to adhere to standards, for instance, indentation may be inconsistent, some lines may be too long, or a few variables may have unobvious names or be undocumented. [2 marks]	There are major problems with the program's design or coding style that would interfere with its comprehension, reuse, or maintenance. File or function comments may be sketchy, variable descriptions or names may be unenlightening. The code may be poorly formatted.[0.5-1M]
2	Output validation [2M]	output is obtained for different cases of input.[2M]	Output is obtained only for some subsets of input.[1M]	no output is obtained. [0 mark]
3	Post Lab Questions [2M]	Answers to all questions are correct and explained in depth. [2 marks]	Answers to most of the questions are correct but not explained in depth. [1 marks]	Answers to most of the questions are incorrect. [0 mark]
4	Promptness {2M]	The laboratory report is submitted on time [2 mark]	The laboratory report is submitted next day. [1 marks]	The laboratory report is submitted in next practical session. [0 marks]

Rubrics for Assignments:

Indicator	Excellent	Good	Below average
Timeline (2)	submitted on time or early (2)	Submitted next day (1)	Submitted in same week (0.5)
Organization (2)	Well organized, neat and clear handwriting, neat diagrams with all labels.(2)	Organized to some extent, diagrams and handwriting is neat with some missing labels(1)	Poorly organized, diagrams incomplete (0.5)
Level of content (3)	All points are covered(3) and answered accurately	Some important points are omitted / addressed minimally (1-2)	Many important points are missing and the answers are not accurate. (1-0)
Knowledge about the topic (3)	All Concepts of a topic are clear and knows the application to real world problems (3)	All Concepts of a topic are mostly clear lacks understanding about the application to real world problems (2-1)	Poor understanding of concepts and application to real world problems.(1-0)

Lesson Plan

Module 1: Introduction to Analysis of Algorithms

Lecture No.	Date		Topic	Content Delivery Method
	Planned	Actual		
1	1/1/2019	2/1/2019	**Introduction to analysis of algorithms:** Introduction to subject and fundamentals of algorithms. What is meant by efficient algorithm?	Chalk and board
2	2/1/2019	3/1/2019	Efficiency of algorithms, Time and Space Complexities Fundamentals	Chalk and board
3	3/1/2019	3/1/2019	Growth of Function – Big O, Omega, Theta	Chalk and board
4	4/1/2019	7/1/2019	Calculation of time complexity for code samples	Chalk and board
5	7/1/2019	8/1/2019	Calculation of time complexity for code samples continued	Chalk and board
6	8/1/2019	9/1/2019	Finding space complexity for code samples	Chalk and board
7	9/1/2019	10/1/2019	Finding Complexities of Bubble, Insertion & Selection Sort & Linear Search	Chalk and board , Lab performance
8	10/1/2019	11/1/2019	**Recurrences:** Solving recurrence using Iteration Method	Chalk and board
9	11/1/2019	14/1/2019	Solving recurrence using Recursion Tree	Chalk and board
10	14/1/2019	15/1/2019	Solving recurrence using Master Method	Chalk and board
11	15/1/2019	17/1/2019	**Divide and Conquer Approach:** General Method of Divide & Conquer, Analysis of Binary Search	Chalk and board, simulation
12	17/1/2019	18/1/2019	Analysis of Merge Sort and quick sort	Chalk and board , Lab performance, animation
13	18/1/2019	21/1/2019	Minmax algorithm	Chalk and board , Lab performance
14	21/1/2019	22/1/2019	Strassen’s matrix multiplication	Chalk and board

Module 3: **Greedy Method**				
15	22/01/2019	29/01/2019	General Method, Knapsack Problem	Chalk and board , Lab performance
16	28/01/2019	29/01/2019	Job Sequencing with deadline	Chalk and board
17	29/01/2019	1/2/2019	SSSP (Dijkstra's Algo)	Chalk and board , visualization,Lab performance
18	1/2/2019	7/2/2019	MST- Prims	Chalk and board , visualization , Lab performance
19	7/2/2019	8/2/2019	MST – Kruskal	Chalk and board , Lab performance, visualization
20	8/2/2019	18/2/2019	Optimal Storage on tapes	Chalk and board
Module 2: **Dynamic Programming**				
21	18/2/2019	21/2/2019	General Method, 0/1 Knapsack	Chalk and board , Lab performance
22	21/2/2019	22/2/2019	Single Source Shortest Path	Chalk and board , Lab performance, visualization
23	22/2/2019	25/2/2019	All pair shortest Path	Chalk and board , Lab performance, visualization
24	25/2/2019	26/2/2019	MultiStage Graph	Chalk and board
25	26/2/2019	1/3/2019	Travelling Salesman Problem	Chalk and board, visualization
26	28/2/2019	1/3/2019	Longest common subsequence	Chalk and board , Lab performance
27	1/3/2019	5/3/2019	Assembly line schedulling	Chalk and board , Lab performance

Module 4: **Backtracking and branch and bound**				
28	5/3/2019	8/3/2019	General Method of backtracking, n queen problem	Chalk and board , Lab performance
29	7/3/2019	11/3/2019	Sum of Subsets	Chalk and board , Lab performance
30	8/3/2019	12/3/2019	Graph Coloring	Chalk and board , Lab performance
31	11/3/2019	14/3/2019	General Method of branch and bound, 15 puzzle problem	Chalk and board
32	12/3/2019	18/3/2019	Travelling Salesman Problem	Chalk and board
Module 5: **String Matching algorithms**				
33	14/3/2019	22/3/2019	Naïve String Maching	Chalk and board
34	28/3/2019	26/3/2019	Rabin Karp Algo	Chalk and board
35	29/3/2019	26/3/2019	KMP Algo	Chalk and board
36	25/3/2019	27/3/2019	String matching with Finite Automata	Chalk and board
Module 6: **Non Deterministic Polynomial algorithms**				
37	26/3/2019	29/03/2019	Polynomial time ,Polynomial time verification	Chalk and board , handouts
38	28/3/2019	1/4/2019	NP completeness and reducibility	Chalk and board , handouts
39	29/3/2019	2/4/2019	Vertex cover problems, Clique Problem	Chalk and board, handouts
40	**1/4/2019**	**4/4/2019**	**Multiplying long integers(divide and Conquer(Content Beyond Syllabus)**	**Chalk and board, handouts**
41	**2/4/2019**	**5/4/2019**	**Optimal binary search tree(dynamic programming (Content Beyond Syllabus)**	

LAB PLAN

Sr. No.	TITLE	Mapped Co	Planned Week	Actual dates Batch A	Actual dates Batch B	Actual dates Batch C	Actual dates Batch D
1	WAP to implement Modified bubble sort, Insertion sort, Selection sort and derive its complexity.	CO1 and CO4	1 st week	14-01-2019	16-01-2019	18-01-2019	14-01-2019
2	WAP to implement Liner search and binary search and derive its time complexity.	CO1 and CO4	1 st week	14-01-2019	16-01-2019	18-01-2019	14-01-2019
3	WAP to implement Quick sort, randomized quick sort, merge sort and derive its complexity.	CO1 and CO4	2nd week	21-01-2019	23-01-2019	25-01-2019	21-01-2019
4	WAP to implement min max algorithm.	CO2 andCO4	2nd week	21-01-2019	23-01-2019	25-01-2019	21-01-2019
5	WAP to implement fractional knapsack using greedy method.	CO2 and CO4	3rd week	28-01-2019	30-01-2019	01-02-2019	28-01-2019
6	WAP to implement Dijkstra's algorithm.	CO2 and CO4	3rd week	28-01-2019	30-01-2019	01-02-2019	28-01-2019
7	WAP to implement Prim's algorithm	CO2 and CO4	4th week	18-02-2019	20-02-2019	08-02-2019	18-02-2019
8	WAP to implement 0/1 knapsack using	CO2 and CO4	4th week	18-02-2019	20-02-2019	08-02-2019	18-02-2019

	dynamic programming.						
9	WAP to implement Floyd Warshall algorithm.	CO2 and CO4	6th week	25-02-2019	27-02-2019	22-02-2019	25-02-2019
10	WAP to implement bellman ford algorithm.	CO2 and CO4	6th week	11-03-1019	6-03-1019	1-03-1019	11-03-1019
11	WAP to implement N queen problem using backtracking approach.	CO2 and CO4	6th week	18-03-2019	13-03-2019	08-03-2019	18-03-2019
12	WAP to implement sum of subset problem using backtracking approach	CO2 and CO4	7th week	18-03-2019	13-03-2019	08-03-2019	18-03-2019
13	WAP to implement graph coloring using backtracking approach.	CO2 and CO4	8th week	25-03-2019	20-03-2019	22-03-2019	25-03-2019
14	WAP to implement Longest common subsequence.	CO2 and CO4	8th week	25-03-2019	27-03-2019	22-03-2019	25-03-2019
15	WAP to implement Knuth Morris Pratt Algorithm	CO3 and CO4	9th week	01-04-2019	03-04-2019	05-04-2019	01-04-2019
16	WAP to implement Assembly Line schedulling	CO2 and CO4	9th week	01-04-2019	03-04-2019	05-04-2019	01-04-2019

AOA ASSIGNMENT 1

CLASS: SE COMPS (SEM IV) YEAR: 2018-19

CO1: Analyze time and space complexity of algorithms
CO2: Analyze various strategies of algorithm design

Q. 1) Let f(n) = $16n^4$ +10 n log n and g(n) = 8758 n^3 log n + 9248 n^2. Which of the following is true?

i) f(n) is O(g(n)) and g(n) is O(f(n)).
ii) f(n) is O(g(n)), but g(n) is not O(f(n))
iii) g(n) is O(f(n)), but f(n) is not O(g(n))
iv) f(n) is not O(g(n)) and g(n) is not O(f(n))

Q. 2) If T(n) = n √ n then:

i) T(n) is O(n^3)
ii) T(n) is O(n log n)
iii) T(n) is O(n)
iv) None of these

Q 3) A new algorithm MaxPack for optimally packing furniture in a transportation container claims to have worst cas A new algorithm MaxPack for optimally packing furniture in a transportation container claims to have worst case complexity O(n2 log n), where n is the number of items to be packed.

From this, we can conclude that:

i) For every n, for every input of size n, MaxPack requires time proportional to n2 log n.
ii) For some n, for every input of size n, MaxPack requires time proportional to n2 log n.
iii) For every n, every input of size n can be solved by MaxPack within time proportional to n2 log n.
iv) For every n, there is an input of size n for which MaxPack requires time proportional to n2 log n.

Q.4) Your final exams are over and you are catching up on sports on TV. You have a schedule of interesting matches from all over the world during the next week. You hate to start or stop watching a match midway, so your aim is to watch as many complete matches as possible during the week. Suppose there are n such matches {M1 ,M2 ,…,Mn } available during the coming week. The matches are ordered by starting time, so for each i ∈ {1, 2, …, n−1}, Mi starts before Mi+1. However, match Mi may not end before Mi+1 starts, so for each i ∈ {1,2,…,n−1}, Next[i] is the smallest j > i such that Mj starts after Mi finishes. Given the sequence {M1 , M2 , … , Mn } and the values Next[i] for each i ∈ {1, 2, …, n−1}, your aim is to compute the maximum number of complete matches that can be watched.

Let Watch[i] denote the maximum number of complete matches that can be watched among {Mi , Mi+1, …, Mn }.

A) Which of the following is a correct recursive formulation of Watch[i]?

   i) Watch[n] = 1
      Watch[i] = max(Watch[Next[i]], 1 + Watch[i + 1]), i ∈ {1, 2, …, n−1}

   ii) Watch[1] = 1
      Watch[i] = max(Watch[i − 1], 1 + Watch[Next[i − 1]]), i ∈ {2, 3, …, n}

   iii) Watch[n] = 1
      Watch[i] = max(1 + Watch[Next[i]], Watch[i + 1]), i ∈ {1, 2, …, n−1}

   iv) Watch[1] = 1
      Watch[i] = max(Watch[i − 1] + 1, Watch[Next[i − 1]]), i ∈ {2, 3, …, n}

B) What is a good order to compute Watch[i] using dynamic programming?

   i) From Watch[1] to Watch[n]

   ii) From Watch[n] to Watch[1]

   iii) Either from Watch[1] to Watch[n] or from Watch[n] to Watch[1]

   iv) None of these

C) Suppose the list of matches to be watched is presented in the form [(11,55),(22,31),(33,45),(44,52),(56,62),(57,58),(59,63),(64,70),(71,80)] where each match Mi is represented by a pair (Si ,Ti) indicated its starting time and ending time. To be able to watch both Mi and Mj , for j > i, it must be the case that Sj > Ti .

   What is the maximum number of matches you can watch in this case

   i) 4
   ii) 5
   iii) 6
   iv) 7

Q.5) We are given a directed graph, using an adjacency matrix representation. For each vertex v, we want to compute the set of incoming edges (u, v). Which of the following is the most accurate description of the complexity of this computation. (Recall that n is the number of vertices and m is the number of edges)

i)O(n)

ii)O(n+m)

iii)O(n^2)

iv)O(m)

Q.6) Consider the following strategy to solve the single source shortest path problem with edge weights from source s. 1. Replace each edge with weight w by w edges of weight 1 connected by new intermediate nodes. 2. Run BFS(s) on the modified graph to find the shortest path to each of the original vertices in the graph. Which of the following statements is correct?

i) This strategy will solve the problem correctly but is not as efficient as Dijkstra's algorithm.

ii) This strategy will solve the problem correctly and is as efficient as Dijkstra's algorithm.

iii) This strategy will not solve the problem correctly.

Q.7) An airline charges a fixed price for each direct flight. For each sequence of hopping flights, the ticket price is the sum of the fares of the individual sectors. TripGuru has pre calculated the cheapest routes between all pairs of cities so that it can offer an optimum choice instantly to customers visiting its website. Overnight, the government has added a 13% luxury service surcharge to the cost of each individual flight. Which of the following most accurately describes the impact of this surcharge on TripGuru's computation?

i) There is no impact. Cheapest routes between all pairs of cities remains unchanged.
ii) The surcharge favours hopping flights with fewer sectors. TripGuru should recompute any cheapest route where there is a shorter route in terms of number of flights.
iii) The surcharge favours hopping flights with more sectors. TripGuru should recompute any cheapest route where there is a longer route in terms of number of flights.
iv) The impact is unpredictable. TripGuru should recompute all cheapest routes.

AOA ASSIGMENT 2

CLASS: SE COMPS (SEM IV) YEAR: 2018-19

CO3: Apply string matching techniques to problems.

CO4: Implement the algorithm using different design strategies

Q.1) A shoemaker has *N* orders from customers that he must execute. The shoemaker can work on only one job each day. For each job *i*, it takes T_i days for the shoemaker to finish the job, where T_i is an integer and ($1 \le T_i \le 1000$). For each day of delay before starting to work for the job *i*, shoemaker must pay a fine of S_i ($1 \le S_i \le 10000$) rupees. Your task is to help the shoemaker find the sequence in which to complete the jobs so that his total fine is minimized. If multiple solutions are possible, print the one that is lexicographically least (i.e., earliest in dictionary order).

Solution hint

Sort the jobs in terms of the ratios S_i / T_i. To compare *a/b* and *c/d*, cross-multiply to avoid floating point comparisons. Use a stable sort to get the lexicographically smallest value.

Input format

The first line of input contains an integer *N* ($1 \le N \le 100000$). Each of the next *N* lines contains two space separated integers: the time T_i and fine S_i for job *i*, $1 \le i \le N$.

Output format

You program should print the sequence of jobs with minimal fine. Each job should be represented by its position in the input and each job should appear on a new line, by itself. If multiple solutions are possible, print the one that is lexicographically least (i.e., earliest in dictionary order).

Sample input

4
3 4
1 1000
2 2
5 5

Sample output

2
1
3
4

Q.2) In Siruseri, there are junctions connected by roads. There is at most one road between any pair of junctions. There is no road connecting a junction to itself. The travel time for a road is the same in both directions.

At every junction there is a single traffic light. These traffic lights are a bit peculiar. Starting from time 0, each light flashes green once every *T* time units, where the value of *T* is different for each junction.

A vehicle that is at a junction can start moving along a road only when the light at the current junction flashes green. If a vehicle arrives at a junction between green flashes, it must wait for the next green flash before continuing in any direction. If it arrives at a junction at exactly the same time that the light flashes green, it can immediatly proceed along any road originating from that junction.

You are given a city map that shows travel times for all roads. For each junction *i*, you are given T_i, the time period between green flashes of the light at that junction. Your task is to find the minimum time taken from a given source junction to a given destination junction for a vehicle when the traffic starts.

Solution hint

Use Dijkstra's algorithm. At each phase, from the current shortest time for a given junction, compute when the next green flash will occur to let you travel to its neighbours and use this to update shortest path information.

Input Format

There are *N* junctions and *M* roads. The junctions are identified by integers 1 through *N*.

- The first line of input contains two integers: the source junction and the destination junction.
- The second line contains two integers: *N* and *M*.
- The third line contains *N* integers, $T_1, T_2,\ldots T_N$, describing the time periods at which the traffic lights flash green. The light at junction *i* flashes green at times 0, T_i, $2T_i$, $3T_i$, ...
- The next *M* lines contain information about the *M* roads. Each line has three integers *i*, *j*, l_{ij} , where:
  - *i* and *j* are the junctions connected by this road
  - l_{ij} is the time required to move from junction *i* to junction *j* using this road

Output Format

A single line consisting of a single integer, the time taken by a minimum-time path from source to destination.

Constraints:

- $2 \leq N \leq 300$
- $1 \leq M \leq 14{,}000$
- $1 \leq T\_i \leq 100$
- $1 \leq l_{ij} \leq 100$

Sample Input

1 4
4 5
4 3 2 5
1 2 4
1 3 8
2 3 6
2 4 10
3 4 7

Sample Output

15

Explanation

- 1 to 2 to 4 takes time 4 + 2 (wait till 6) + 10 = 16.
- 1 to 3 to 4 takes time 8 + 0 (no wait) + 7 = 15.
- 1 to 2 to 3 to 4 takes time 4 + 2 (wait till 6) + 6 + 0 (no wait) + 7 = 19.
- 1 to 3 to 2 to 4 takes time 8 + 0 (no wait) + 6 + 1 (wait till 15) + 10 = 25.

Q 3) As we all know, a *palindrome* is a word that equals its reverse. Here are some examples of palindromes: malayalam, gag, appa, amma.

We consider any sequence consisting of the letters of the English alphabet to be a word. So axxb,abbba and bbbccddx are words for our purpose. And aaabbaaa, abbba and bbb are examples of palindromes.

By a *subword* of a word, we mean a contiguous subsequence of the word. For example the subwords of the word abbba are a, b, ab, bb, ba, abb, bbb, bba, abbb, bbba and abbba.

In this task you will be given a word and you must find the longest subword of this word that is also a palindrome.

For example if the given word is abbba then the answer is abbba. If the given word is abcbcabbacba then the answer is bcabbacb.

Solution hint

Any subword of w that is a palindrome is also a subword when w is reversed.

Input format

The first line of the input contains a single integer *N* indicating the length of the word. The following line contains a single word of length *N*, made up of the letters a,b,…, z.

Output format

The first line of the output must contain a single integer indicating the length of the longest subword of the given word that is a palindrome. The second line must contain a subword that is a palindrome and which of maximum length. If there is more than one subword palindrome of maximum length, print the one that is lexicographically smallest (i.e., smallest in dictionary order).

Test Data:

You may assume that $1 \leq N \leq 5000$. You may further assume that in 30% of the inputs $1 \leq N \leq 300$.

Example:

We illustrate the input and output format using the above examples:

Sample Input 1:

```
5
abbba
```

Sample Output 1:

```
5
abbba
```

Sample Input 2:

```
12
```

abcbcabbacba

Sample Output 2:

8
bcabbacb

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Fr. Agnel Ashram,Bandstand,Bandra (west),Mumbai 400050

I Unit Test

Semester/Branch: (IV Computer)
Subject: Analysis of Algorithms **Max. Marks: 20**
Date: 4 th Feb 2019 **Time: 1:00-2:00 p.m.**

CSC402.1 (CO1): Apply the methods for analyzing complexity of the algorithm

CSC402.2 (CO2): Analyze various strategies of design of an algorithm

Q. 1 a)	Arrange the following function in an increasing order n, logn , n^3,n^2,nlogn,2^n,n!	01M	CO1

b)	What is time complexity of fun()?
	
	int fun(int n)
	{ int count = 0;
	for (int i = n; i > 0; i /= 2)
	for (int j = 0; j < i; j++)
	count += 1;
	return count; }
	(A) O(n^2)
	(B) O(nLogn)
	(C) O(n)
	(D) O(nLognLogn)
		02M	CO1

c)	An algorithm takes 6 seconds for an input size n=10 how much time it will take when n=100 if time complexity is given as a) n^3 b) log n	02M	CO1

<table>
<tr><td>d)</td><td>What is recurrence for worst case of Quick Sort and what is the time complexity in Worst case?

a) Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2)

b) Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2)

c) Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn)

d) Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn)</td><td>01M</td><td>CO1</td></tr>
<tr><td>e)</td><td>For the following code determine space complexity
int gcd(n,m)
{
 if (n%m ==0) return m;
 n = n%m;
 return gcd(m, n);
}</td><td>02M</td><td></td></tr>
<tr><td>Q.2</td><td>Find an optimal solution to the knapsack instance n=7, M=15

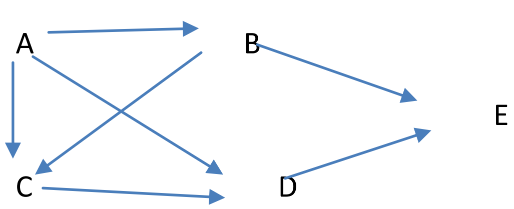
Profit ={10,5,15,7,6,18,3}

Weight={ 2,3,5,7,1,4,1}

OR

Consider the following processes, their profits and deadlines and apply the suitable algorithm to maximize the profit.

<table>
<tr><td>Jobs</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>Profits</td><td>40</td><td>15</td><td>60</td><td>20</td><td>10</td><td>45</td><td>55</td></tr>
<tr><td>Deadlines</td><td>2</td><td>4</td><td>3</td><td>2</td><td>3</td><td>1</td><td>1</td></tr>
</table></td><td>06M</td><td>CO2</td></tr>
<tr><td>Q. 3</td><td>Apply suitable algorithm to determine optimal path to reach other cities from source city ‘A’


</td><td>06M</td><td>CO2</td></tr>
</table>

	Dist(A,B)=2 , Dist(A,C)= 6 Dist(A,D) = 12, Dist(B,C)=7, Dist(B,E) =3, Dist(D,E)=3, Dist(C,D)=5 **OR** Multiply two matrices using Divide and conquer strategy $A=\begin{Bmatrix} 2\ 4\ 6\ 3 \\ 1\ 2\ 2\ 1 \\ 3\ 1\ 1\ 3 \\ 1\ 1\ 1\ 1 \end{Bmatrix}$ **B** $=\begin{Bmatrix} 1\ 1\ 1\ 1 \\ 1\ 1\ 1\ 2 \\ 2\ 1\ 1\ 2 \\ 3\ 1\ 1\ 3 \end{Bmatrix}$		

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

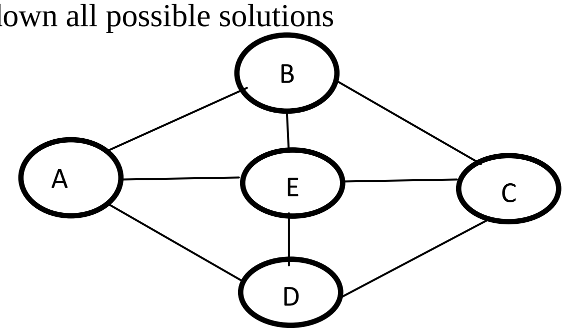
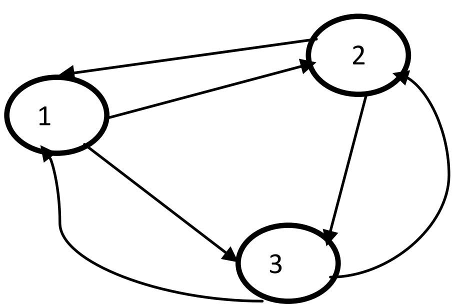
Fr. Agnel Ashram,Bandstand,Bandra (west),Mumbai 400050

II Unit Test

Semester/Branch: (IV Computer)
Subject: Analysis of Algorithms **Max. Marks: 20**
Date: 8 th April 2019 **Time: 1:00-2:00 p.m.**

CSC402.2 (CO2): Analyze various strategies of design of an algorithm
CSC402.3 (CO3): Apply string matching techniques to problems.

Q. 1 a)	Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then x + 10y = ____. a) 33 b) 23 c) 34 d) 43	02M	CO3
b)	In a weighted graph, assume that the shortest path from a source ‘s’ to a destination ‘t’ is correctly calculated using a shortest path algorithm. Is the following statement true? If we increase weight of every edge by 1, the shortest path always remains same. **(A)**Yes	01M	CO2

	(B) No		
c)	Which of the following is not a backtracking algorithm? **(A)** Knight tour problem **(B)** N queen problem **(C)** Tower of hanoi **(D)** M coloring problem	01M	CO2
Q.2	Construct Finite state automata for the pattern ababaca and illustrate its operation on the text abababacabacba OR Compute prefix function for the pattern ababbabbabab. Derive time complexity of prefix function.	05M	CO3
Q. 3	Consider Two DNA strands S1=ACCGGTCGAG and S2= GTCGTTCGGA. Use suitable technique to compare two strands to determine how similar the two strands are.	05M	CO2
Q. 4	A graph given below needs to be colored using 3 colors. Draw state space tree and list down all possible solutions  **OR** Find optimal tour path a salesman should use for a graph given below using branch and bound strategy  Dist(1,2)=4 Dist(3,1)=1 Dist(2,3)=4 Dist(2,1)=3 Dist(1,3)=2 Dist(3,2)=8	06M	CO2