# Algorithm Design - II
# Session: July'2018 - December'2018

**1. Course Number and Name:**

CSE 4131, Algorithm Design II

**2. Credits and Course Format:**

Credits = 4
3 Classes/week, 1hr/Class,
1 Problem Solving Session/Week, 2hr/Problem Solving Session

**3. Target Students:**

Programme: B.Tech. (5th Semester)
Branch: CSE, CS&IT

**4. Instructor's Names:**

1. Mr. Satya Ranjan Das, Assistant Professor (CSE), ITER
Contact # satyadas@soauniversity.ac.in
2. Mr. Paresh Baidya, Faculty (CSE), ITER
Contact # pareshbaidya@soa.ac.in
3. Mrs. Bishnupriya Panda, Faculty (CSE), ITER
Contact # bishnupriyapanda@soa.ac.in
4. Mr. Protik Paul, Faculty (CAM), ITER
Contact #protikpaul@soa.ac.in
5. Mr. Prithviraj Mohanty, Faculty (CS&IT), ITER
Contact # prithwirajmohanty@soa.ac.in
6. Ms. Sushree B.B. Priyadarshini, Faculty (CS&IT), ITER
Contact # sushreepriyadarshini@soa.ac.in
8. Mr. Krishanu Maity, Faculty (CS&IT), ITER
Contact # krishanumaity@soa.ac.in

**5. Text Book and References:**

Text book

1. (T1) The Algorithm Design Manual by Steven S. Skiena, Springer Publication

Reference book

1. (P1)

**6. Specific Course Information:**

**a. Course Description:**

The course topics will include concepts of algorithm complexity, and various algorithmic design patterns like greedy algorithms, dynamic programming, backtracking. Course will also

cover major algorithms and data structures for searching and sorting, graphs, and some optimization techniques.

**b. Prerequisites and/or Co-requisites:**

Prerequisite: CSE 2031(INT), CSE 1002(DM), CSE 3131(AD I)
Co-requisite: CSE 2041(PP)

**7. Course Learning Outcomes:**

By the end of course through lectures, readings, homeworks, lab assignments and exams, students will be able to:

(i) understand the network flow problem and apply it to real-world problems.
(ii) use a greedy approach to solve an appropriate problem and prove if the greedy rule chosen leads to an optimal solution.
(iii)use recursive backtracking to solve an appropriate problem and identify errors in incorrect implementations.
(iv) describe various heuristic problem solving methods.
(v) use dynamic programming to solve an appropriate or provide a recursive solution using memoization.
(vi) - distinguish between computationally tractable and intractable problems.
- define and relate class-P, class-NP and class NP-complete.
- given a problem in NP, define an appropriate certificate and the verification algorithm.
(vii) understand the concept of approximation ratio (with emphasis on constant-factor approximation)
(viii) identify and apply an appropriate algorithmic approach to solve a problem.
(ix) given a numerical problem, explain the challenges to solve it.

**8. Brief List of Topics to Be Covered: (L: Lecture, P: Practical/Problem Solving Session)**

- Weighted Graph Algorithms
- Greedy Method
- Combinatorial Search and Heuristic Methods
- Dynamic Programming
- Intractable Problems and Approximation Algorithms
- A Catalog of Algorithmic Problems
- Data Structures
- Numerical Problems

| Contact Hour | Topics To Be Covered | Remarks(if any) |
|---|---|---|
| Week # 1: | | |
| L 01 | Introduction to the course/subject: Lesson plan; Course Goal; Teaching methodology; Evaluation strategy etc. | |
| L 02 | Review on Sorting and Searching (through Quiz test-1 and Discussion) | |
| L 03 | Review on Graph representation and traversal (through Quiz test-2 and Discussion) | |

| | | |
|---|---|---|
| P 01 | Review on Weighted Graph Algorithms: spanning tree and shortest path (through Quiz test-3 and Discussion) | |
| Week # 2: | | |
| L 04 | Weighted graph algorithms: Network flows and Bipartite matching | To be referred from T1(Sec-6.5) |
| L 05 | Weighted graph algorithms: Network flows and Bipartite matching (contd.) | To be referred from T1(Sec-6.5) |
| L 06 | Weighted graph algorithms: Network flows and Bipartite matching (contd.) | To be referred from T1(Sec-6.5) |
| P 02 | Weighted graph algorithms: Implementation of problems on network flows | |
| Week # 3: | | |
| L 07 | Greedy Method : Control Abstraction; Elements of Greedy Strategies | |
| L 08 | Greedy Method : Huffman coding | |
| L 09 | Greedy Method : Proof of correctness and optimality of greedy algorithm | |
| P 03 | | |
| Week # 4: | | |
| L 10 | Greedy Method : Proof of correctness and optimality of greedy algorithm | |
| L 11 | Backtracking: Control Abstraction; Search pruning | |
| L 12 | Backtracking: Sudoku | |
| P 04 | | |
| Week # 5: | | |
| L 13 | Backtracking: Sudoku (contd.) | |
| L 14 | Backtracking: N-Queen | |
| L 15 | Backtracking: Sum-of-subsets | |
| P 05 | | |
| Week # 6: | | |
| L 16 | Backtracking: Heuristic search methods | |
| L 17 | Dynamic Programming: Control Abstraction; Comparison with D & C, Greedy and Backtracking | |
| L 18 | Dynamic Programming: Caching vs. Computation | |
| P 06 | | |

| | | |
|---|---|---|
| Week # 7: | | |
| L 19 | Dynamic Programming: Approximate String Matching | |
| L 20 | Dynamic Programming: Longest Increasing Sequence | |
| L 21 | Dynamic Programming: The Partition Problem | |
| P 07 | | |
| Week # 8: | | |
| L 22 | Dynamic Programming: Parsing Context-Free Grammars | |
| L 23 | Limitations of Dynamic Programming | |
| L 24 | Intractable Problems and Approximation Algorithms: Fundamental Concepts | |
| P 08 | | |
| Week # 9: | | |
| L 25 | Intractable Problems and Approximation Algorithms: Problems and Reductions | |
| L 26 | Intractable Problems and Approximation Algorithms: Problems and Reductions (contd.) | |
| L 27 | Intractable Problems and Approximation Algorithms: Reductions for Algorithms | |
| P 09 | | |
| Week # 10: | | |
| L 28 | Intractable Problems and Approximation Algorithms: Elementary Hardness Reductions | |
| L 29 | Intractable Problems and Approximation Algorithms: Elementary Hardness Reductions (contd.) | |
| L 30 | Intractable Problems and Approximation Algorithms: Satisfiability | |
| P 10 | | |
| Week # 11: | | |
| L 31 | Intractable Problems and Approximation Algorithms: Satisfiability (contd.) | |
| L 32 | Intractable Problems and Approximation Algorithms: Creative Reductions | |
| L 33 | Intractable Problems and Approximation Algorithms: The Art of Proving Hardness | |
| P 11 | | |
| Week # 12: | | |
| L 34 | Intractable Problems and Approximation Algorithms: P vs. NP | |
| L 35 | Intractable Problems and Approximation Algorithms: Dealing with NP-complete problems | |

| | | |
|---|---|---|
| L 36 | Intractable Problems and Approximation Algorithms: Dealing with NP-complete problems | |
| P 12 | | |
| Week # 13: | | |
| L 37 | A catalog of Algorithmic Problems | |
| L 38 | Data Structures | |
| L 39 | Data Structures (contd.) | |
| P 13 | | |
| Week # 14: | | |
| L 40 | Numerical Problems | |
| L 41 | Numerical Problems (contd.) | |
| L 42 | Numerical Problems (contd.) | |
| P 14 | | |

## 9. Evaluation scheme (under Grading Pattern-1):

| | |
|---|---|
| Assignments : | 20% |
| Attendance: | 5% |
| Mid-semester: | 15% |
| End-semester(Lab. Test): | 15% |

End-semester(Theory)