Algorithms and Complexity (CE-ALG)

CE-ALG0. History and overview of algorithms and complexity [core]
CE-ALG1. Basic algorithmic analysis [core]
CE-ALG2. Algorithmic strategies [core]
CE-ALG3. Fundamental computing algorithms [core]
CE-ALG4. Distributed algorithms [core]
CE-ALG5. Basic computability theory [elective]
CE-ALG6. The complexity classes P and NP [core]

Deleted: core
Deleted: elective

Algorithms are fundamental to computer engineering. The real-world performance of any software or hardware system depends on two things: (1) the algorithms chosen and (2) the suitability and efficiency of the various layers of implementation. Good algorithm design is therefore crucial for the performance of all systems. Moreover, the study of algorithms provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, computer hardware, or any other implementation aspect.

An important part of computing is the ability to select algorithms appropriate to particular purposes and to apply them, recognizing the possibility that no suitable algorithm may exist. This facility relies on understanding the range of algorithms that address an important set of well-defined problems, recognizing their strengths and weaknesses, and their suitability in particular contexts. Efficiency is a pervasive theme throughout this area.

CE-ALG0. History and overview of algorithms and complexity [core]

Suggested time: 1 hour

Topics:

- Indicate some reasons for studying analysis, complexity, and algorithmic strategies.
- Highlight some people that contributed or influenced the area of algorithms and complexity.
- Mention some basic algorithms and some reasons for their differences.
- Highlight how the use of theory influences algorithms and complexity.
- Indicate how algorithms are part of many different computer applications.
- Provide some knowledge themes such as relating complexity with algorithms.
- Contrast complexities of different algorithmic strategies.
- Explore some additional resources associated with algorithms and complexity.
- Explain the purpose and role of algorithms and complexity in computer engineering.

Learning objectives:

- Identify some contributors to algorithms and complexity and relate their achievements to the knowledge area.
- Associate some of the themes involved with algorithms and complexity.
- Name some applications where algorithms are important.
- Relate contributors with their achievements to the subject.

- Describe how computer engineering uses or benefits from algorithms and complexity.

CE-ALG1. Basic algorithmic analysis [core]

Suggested time: 4 hours

Topics:

Asymptotic analysis of upper and average complexity bounds
Identifying differences among best, average, and worst case behaviors
Big "O," little "o," omega, and theta notation
Empirical measurements of performance
Time and space tradeoffs in algorithms
Using recurrence relations to analyze recursive algorithms

Learning objectives:

1. Use big O, omega, and theta notation to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms.
2. Determine the time complexity of simple algorithms.
3. Deduce the recurrence relations that describe the time complexity of recursively defined algorithms, and solve simple recurrence relations.

CE-ALG2. Algorithmic strategies [core]

Suggested time: 10 hours

Deleted: 8

Topics:

Brute-force/Exhaustive Search algorithms
Greedy algorithms
Divide-and-conquer
At least one of
- Backtracking
- Branch-and-bound
- Heuristics

Covered elsewhere

Deleted: Pattern matching and string/text algorithms

Covered in ALG3 Simple Numberical algorithms.

Deleted: Numerical approximation algorithms

Learning objectives:

1. Design algorithms using the brute-force, greedy, and divide-and-conquer strategies.
2. Design algorithms using at least one other algorithmic strategy from the list of topics for this unit.

CE-ALG3. Fundamental computing algorithms [core]

Suggested time: 12 hours short on hours

Topics:

Simple numerical algorithms
Sequential and binary search algorithms
sorting algorithms for example: ????
Hash tables,Covered in PRF including collision-avoidance strategies
Binary search trees Covered in PRF
Representations of graphs (adjacency list, adjacency matrix)Covered in PRF and DSC
Depth- and breadth-first traversalsCOvered in DSC
Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
Transitive closure (Floyd's algorithm)
Minimum spanning tree (Prim's and Kruskal's algorithms)
Topological sort

Deleted: Quadratic

Deleted: (selection, insertion)¶ O(N log N) sorting algorithms (Quicksort, heapsort, mergesort)

Learning objectives:

1. Use and implement the fundamental abstract data types—specifically including hash tables, binary search trees, and graphs—necessary to solve algorithmic problems efficiently.
2. Solve problems using an efficient sorting algorithms, and fundamental graph algorithms, including depth-first and breadth-first search, single-source and all-pairs shortest paths, transitive closure, topological sort, and at least one minimum spanning tree algorithm. Long sentence split this up into more than one objective.
3. Demonstrate the following abilities: to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in simple programming contexts.

Deleted: sequential search, binary search, O(N log N)

CE-ALG4. Distributed algorithms [core] Duplicated elsewhere OPS?

Suggested time: 3 hours

Topics:

Concurrency
Scheduling
Fault tolerance Not a distributed algorithm

Learning objectives:

1. Explain the distributed paradigm.
2. Distinguish between logical and physical clocks.
3. Describe the relative ordering of events.
4. Explain one simple distributed algorithm.

CE-ALG5. Basic computability theory [elective]

Deleted: core

Suggested time: 6 hours

Topics:

Deterministic finite Automata (DFA)
Non-deterministic finite Automata (NFA)
Equivalence of DFA's and NFA's
Context-free grammars Compiler stuff.
Pushdown automata (PDA)

Deleted: Finite-state machines¶

Deleted: Tractable and intractable problems¶
Uncomputable functions¶
The halting problem¶
Implications of uncomputability¶

Learning objectives:

1. Explain the idea that some problems may have no algorithmic solution.
2. Provide examples that illustrate the implications of uncomputability.

CE-ALG6. Basic Decidability and complexity [Core]

Deleted: The
Deleted: classes P and NP
Deleted: elective
Deleted: 6

Suggested time: 9 hours

Topics:

Tractable and intractable problems
Definition of the classes P and NP
NP-completeness (Cook's theorem)
Standard NP-complete problems

Uncomputable functions
The halting problem
Implications of uncomputability

Deleted: Reduction techniques

Learning objectives:

1. Define the classes P and NP.
2. Explain the significance of NP-completeness.
3. Prove that a problem is NP-complete by reducing a classic known NP-complete problem to it.