Common part

1. **Polynomial algorithms for standard graph problems. Combinatorial and number-theoretical algorithms, isomorphism, prime numbers. Search trees and their use. Text search based on finite automata.** BE4M33PAL (*Course web pages*)

- Notation of asymptotic complexity of algorithms. Basic notation of graph problems - degree, path, circuit, cycle. Graph representations by adjacency, distance, Laplacian and incidence matrices. Adjacency list representation.
- Algorithms for minimum spanning tree (Prim-Jarník, Kruskal, Borůvka), strongly connected components (Kosaraju-Sharir, Tarjan), Euler trail. Union-find problem. Graph isomorphism, tree isomorphism.
- Generation and enumeration of combinatorial objects - subsets, k-element subsets, permutations. Gray codes. Prime numbers, sieve of Eratosthenes. Pseudorandom numbers properties. Linear congruential generator.
- Search trees - data structures, operations, and their complexities. Binary tree, AVL tree, red-black tree (RB-tree), B-tree and B+ tree, splay tree, k-d tree. Nearest neighbor searching in k-d trees. Skip list.
- Finite automata, regular expressions, operations over regular languages. Bit representation of nondeterministic finite automata. Text search algorithms - exact pattern matching, approximate pattern matching (Hamming and Levenshtein distance), dictionary automata.

2. **Problem/language complexity classes with respect to the time complexity of their solution and memory complexity including undecidable problems/languages.** BE4M01TAL (*Course web pages*)

- Asymptotic growth of functions, time and space complexity of algorithms. Correctness of algorithms - variant and invariant.
- Deterministic Turing machines, multitape Turing machines, and Nondeterministic Turing machines.
- Decision problems and languages. Complexity classes P, NP, co-NP. Reduction and polynomial reduction, class NPC. Cook theorem. Heuristics and approximate algorithms for solving NP complete problems.
- Classes based on space complexity: PSPACE and NPSPACE. Savitch Theorem.
- Randomized algorithms. Randomized Turing machines. Classes based on randomization: RP, ZPP, co-RP.
- Decidability and undecidability. Recursive and recursively enumerable languages. Diagonal language. Universal language and Universal Turing machine.

3. **Combinatorial optimization problems - formulation, complexity analysis, algorithms and example applications.** BE4M35KO (*Course web pages*)

- Integer Linear Programming. Shortest paths problem and traveling salesman problem ILP formulations. Branch and Bound algorithm. Problem formulations using ILP. Special ILP problems solvable in polynomial time.
- Shortest paths problem. Dijkstra, Bellman-Ford, and Floyd–Warshall algorithms. Shortest paths in directed acyclic graphs. Problem formulations using shortest paths.
- Network flows. Maximum flow and minimum cut problems. Ford-Fulkerson algorithm. Feasible flow with balances. Minimum cost flow and cycle-canceling algorithm. Problem formulations using network flows. Maximum cardinality matching.
- Knapsack problem. Approximation algorithm, dynamic programming approach, approximation scheme.

- Traveling salesman problem. Double-tree algorithm and Christofides algorithm for the metric problem. Local search k-OPT.
- Scheduling - problem description and notation. One resource - Bratley algorithm, Horn algorithm. Parallel identical resources - list scheduling, dynamic programming. Project scheduling with temporal constraints - relative order and time-indexed ILP formulations.
- Constraint Satisfaction Problem. AC3 algorithm.

Software Engineering

1. **The methodology of software testing. Methods for test creation from the application model. Automated testing.** BE4M36ZKS (*Course web pages*)

- Describe and compare the V and W models of the software testing process. Explain static testing and its role in the W model. Describe individual methods of static testing.
- Explain the principle of Model Based Testing (MBT) and compare its advantages and disadvantages with manual testing approach. Give some examples of models that can be employed in MBT. How does MBT relate to test automation?
- Outline the main test automation principle and economics. What are possible levels at which tests can be automated? Give some examples of main approaches and technologies that can be used in software test automation.
- Explain the equivalence class and boundary values concepts and principle of the Combinatorial Interaction Testing. What is a combinatorial explosion effect, how to effectively reduce the input data combinations? Principle of pairwise (2-way) and N-way testing.
- Principle of path-based testing. Formal definition of system model and test coverage criteria (node/edge coverage, edge-pair coverage, prime-path coverage). How prioritization of process/workflow activities is modelled and handled in the generation of the test cases?

2. **Software architectures, their parameters and qualitative metrics. Architectural patterns, styles and standards.** BE4M36SWA (*Course web pages*)

- Describe Krutchen's 4+1 View Model of a software architecture. Explain how it captures the complete behavior of a developed software from multiple perspectives of the system. How is this model aligned with the UML models?
- What is a software architecture? Describe the importance of software architecture when developing a system. What are the software architecture design guidelines? What are the architectural styles? Give an example of an architectural style and describe it in detail.
- What is a design pattern? What problem are design patterns solving? What types of design patterns exist? Why is it important to know the design patterns? Are there design antipatterns?
- What is a microservice architecture? What are its advantages and disadvantages compared to a monolithic architecture. How is microservices development different from developing a monolithic application? Are there any methodologies or guidelines or best practices to follow when developing microservices? Obviously, there are patterns that can be applied in this area, are there any antipatterns that should be avoided?
- Software architects can choose one architecture over another. The choice may affect the quality of the final product. Can you tell if one architecture is better than another or if one architecture is bad while the other is not? How can you measure the quality of an architecture? Can you measure the quality from different perspectives?

3. **Properties of parallel and distributed algorithms. Communication operations for parallel algorithms. Parallel algorithms for linear algebra.** BE4M35PAG (*Course web pages*)

- Describe basic communication operations used in parallel algorithms. Show cost analysis of one-to-all broadcast, all-to-all-broadcast, scatter, and all-to-all personalized communication on a ring, mesh, and hypercube. Describe All-Reduce and Prefix-Sum operations and outline their usage.
- Describe performance metrics and scalability for parallel systems. How efficiency of a parallel algorithm depends on the problem size and the number of processors? Derive isoefficiency functions of a parallel algorithm for adding numbers (including communication between processors) and explain how it characterizes the algorithm.

- Explain and compare two parallel algorithms for matrix-vector multiplication. Describe a parallel algorithm for matrix-matrix multiplication and explain the idea of Cannon's algorithm. Discuss the principle and properties of the DNS algorithm used for matrix-matrix multiplication.
- Outline the principle of sorting networks and describe parallel bitonic sort, including its scalability. Explain parallel enumeration sort algorithm on PRAM model, including its scalability.
- Explain all steps of a parallel algorithm for finding connected components in a graph given by the adjacency matrix. Using an example, illustrate a parallel algorithm for finding a maximal independent set in a sparse graph.

4. **Effective algorithms and optimization methods. Data structures, synchronization and multithreaded programs.** BE4M36ESW (*Course web pages*)

- Java Virtual Machine, memory layout, frame, stack-oriented machine processing, ordinary object pointer, compressed ordinary object pointer. JVM bytecode, Just-in-time compiler, tired compilation, on-stack replacement, disassembler, decompiler. Global and local safe point, time to safe point. Automatic memory Management, generational hypothesis, garbage collectors. CPU and memory profiling, sampling and tracing approach, warm-up phase.
- Data races, CPU pipelining and superscalar architecture, memory barrier, volatile variable. Synchronization - thin, fat and biased locking, reentrant locks. Atomic operations based on compare-and-set instructions, atomic field updaters. Non-blocking algorithms, wait free algorithms, non-blocking stack (LIFO).
- Static and dynamic memory analysis, shallow and retained size, memory leak. Data Structures, Java primitives and objects, auto-boxing and unboxing, memory efficiency of complex data structures. Collection for performance, type specific collections, open addressing hashing, collision resolution schemes. Bloom filters, complexity, false positives, bloom filter extensions. Reference types - weak, soft, phantom.
- JVM object allocation, thread-local allocation buffers, object escape analysis, data locality, non-uniform memory allocation.
- Networking, OSI model, C10K problem. Blocking and non-blocking input/output, threading server, event-driven server. Event-based input/output approaches. Native buffers in JVM, channels and selectors.
- Synchronization in multi-threaded programs (atomic operations, mutex, semaphore, rw-lock, spinlock, RCU). When to use which mechanism? Performance bottlenecks of the mentioned mechanisms. Synchronization in "read-mostly workloads", advantages and disadvantages of different synchronization mechanisms.
- Cache-efficient data structures and algorithms (e.g., matrix multiplication). Principles of cache memories, different kinds of cache misses. Self-evicting code, false sharing – what is it and how deal with it?
- Profiling and optimizations of programs in compiled languages (e.g., C/C++). Hardware performance counters, profile-guided optimization. Basics of C/C++ compilers, AST, intermediate representation, high-level and low-level optimization passes.

5. **Big Data concept, basic principles of distributed data processing, types and properties of NoSQL databases.** BE4M36DS2 (*Course web pages*)

- Big Data (V characteristics, current trends), NoSQL databases (motivation, features). Scaling (vertical, horizontal, network fallacies, cluster). Distribution models (sharding, replication, master-slave and peer-to-peer architectures). CAP theorem (properties, ACID, BASE). Consistency (strong, eventual, read, write, quora). Performance tuning (Amdahl's law, Little's law, message cost model). Polyglot persistence.
- MapReduce (architecture, functions, data flow, execution, use cases). Hadoop (MapReduce, HDFS).

- XPath (path expressions, axes, node tests, predicates). XQuery (constructors, FLWOR, conditional, quantified and comparison expressions). SPARQL (subgraph matching, graph patterns, datasets, filters, solution modifiers, query forms).
- RiakKV (CRUD operations, links, link walking, convergent replicated data types, Search 2.0, vector clocks, Riak Ring, replica placement strategy). Redis (data types, operations, TTL). Cassandra (keyspaces, column families, CRUD operations). MongoDB (CRUD operations, update and query operators, projection, modifiers).
- Graph data structures (adjacency matrix, adjacency list, incidence matrix). Data locality (BFS layout, bandwidth minimization problem, Cuthill-McKee algorithm). Graph partitioning (1D partitioning, 2D partitioning). Neo4j (traversal framework, traversal description, traverser). Cypher (graph matching, read, write and general clauses).

6. **Security analysis of operating systems, development of secure software and web applications security. Analysis of cyberattacks and malware. Security of mobile devices.**

   BE4M36BSY (*Course web pages*)

- Side channel attacks, Basics of Steganography.
- Discretionary access control (Access control list, Capabilities).
- Detect intruders in operating systems. Hardening operating systems.
- Privilege escalation.
- Virtualization.
- Network protocols, TCP, DNS, TSL, mechanism of certificates, security of protocols.
- Firewalls, network intrusion detection, network intrusion prevention, intrusion deflection.
- Web attacks, SQL injection.
- Denial of service attacks, binary exploitation.