UL FRI – Algorithms and data structures 1 – VSŠ - contents

1. Algorithms and problems
   1. Concepts of algorithm, instance, solution and problem
   2. Problem kinds: decision, search, counting, enumeration, optimization
   3. Algorithm description: natural language, flow diagram, pseudocode, programming language, …
   4. Implementation, algorithm trace, design methods in general
   5. Correctness of algorithms: intuitive understanding, testing
   6. Formal proof of correctness: loop invariant and induction
2. Complexity of algorithms
   1. Time and space complexity, model of computation, RAM and other models
   2. Complexity depending on instance size and data (best, worst, average)
   3. Exact complexity is cumbersome: asymptotic complexity
   4. Asymptotic notation: O, Ω, Θ – upper, lower and tight bound
   5. *Bonus: o, ω, tilde notation
   6. Using limits for asymptotic complexity, complexity classes, using asymp. notation
3. Abstract data types
   1. Data type and abstract data type
   2. Set, bag, stack, queue, deque, priority queue, sequence, dictionary
4. Array
   1. Capacity, size, efficiency, static and dynamic arrays
   2. Array as a stack, queue, deque, sequence, set, bag
   3. Dynamic arrays
   4. Amortized complexity of add/remove/resize operations
5. Linked lists and pointers
   1. Singly linked list, operations, stack and queue
   2. Doubly linked list, cyclic lists, guarded lists (dummy element)
   3. Representing linked list in array
   4. Persistent stack with linked lists
   5. Implicit and explicit data structures
6. Rooted trees
   1. Root, vertex, edge, internal edge, leaf, parent, child, ancestor, descendant
   2. Path, subtree, forest, ordered tree, depth/height/degree of vertex/tree
   3. Binary, ternary, and d-arry trees
   4. Full, perfect and complete trees
   5. Basic algorithms: counting vertices, leafs, internal nodes, …
   6. Tree traversals: pre/post/in/level -order of a tree
   7. Representing trees with pointers
   8. Representing trees with arrays
7. Priority queues and heaps
   1. Basic implementations with array, sorted arrays, etc.
   2. Heap: definition, min, max, properties
   3. Operations: siftUp, enqueue, siftDown, dequeue
   4. Construction heap: a) insertion and siftUp, b) siftDown
   5. Other operations: max, 2. max, find, increase/decrease key, ...
8. Sorting arrays (using comparisons)
   1. Kind of data and sorting problem, stability
   2. Straight sorting: straight insertion, straight, selection, straight exchange
   3. Advanced sorting: heap sort, merge sort, quicksort
   4. Complexity of the sorting problem: lower bound
   5. Algorithm engineering: TimSort, Jaroslavski, 5-pivot QS

9. Sorting arrays (without comparisons)
    1. Counting sort
    2. Radix sort
    3. Bucket sort
10. Order statistics and k-th smallest element
    1. Problem and special cases
    2. Finding min and max with 3/2n comparisons
    3. QuickSelect
    4. Median of medians
11. Graphs
    1. Undirected graph: vertex, edge, label, weight, adjacency, incidence, degree
    2. Directed graph (digraf): in- and out- degreed of vertices
    3. Representing graphs and digraphs: adjacency lists, adjacency matrix, distance matrix, incidence matrix
12. Algorithms on graphs
    1. Walks, trails, paths and cycles
    2. Algebraic algorithms and adjacency matrix: counting walks, reachability, counting triangles
    3. Graph traversals: DFS (enter and exit order) and BFS (order)
    4. Using dfs / bfs: reachability, shortest paths, ...
    5. Topological sort: problem, two algorithms, cycle detection
    6. Graph connectedness: undirected graph, digraph (weak and strong connectedness and components)
    7. Kosaraju's and Tarjan's algorithms for strongly connected components
13. Brute force and exhaustive enumeration
    1. Brute force: examples, substring search (naive, Rabin's algorithm)
    2. Exhaustive search: idea, generating permutations
14. Backtracking, branch and bound
    1. Backtracking: decision tree, maze solving, chess queens
    2. Vertex cover: selecting vertices, selecting edges
    3. 0/1 knapsack: backtracking with pruning, branch and bound
15. Divide and conquer
    1. Principle and recursive equations: substitution
    2. Master theorem and its proof
    3. Examples of analysis of known algorithms
    4. Big integer multiplication (D&C, Karatsuba's algorithm)
    5. Matrix multiplication (for-for-for, D&C, Strassen's algorithm), state-of-the-art theory
16. Greedy algorithms
    1. Idea of the greedy method and money changing problem
    2. Storing files on tape, multi-tape problems
    3. Simple knapsack problem