CPS 305 Data Structures, Fall 2018

Daytime CPS 305: Ryerson University, School of Computer Science
Evening CCPS 305: Ryerson University, Chang School of Continuing Education

Instructors

Lecturers:

- Ilkka Kokkarinen (daytime and evening), `ikokkari@ryerson.ca`, ENG 241 (hidden in the little stub hallway right off the east hallway), office hours Thursdays 2-3:30
- Alex Ufkes (daytime only), `aufkes@ryerson.ca`, ENG 209, office hours Monday 12-2 or by appointment

Lab TA's (daytime course only):

- Saeideh Gholamrezazadeh Motlagh (Monday 9-10 ENG205, Monday 10-11 ENG205, Monday 3-4 ENG202), `sgholamrezazadeh@ryerson.ca`

Material

The official, and by far the most convenient, course material are the Wikipedia pages on the course topics, as linked in the outline below. All final exam questions can be answered based on the information given in the linked Wikipedia pages. The friendly Wikipedia elves have compiled most of these into a virtual textbook "Fundamental Data Structures", or the expanded version "Data Structures", for your easier download and perusal.

For students who are looking for a more traditional style textbook, any classic introduction to data structures and algorithms originally hailing from the nineties such as "Introduction to Algorithms", "Algorithms: 4th Edition" or "The Algorithm Design Manual" would do just fine, should you find an

online version or a second-hand dead tree copy in some bargain bin as the last step before being pulped. For a slightly more playful approach, some students might want to check out "Competitive Programmer's Handbook" freely available online. Motivated students, especially those who plan to take CPS 616 after this course, might also want to check out equally free "Algorithms, Etc."

Most of the code examples used to demonstrate the theory taught in the lectures are taken from the page "Java Algorithms and Clients", aided by some additional Java examples created by the instructors. Some algorithms are easiest to explain visually using the interactive "Data Structure Visualizations". That Medium article that had links to 500 good algorithmic problems has since disappeared into the void, but to replace it, "Geeks for Geeks Fundamental Algorithms" should be more than enough.

To spice up the lectures, the instructors will also occasionally use some sets of slides of the introductory data structures and algorithms courses offered by some of the finest universities that we strive to emulate. Some of these are:

- Lecture slides for Algorithms, 4th Edition
- Algorithm Design
- MIT OCW 6.006 Lecture Notes (Spring 2008)
- Radboud OCW Algorithms and Data Structures Slides
- University of Washington CSE 373 (Autumn 2002)
- Stanford CS 161
- Duke CPS 130
- University of British Columbia CPSC 490

Grading

The course grade is calculated as the sum of three components:

- Final exam 50%
- Three Java programming projects 10% each
- Weekly labs 20%

The weekly labs are done in a classroom and consist of TA-led theoretical "whiteboard" discussion of the most elegant solutions of that week's problems about the data structures and algorithms of the previous week's lecture. The weekly lab problems are given on a separate page. **Labs will start Monday September 17**.

In the daytime course, the lab marks are given for full attendance and participation of at least giving the appearance of caring about what is going on, by discretion of the lab TA. **The student must attend the entire lab to receive any marks.** There will be no jockeying between the daytime lab sessions, but you must attend the lab session of your assigned section. (We know that you don't have any other Ryerson classes taking place at that time.) If you miss a lab by valid documented reason as allowed by the rules of Ryerson University, we will decide how you can make up for it.

In the Chang School course that meets one evening per week without a separate lab session, the labs are integrated to the lectures.

The programming projects are each worth 10% of the total course grade. The grading of each project is **competitive over all student submissions of that particular project**, as explained on the project pages. There are four programming projects in total, but **only the marks for your three highest scoring projects will count towards your course grade**. Therefore, if you do well enough to your satisfaction in your first three projects, you don't even need to do the fourth project, unless you feel like getting some frisk additional exercise. On the other hand, should you mess up your first or second project, there is still a chance to redeem yourself with good marks for the last two projects.

Before starting to code any project, read very carefully its ENTIRE specification page! And then do exactly what the specification tells you to do, leaving the room for improvisation and your freedom to dance in how you act within the boundaries set up by the specification.

Project	**Day school deadline**	**Chang School deadline**
Union of Intervals	**Noon** Sunday Oct 7 (**CHANGED**)	**Noon** Sunday Oct 14 (**CHANGED**)

Linked Lists from Scratch	**Noon** Sunday Nov 4 (**CHANGED**)	**Noon** Sunday Nov 11 (**CHANGED**)
Packing Words in Bins	**Noon** Sunday Dec 2	**Noon** Sunday Dec 9
Word Filling	**Noon** Sunday Dec 2	**Noon** Sunday Dec 9

The project deadlines are firm. No extensions will be given without valid documented medical or other reason that is acceptable per the rules of Ryerson University. Please do not leave your coding and submission to the last day and the last moment.

Topic Outline

Week One: Searching Inside an Array

Topics: Data structures and algorithms. Linear search, sentinel search. Binary search and interpolation search. Some classic micro-optimizations of loops. Loop invariants in program analysis. Subarray searching ("string matching") algorithms as a sneak peek of many important algorithmics topics such as state machines, hashing, checksums, amortized analysis and one-pass algorithms.

Material: Algorithm, Data structure, Binary search, Interpolation search, Loop invariant, String matching, Boyer-Moore, Rabin-Karp, Knuth-Morris-Pratt, Amortized analysis, Accounting method, In-place algorithm.

Java examples: ArraySearchDemo.java BinarySearch.java BoyerMoore.java RabinKarp.java KMP.java (this last example is actually the $O(n + \Sigma m)$ time DFA matcher, instead of the genuine $O(n)$ time KMP matcher)

Week Two: Pushing Quadratic Time down to Linear

Topics: Pushing down the asymptotic running time for some simple array algorithms. The tragically ludicrous and ludicrously tragic tale of "Poor Shlemiel". Good techniques to avoid being a Shlemiel yourself. Trading memory for time. Efficient evaluation of polynomials. Priority queues implemented with binary heaps. Practical uses of priority queues in some other algorithms.

Heapsort.

Material: One-pass algorithm, Space-time tradeoff, Optimization, Fisher-Yates Shuffle, Reservoir sampling, Horner's method, Priority queue, Binary heap, Heapsort.

Java examples: Shlemiel.java Knuth.java Heap.java MaxPQ.java BigHamming.java

Week Three: Linked Data Structures

Topics: Linked lists. Subtypes of linked lists. Singly and doubly linked lists. Free list, block allocation. Abstract data structures queue, deque and stack, along with their efficient implementations with lists and arrays. Hierarchical and nested lists. Linked representation of incomplete binary tree structures. General tree representation with arbitrary branching.

Material: Queue, Stack, Deque, Circular buffer, Linked list, Sentinel node, Free list, Memory pool, Cons, Memory hierarchy, Locality of reference, Left-child right-sibling binary tree.

Java examples: Stack.java LinkedStack.java ResizingArrayStack.java Queue.java

Week Four: Comparison Sorting

Topics: The power of a single two-way comparison. Theoretical lower bounds for sorting. What sorting can teach us about algorithms in general. The inefficient selection sort, perhaps better renamed in this course as "Shlemiel sort". Insertion sort and its $O(n^2)$ worst case running time resulting from the array element inversions. The inherent $O(n \log n)$ lower bound for comparison sorts. Mergesort. Quicksort and its myriad little intricacies and optimizations. Tail calls and why those are a good thing. Efficiently computing the median and arbitrary order statistics.

Material: Sorting algorithm, Selection sort, Insertion sort, Mergesort, Quicksort, Quickselect, Median of medians, Tail call.

Java examples: Selection.java Insertion.java Merge.java Quick.java QuickBentleyMcIlroy.java

Week Five: Unconventional Sorting

Topics: The non-comparison based linear time sorting algorithms for integers, floating point numbers and strings. Tries and trie sorting of strings. Some general ideas and variations of tries.

Material: Counting sort, Bucket sort, Radix sort, Trie, Suffix tree, Radix tree.

Java examples: LSD.java MSD.java TrieSET.java

Week Six: Hashing

Topics: Dynamic sets and maps. Hash tables. Hash functions. Collision resolution with chaining and open addressing. Probing schemes. Chaining. Other collision resolution schemes. Bloom filters. Cuckoo hashing. Checksums and other uses for hash functions.

Material: Set (abstract data type), Hash table, Hash function, Pigeonhole principle, Birthday problem, Open addressing, Bit array, Bloom filter, Cuckoo hashing, List of hash functions (no need to memorize any of these, these are just for reference).

Java examples: SeparateChainingHashST.java LinearProbingHashST.java BloomFilter.java

Week Seven: Taming the Branching Recursions

Topics: Divide and conquer approach to problem solving. Problems that have an optimal substructure. Converting the optimal substructure into recursion. Taming the exponentially repeated recursive subproblems with memoization or dynamic programming. Dynamic programming examples. Problems that have the greedy choice property.

Material: Divide and conquer, Optimal substructure, Travelling salesman problem, Memoization, Dynamic programming, Greedy algorithm.

Examples: DynProg.java

Week Eight: Binary Search Trees

Topics: Binary search trees and their basic dynamic set and iterator operations. Recursive preorder, inorder and postorder tree walks. Tree rotations to rebalance the BST structure. The ideas behind the classic balanced BST schemes of AVL-tree, red-black tree (as isomorphism of 2-3-4 trees), and splay tree.

Material: Binary tree, Binary search tree, Tree rotation, 2-3-4 Tree, AVL tree, Red-black tree, Splay tree.

Java examples: BST.java RedBlackBST.java

Week Nine: Graph Traversal

Topics: Graphs in computer science. Directed and undirected graphs. Explicit versus implicit representations of graphs. Adjacency list versus adjacency matrix explicit representations. Implicit state spaces of puzzles and other discrete systems. Classic graph searching and shortest path

algorithms breadth-first search, depth-first search and Dijkstra's algorithm. The heuristic search algorithm A*.

Material: Graph, Adjacency list, Adjacency matrix, BFS, DFS, Strongly connected components, Tarjan's algorithm, Depth-limited search, Dijkstra's algorithm, A*.

Java examples: BFS.java IterativeDFS.java AStar.java SearchMain.java TarjanSCC.java

Week Ten: Backtracking

Topics: Constraint satisfaction problems. Eight queens puzzle, graph colouring, Sudoku and other famous CSP's. The recursive backtracking algorithm. Iterative generation of all solutions using internal state tables instead of recursion. Forward checking and other search tree pruning techniques to help the search realize that it has painted itself into a corner. Heuristics for choosing the next unassigned variable to be processed. Propositional logic satisfiability problem. Converting other computational problems to SAT.

Material: Constraint satisfaction problem, Eight queens puzzle, Backtracking, Dancing links, Look-ahead.

Java examples: NQueens.java Permutations.java DominatingSet.java SATSolver.java SudokuLogic.java

Week Eleven: Graph Structural Analysis

Topics: Minimum spanning tree. Greedy algorithms of Kruskal and Prim to compute the minimum spanning tree. Topological sorting of a DAG. Techniques of random maze generation. All-pairs shortest paths with Floyd-Warshall dynamic programming algorithm.

Material: Greedy algorithm, Kruskal's algorithm, Prim's algorithm, Maze generation (interested students can check out Jamis Buck's compilation as extra material), Topological sorting, Floyd-Warshall algorithm.

Java examples: KruskalMST.java PrimMST.java Topological.java

Week Twelve: Computational Geometry

Topics: Generalizations of binary search trees for two- and higher-dimensional spaces. Computational geometry. Vector cross product on the plane with its surprisingly powerful applications. Line segment algorithms of computational geometry. Polygons and their basic intersection and containment algorithms. Convex hull computations, applications of convex hull.

Comparison of sweepline and recursive algorithms for sets of points and other geometric objects.

Material: Quadtree, Binary space partitioning, Polygon, Cross product, Computational geometry, Convex hull algorithms, Closest pair of points, Point in polygon, Sweepline algorithm, Point location, Clipping (follow the links to the individual line and polygon clipping algorithms), Dot product, Voronoi diagram, Delaunay triangulation.

Java examples: GrahamScan.java