~~~ last update: May 2023 ~~~

TABLE OF CONTENTS
~~~

ABOUT THIS TRACK

Computer Science (CS) skills should go far beyond just programming and syntax. It should focus on problem-solving ability with computational thinking (CT), even for grade schools. CS with CT skills is indispensable to prepare our generation for the world of artificial intelligence.

Storming Robots utilizes Robotics to animate problem-solving efforts starting in Grade 5. However, we encourage students to study in this Algorithms in C/C++ Track starting no later than Grade 8. This syllabus consists of four levels. The levels will be comparable up to college level from Freshman to junior year topics in CS, such as data structure, introduction to algorithms, and complexity analysis.

Characteristics (important to read first)

1) Go far beyond just syntax, but algorithmic thinking (computational thinking) along the way, and how to build high-quality software. At the later levels, students will engage in more sophisticated algorithms.
2) **Focus on problem-solving**/software development skills, so students will not work with the physical robot.
3) **Heavily on analytical skills development, NOT by rote memorization**.
4) **Progress is self-paced**. All assigned exercises should be completed with excellent quality. In the later levels, more emphasis on robustness, memory consumption, readability, maintainability, etc. All levels stress in Computational Thinking and Efficiency.
5) Students will be allowed to move quickly to the next concept after demonstrating satisfactory understanding through their homework and/or on a pop-quiz in class if necessary. Those with gaps in their prerequisite knowledge will receive additional exercises to address the shortcomings.
6) Most exercises/instructions constantly require higher order of thinking. (i.e., no spoon-feeding method).

Levels Aligning with Storming Robots Roadmap

Allow students to embed competitions and standardized exams in-between levels. Therefore, most of the homework assignments provided in the later levels are not from the book but projects adhering to professional quality. The following shows a general summarized idea of expectations.

1) Students who completed level I with high "proficiency" have the capacity to self-study for Advanced Placement Computer Science (AP CS A). The majority of these students score 5 in AP with a small extent of review on the fundamental object-oriented structure.
2) Some Level II students participate in the USA Computing Olympiad (USACO) online exam. Do note that USACO can be very challenging even to the best students, and far more demanding in analytical and computational programming skills than AP CS. One of the common traits among the students with high proficiency in this level is their ability to adapt to other technologies.
3) Level III students participate in most of SR's advanced robotics activities. High proficiency in this level matches the professional quality, and gain a higher competitive edge to Internship available for high school students in the USA – see https://mp.stormingrobots.com.

How to do Well in this Track?

1) **Review the book materials** even after concepts are explained to you in class. Jot down questions and bring them to class — review especially the QandA section.
2) Completing all exercises naturally follows deep learning, but not necessarily vice versa. It takes "**self-disciplines**" and "**stay inquisitive**."
   a) Be self-disciplined — while there are a lot of learning materials online, such as StackOverflow, you should never just copy and paste. Acquire the "thinking process." This goes true as well if you receive external help.
   b) Stay inquisitive— not just know how to implement the solutions, but acquire the analysis process to arrive at the solutions. Do not allow yourself to be satisfied just because a program set seems to be complete, but only after you feel you do understand.
   c) Ask questions— Jot notes while you are reviewing the chapters. Bring questions to class for discussion.
3) **Limit acquiring external assistance** without putting effort in trying to solve problems.
   a) e.g., Receiving substantial external assistance at home to complete their assigned work often will quickly cause you to miss out on the critical process of analysis. This indispensable learning element strengthens your ability to apply in various scenarios.
   b) Stay perseverant in taking on challenging problems
4) Should expect approx. 3+ hours of programming homework per week, with at least an hour each time for homework.
5) Utilize the Debugging tool.

Limit seeking external assistance

While external assistance can be beneficial, it can be detrimental. The ultimate question is whether students have gone thru **the process of analysis** instead of given solutions that students just implemented.

When students were given a particular technique for solving a problem without going thru "the process of analysis" (the most critical part), they become dumbfounded when the problem is presented again in a different fashion.

The external assistance may be from coping from the internet site such as StackOverflow, or assistance from family members, etc.

*** **Missing Bottom line: Indispensable Process of Analysis**. **Part of true learning is to go through the journey of solving a problem by making mistakes**, not just by given the technique/methods to solve a problem using rote memorization.

What was deemed to be "complete" with an assignment?

1) Proper testing is required to ensure correctness.
2) Any error must be corrected.

3) Students must be able to explain their work, not just the syntax but thought processes.
4) Suggested improvement must be made if code is found poorly written and inefficient.
5) All completed source codes must be uploaded onto the student work folder.

Requirement for Level Promotion:

1) MUST demonstrate a satisfactory level of understanding thru their work – see "What was deemed to be complete with an assignment?" above.
2) MUST upload your work onto your student google folder (provided to you from Storming Robots)
3) MUST pass pop-quizzes.

Learning Resources—Books and online packets:

Level I, II :

- C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
- Additional Supplemental (optional) and Storming Robots online learning site — Computer Science Track.

Level III :

1) Part of K.N. King book
2) Class Notes

Level IV:

1) Mastering Algorithms with C: ISBN-13: 978-1565924536, or ISBN-10: 1565924533
2) Class Notes

Level V:

1) C++ Primer Plus (Developer's Library) 6th Edition, by Stephen Prata. ISBN-13: 9780321776402.
2) (Optional) : Data Structures and Algorithm Analysis in C++, 4th Edition by Mark A. Weiss—ISBN-10: 0273769383
3) Class Notes.

Software required

Microsoft Visual Studio Community Version (Windows OS only).

If you use MAC, you will may use one of the followings :

1) an online tool, such as online-GDB IDE.

2) XCode.

3) **Special note:** We shall not be able to provide support for any other IDE other than the Visual Studio Community Version. Besides, the VS Community Version is preferable for its highly versatile debugger, a vital learning tool.

Honor Code

Programming assignments are pledged work and are bound by the honor code. To simply put, the violation may be in any of the following forms:

— Claim someone else's work as their own.

— Edit someone else's work by simply changing variables or style, etc., and claim to be the result of your own work.

— Complete assigned work with substantial help from others to the extent that you cannot even explain your own work.

If found honor code is violated, parents will be informed. Students will not be allowed to participate in any competitions with Storming Robots. Repeating offenders will not be accepted back to SR programs.

Should you aim for all levels?

No, it depends on your interests and strength.

Level I: Anyone who wishes to study any engineering and science major; minimal requirement for entering our Robotics and Electronic.

Level II & III: Gain entrance to more advanced robotics activities thru SR. Most students who wish to engage in more advanced robotics that require more sophisticated algorithms will take this level simultaneously with Robotics at SR.

Level IV: Particularly for future Computer Science majors in college.

Duration for each level?

Self-paced.

⏮ _Level_B-I_: I II III IV

LEVEL B-I : THE FUNDAMENTALS

Learning Outcome

1) Students, who accomplish this level with high proficiency, will find it very easy to self-study Advanced Placement in Computer Science and achieve a high score. A large majority of them reported to us that they all achieved a score of 5 with ease.
2) By the time this level is completed, students should have completed approximately 30 programming exercises up to Chapter 8 from the K.N. King book and basics in struct and enum.
3) Possess knowledge in programming fundamentals with an emphasis on producing clear, robust, and reasonably efficient code using top-down design, followed by informal analysis and effective testing and debugging.
4) Build the habit in analytical thinking, NOT just syntax recall.
5) Build the mindset with efficiency in mind.
6) Experience the importance of proper testing
7) Experience the importance of using Debugger for troubleshooting.
8) Gain experience in using one of the most versatile integrated development environment
9) Basic mechanic understanding in utilizing Array structure.
10) Exploratory usage of functions.
11) For Level B (up to ch.6): students should be able to do simple control structure, take a very straightforward problem set to resolve it with computer programming in C.
12) For Level I (up to ch.8 + basic enum and struct), students should master nested control structure, analyze a problem set, and figure out how to resolve it with computer programming in C.

If you join this class with a prior programming background in RobotC Robotics Projects I & II Analytics, it should help you accelerate through the first five to six chapters.

Learning Resources:

1) Book required: C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
2) Class Notes and online packets.

Covered Concepts:

Level B

1) How to use the Microsoft Visual C/C++ Compiler IDE.
2) C Fundamentals in writing Simple Program and Standard Formatted I/O - Chapter 2, 3
3) Expressions (simple to compound) - Chapter 4
4) Rudimentary level in using void functions() – Class note
5) Selection/ if - Control Statements | Boolean Expressions - Chapter 5
6) Loops Control Structure - Chapter 6

7) Rudimentary level understanding in creating functions. - (partial Ch.9 & 10 with Class Note)
8) Fundamental understanding in using Multiple files within a project - (partial Ch.9 & 10 with Class Note)
9) Base Conversions— Octet, Hexadecimal, Binary. (with Class note)

Level I

1) Chapter 7 (with Class note) (data sizes, range, ascii)
2) 1D and 2D Array – Chapter 8.
3) Fundamental level of understanding in Enum (with Class note)
4) Fundamental level of understanding in struct (with Class note and Partial ch.16)
   a) Anatomy of a struct type
   b) Importance in encapsulation and reusability
5) Boolean Algebra
6) Rudimentary level of Object-oriented Programming Structure (with struct Abstraction).
   a) Anatomy of a Class structure
   b) Overloading / Overriding
   c) Getter/setter functions
   *** Slightly touch on inheritance. Won't touch on polymorphism until later in advanced pointer

All final work should follow prescribed Programming Style. There will be pop quiz given at random time to some students in order to ensure satisfactory proficiency in the covered concepts.

To conclude Level I:

Students must demonstrate satisfactory ability in both mechanics and analysis portion of all content listed above.

What's next:

Advanced Placement in Computer Science:
While a large majority of our CS students reported to us that they all achieved a score of 5 with great ease. you are recommended to read up on fundamentals in object-oriented principles in design pattern. (Using the AP Barron AP CS A book will be good.) Or, take an AP CS at SR during the summer – SR's AP CS class will also cover more data structure and problems analysis beyond the bare mechanics covered in AP CS – A.

Interested in Robotics Electives, i.e. competitions (either Hardware platform or Simulation) ? Must possess proficiency by passing our Level I Test. May view the programs and competitions criteria online.

Continue on Algorithms in C/C++ - Level II. Many do both Algorithms in C/C++ along with Robotics with Electronics.

⏮_Level_B-I_: I II III IV V

LEVEL II : DATA ABSTRACTION TYPE (ADT) - LINEAR STRUCTURE

This level involves software development skill well beyond Advanced Placement CS-A. Students should expect to work lesser number of exercises, but more demanding than Level I.

Learning Outcome:

1) Should be able to tackle *Bronze/Silver in the USACO*. Based on our history, all students who demonstrate high proficiency in completing Level II and possess a high chance to be promoted to the Silver Level. Some even got Gold – see our students' achievements here.
    - Do note that the target is not just about participating in USACO. Achieving Silver in USACO is just a by-product of excellent preparation and building a dynamic knowledge base to prepare our students to participate in more challenging opportunities, such as competitions, summer engineering, and research internships, etc.
2) Strengthen programming analysis. Most programs exercise you have completed in even Level II under this program require higher analytical skills over what is required in AP Comp. Sci.
3) Become more efficient in Debugger, including breakpoints, observation of variables, watch feature.
4) Ability to recognize patterns, pay attention to reduction for efficiency, and algorithmic thinking instead of brute-force.
5) Building mindset of computational thinking.
6) Should be able to conduct a more complex project than typical College upper-year level in CS.
7) Should feel comfortable to navigate around Linux System, and be able to build programs and libraries yourself with gcc and g++ and makefile
8) Gain a better understanding of the compilation and link process and the complexity of building larger projects consisting of multiple folders and files.
9) (tentative) Get experience in versioning control system—Git.

Learning Resources:

1) Book required: Same book for previous level. C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
2) Class Notes and online packets.

Covered Concepts:

1) Review of the number base system, solid understanding on data types and their ranges, etc... (MUST pass the LEVEL-II-beginner Test)
2) Explore the concept about prefix/infix/postfix (for complex Math Expression) . (class notes)
3) More advanced analytical work on Arrays (class notes)
    a) -Heavily focus on robust implementation and error checking
    b) -More focus on Multi-Dimensional
    c) -Additional analytical work, exploring the importance of unique key in a lookup table.

4) Memory pointers with 1D and 2D array – Chapter-11 & 12
   a) -Array of pointers to function
5) String manipulation — Chapter-13
   a) -C string functions with a focus on pointers arithmetic.
   b) Array of pointers vs. static 2D array,
   c) How to utilize the memory addresses shown in the debugger
   d) Command-line arguments
   e) Dynamic allocation — Chapter 17-1 to 17-4
      i) malloc/free, calloc/free, vs. realloc
      ii) Utilize memxxx() functions
   f) Learn how to utilize C++ String for productivity purpose
6) Preprocessor - Chapter-14 —Conditional Compilation, such as #if, #elif, #line, ##, etc. Its role in compilation process.
7) Bitwise operations — Chapter 20 . and BitString flicking concepts
8) Recursions (with class note)
   a) Explore stack data structure — more in-depth understanding of how it works in memory stack in the later level.
   b) Basics in dynamic programming method.
9) Commonly used Search / Sort algorithms (both in iterative and recursive) :
   a) Basic in Binary Search
   b) Insertion sort (implementation)
   c) Qsort with integers list (using the API)
10) Error Handling – Ch.24.1 to 24.2. and classnotes
11) Basic File I/O — Chapter 22 (with C & C++ - add'l note)
   a) Try out a few USACO problems and submission.

To conclude Level II:

— This level involves more mini-projects and exercises external to the book.

— Summary of the chapters covered in this level (not necessary in this order): Chapter 9 to 16, and 20, 22.

— Reference : Standard Library—Chapter 21, 24

What's next:

— Should continue working on the USACO online practices. See Program criteria for more suggestions.

— Continue on Algorithms in C/C++ - III.

— And/or participate in more advanced robotics competition.

⏮_Level_B-I_: I II III IV V

LEVEL III : DATA ABSTRACTION TYPE (ADT) – NON-LINEAR STRUCTURE

Completing Level III is required for all students who wish to participate in any **Open level** of Robotics Competition.

The majority of exercises in this level are external from the book. Many of them take a more pragmatic approach to emphasize its application and analysis and require much beyond just knowing the mechanic of coding.

Learning Outcome:

1) Should be able to tackle *Silver/Gold in the USACO*. Based on our history, all students who demonstrate high proficiency in completing Level III possess a higher chance to be promoted to Silver or Gold level.
2) Complete College level Data Structure.
3) Improvement in modular programming design, including abstraction layers.
4) By the time you complete this level, you are already equipped with sound programming skills comparable to most undergrad CS capacity. Now, you are truly stepping into Computer Science — as opposed to programmings — such as abstraction, correctness, complexity, and modularity. Write code using external libraries when given a library interface.
5) Know how simple Assembler instruction sets work.
6) During November and April, students are encouraged to continue working on USACO problems set and take the Bronze to Silver Level online Exam.
7) Acclimated with Linux system (via Windows Linux SubSystem WSL) (Class notes)
    a. Compile and link programs with makefile
    b. Shared vs. static linked library.-
    c. Automate testing with a set of test data files
8) (Optional) Basics in using Versioning Control System—Git.

Higher Expectation in the following:

1) Error Checking
2) Well tested
3) Concise and elegant , and well-documented
4) No program memory leaks

Learning Resources:

1) *Book required:* Same book for previous level. C Programming: A Modern Approach, 2nd Edition by K.N. King - ISBN-13: 978-0393979503, or 978-0393979503.
2) Class Notes and online packets.

Covered Concepts:

1) Advanced understanding in Advanced memory pointers.
2) FSAs and Regular Expressions
3) Introduction to Digital Logic Gate.
4) Explore a simple Assembler Language.
5) Know how to do generic programming.
6) More seasoned in recursion and understanding basic dynamic programming.
7) Explore Tree and Graph Structure
8) Good understanding of Backtracking Algorithm with BFS for Maze Navigation (explore Tree and Graph structure).
9) Advanced Struct, Union, Enumeration — Chapter-16 and class notes
   a) Adv usage in Struct | Union | Enumeration
   b) Using bits in structure
   c) Big vs. Little Endian, Data Padding and Alignment
10) Advanced Uses of Pointers —Chapter-17.7 and Chapter-18, and class notes
   a) -Double pointer
   b) -Variable list of arguments.
   c) -Deterministic Finite State Machine | Flowchart | Storage Classes (Chapter-18)
   d) Generic programming with void* with Quicksort (Divide and Conquer Algorithm). (Implementation)
11) (optional) Advanced File I/O – memory buffering , redirection
12) Linked List— single | double | circular. (Explore simple tree structure)
   a) -Create their own stack / queue library
   b) -Basic Stack (LIFO)— Push / Pop
   c) Queue (FIFO) concepts— Enqueue / Dequeue (with/without own memory pool)
13) Solving Math Expression - Complex expression with postfix (class notes)
14) System Signal (interrupt) - Chapter 24.3 & 24.4

To conclude Level III:

— This level will conclude the usage of Dr. K.N. King's Book. Students should skim thru Chapter 21, 23 to 25-27 on their own, as they all should be used for reference and self-study purposes only.

— Will complete a final larger scale real-world application project that conglomerates all concepts learned thus far.

— May Practice several USACO Silver and/or Gold level exercises depending on the timeframe and students' analytical skills.

⏮_Level_B-I_: I II III IV V

LEVEL IV - NON-LINEAR DATA STRUCTURE AND ALGORITHMS I (WITH C)

There are two parts of non-linear Data Structure. Part I utilizes primarily C. Both focus on learning through practical application along with more the vernacular necessary in data structures used in some more commonly used algorithms.

You will implement a more complex non-linear data structure instead of using pre-built APIs.

As always, our style is to learn through application. For example, part of the non-linear data structure is the Heap tree widely used in many industrial algorithms. Instead of drilling into, for example, an AVL tree structure, we start with solving a problem/algorithm that requires AVL tree implementation. Application with projects approach enhances the interests level and allows students to see the application upfront.

Learning Outcome:

Students with high proficiency:

- Should be able to tackle *Gold in the USACO*.
- Will be able to execute various algorithms with higher productivity and performance.
- Reach professional quality
- Able to pursue more complex algorithms and conduct advanced projects on their own.

Learning Resources:

1) *Book required:* Mastering Algorithms with C: Useful Techniques from Sorting to Encryption. ISBN-13: 978-1565924536, or ISBN-10: 1565924533. This book will be used for Level IV with additional notes.
2) Class Notes and online packets.

Covered Concepts:

1) Review :
   a) Introduction to Data Structures and Algorithms —Chapter 1
   b) Advanced Pointer Manipulation — Chapter 2
2) Analysis of Algorithms — Chapter 4
   a) Explore NP-completeness
3) Adversarial Search (DFS)
   a) -Minimax Tree with Depth First Search (for Zero-sum games) (Class Note).
   b) -Alpha Beta Pruning (DFS | Tree Structure)
4) Backtracking Maze algorithm with Breath First Search : (Class note)
   a) -Maze Navigation
      - This may be done in a maze setting with two possible challenge: Traverse the whole maze to seek for location of a certain target. BFS to go back to the start point to report the location of the targets

- *** important - Practice Abstraction Layer programming

b) Heuristic search – A* (class note)

5) Review Stacks and Queue with Event Handling – Chapter 6
6) Trees—Chapter 9
   a) Binary Tree Algorithms — Class Notes
      i) Binary Search Tree (completed sorted)
      ii) Self-balance Binary Tree (AVL)
      iii) (optional) Red-Black Binary Tree
   b) Binary-indexed Tree / Fenwick — Class Notes
7) Heaps and Priority Queue—Chapter-10
   a) Loss-less Data Compression with Huffman Coding — Chapter 14 (Greedy Algorithm)
8) Hashing—Chapter 8
   - Chain linked vs Open addressing Methods
   - Understanding loading factor and Collision avoidance
   - Evaluate between two implementations with larger datasets
   - Creating your own dictionary structure and searching with hashing method
9) Sets and Graphs – Chapter 7 & 11
   a) Dykstra Algorithm.
   b) Minimum Spanning Tree
10) Numerical Methods—Chapter 13
11) Knapsack Algorithm (including items and using dynamic programming).
12) Genetic Algorithms
13) Geometric Algorithms with Convex Hull —Chapter 17

To Conclude Level IV

- Completion of all given projects given in this level.

⏮ Level_B-I_: I II III IV V

LEVEL V: NON-LINEAR DATA STRUCTURE WITH ALGORITHMS—2 (C++ & STL)

This course will switch you completely into C++. You will continue to gain deeper understanding in the techniques and data structure implementation skills required by more complex system software and algorithms design. In addition, your work will achieve higher productivity.

Learning Outcome:

1) By the time this level is completed, students should have a sound grasp of Object-Oriented Design Pattern. If you are going into computer engineering or any courseware which involves working with micro-controller libraries, completion of this will help you greatly in mastering utilization of these libraries, as well as design aspect.
2) Students will gain the advantage of using Standard Library (STL) Abstractions to reduce complexity of their own program solutions, increase productivity, but still understanding the C++'s extra features coming with overhead. Besides, C++ exemplifies the usage of abstract data structure with reduced amount of low level detailed work as well.
3) This is designed for students who are interested in Robotics Engineering and AI learning; both realms require high performance. Most resources / libraries written in high level languages for embedded systems are in either C++ or C .
4) C++ is a superset of C, it means it will contain many important features which will reduce your work load vs using just C, especially in arrays, list, string, memory management, and non-linear data structure implementation.
5) (Do note: highly recommend to go into his level if you wish to continue working on USACO Gold+ Level. STL will alleviate much complexity in your coding.) However, nothing can replace with the passion of "problem solving" and "Practice!"

Learning Resources:

1) *Book:*
    a. C++ Primer Plus (Developer's Library) 6th Edition, by Stephen Prata. ISBN-13: 9780321776402.
    b. (Optional) : Data Structures and Algorithm Analysis in C++, 4th Edition by Mark A. Weiss—ISBN-10: 0273769383
2) Class notes (a lot of it)

Outcome

Expect to reach professional level of software development knowledge. Students will gain highly competitive edge in seeking internship which requires software development skills.

Covered Topics

C++ Language Features (differences from C)

1) C++ variables and functions (including reference and overloading)
2) Intro to streams: cout and cin vs. C standard File I/O
3) Namespaces vs. header files
4) New and delete memory allocation
5) Auto and decltype

Objects and Classes

A - - The Basics

1-- Members and methods
2-- Member access control
3-- Constructors and Destructors
4-- Class scope
5-- Const and static members
6-- Comparison to structs
7-- The This pointer
8-- Friend functions
9-- Operator overloading (including C++20 <=> operator)

B - - Inheritance

1-- Basics
2-- Polymorphism
3-- Abstract Base Classes
4-- Inheritance and dynamic memory allocation:
5-- Multiple inheritance

C - - Template classes

1-- Generic programming
2-- Template class
3-- Auto -> decltype() return
4-- Friend functions and Template specialization

Additional language Features

1) Move semantics and Rvalue references
2) Exceptions
3) Runtime Type Identification and Casting (const_cast and reinterpret_cast)
4) Lambda Functions

The Standard Template Library

1) String Class
2) Smart Pointers – in depth
   - What is RAII (Resource Acquisition is Initialization)
   - Scope-Bound Resource Management
   - Grabbing and releasing resources for memory, sockets, etc.
   - How do Smart Pointers apply to RAII
3) STL Containers (greatly enhance productivity vs pure –C)
   - Sequence Containers - vector, linked list
   - Container Adaptors - Queue, Priority Queue, Stack
   - Associative Containers

14) Ordered Set, Multi-set, Map, Multi-map
15) UnOrdered Set, Multi-set, Map, Multi-map
   - Range based For loops

4) Iterators
5) Algorithms
6) Functors
7) Initializer_list

Input, Output and File IO

- iostream, File streams and Stringstream

Advanced Topics (Optional)

1) Multi-threading
2) Socket Communication

3) More advanced performance features in C++17 & up

To Conclude Level V

Revise several algorithms, such as

1) Huffman Coding
2) Shortest Path with Dijkstra
3) Shortest Path with Minimum Spanning Tree
4) KnapSack
5) ML Algorithms such as Linear Regression, KNN, K-means And more (depending on progress)

ALGORITHMS CHECKLIST

This list is a sample list, and may be modified.

With C (all implementation levels, not just conceptual)

1) Quicksort (Divide and Conquer Algorithm
2) Math Expression Parsing—infix to postfix
3) Common Tree Structure algorithms
4) Huffman Coding Compression
5) MinMax & Alpha-Beta Pruning
6) Maze Navigation (Breath First Search / Queuing Structure)
7) Dijkstra's Shortest path
8) Longest Common Subsequence, SubString, Increasing Subsequence
9) Binary-indexed Tree Algorithm
10) KNN (for regression problem)
11) FloodFill

With C++

1) Knapsack Algorithm with Dynamic Programming as well
2) Kruskal's Minimal Spanning Tree
3) Prim's Minimal Spanning Tree
4) Travelling Salesman Problem
5) A* algorithm
6) Classroom scheduling
7) Lempel Ziv (LZ78)
8) SHA-1 or 5
9) Genetic Algorithm (for job scheduling)

COPYRIGHT