Search: ~dwharder ECE Undergrad UW Search

E
U dies
Douglas Wilhelm Harder
Algorithms and Data Structures Home
Projects
List of all projects

List of all projects

In order to ensure that it is more difficult for students to be tempted by cheating, each of the projects has numerous possible projects associated with it. These projects should change from term to term.

1. linked-list–based data structures,
2. array-based data structures,
3. tree-based data structures,
4. hash-table based data structures, and
5. graph data structures and algorithms.

For access, please contact the author at dwharder@uwaterloo.ca.

1. Linked-list–based data structures

Project 1 will always be comprised of a linked-list projects:

1. Cyclic_double_list
2. Cyclic_double_sentinel_list
3. Cyclic_list
4. Cyclic_sentinel_list
5. Double_list
6. Double_sentinel_list
7. Sentinel_list
8. Single_list
9. Sorted_double_list
10. Sorted_double_sentinel_list
11. Sorted_sentinel_list
12. Sorted_single_list

Each class above requires one of the following two node classes:

1. Double_node
2. Single_node

2. Array-based data structures

Project 2 will always be comprised two projects. One will be a statically sized array-based container, while the other will be dynamic. In general, one should be stack-based, and the other either queue or deque based.

The static memory projects are:

1. Stack
2. Queue
3. Deque
4. Drop_off_stack
5. Navigation_stack
6. Range_stack
7. Dual_stack

The dynamic memory projects are:

1. Dynamic_stack
2. Dynamic_queue
3. Dynamic_deque
4. Dynamic_navigation_stack
5. Dynamic_range_stack
6. Dynamic_dual_stack

The linked-memory projects are:

1. Linked_stack
2. Linked_queue
3. Linked_deque
4. Linked_navigation_stack
5. Linked_range_stack

Students will be allowed to use the STL `std::list` class for the linked-memory projects.

2.1 Definitions

Dynamic
: The size of the array will be increased when the container is full, and decreased when the container is one-quarter full.

Linked
: The container will be made up of a linked-list of arrays of a fixed size. When one array is filled, a new node will be added with a new array.

Navigation stack
: A stack with additional *backward* and *forward* operations mimicing the behavior of recording events to allow *undo* and *redo* operations.

Range stack
: A stack that records not only the entries but also the current minimum and maximum entries in the stack. This can be implemented by having three arrays or a stack of a struct containing three entries.

Dual stack
: One array permits two stacks, one growing from each end.

Drop-off stack
: A stack where, if the stack is full, discards what is currently at the bottom of the stack and places the new object on the top.

3. Tree-based data structures

Project 3 will always be comprised one project related to node-based trees.

1. AVL tree
2. B tree
3. Expression tree
4. File system
5. Lazy deletion tree
6. Quad-tree

4. Hash-table data structure

Project 4 will usually be comprised of a hash table. Previously it has also possibly been a heap-based structure, but it is more useful to have a hash table structure.

1. Binary heap
2. Cuckoo hash table
3. Double hash table
4. Dynamic double hash table
5. Dynamic linear hash table
6. Dynamic min heap
7. Heapify
8. Linear replacement hash table
9. Quadratic hash table
10. Quaternary heap
11. Stable binary heap
12. Ternary heap

5. Grpah data structures and algorithms

Project 5 is always graph based.

1. Dijkstra's algorithm
2. Prim's algorithm
3. Topological sort
4. Kruskal algorithm

Algorithms and Data Structures
Department of Electrical and Computer Engineering
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
Phone: +1 519 888 4567 extension 37023
Facsimile: +1 519 746 3077