ECE 250 - Project 1
Dynamic Deques
Design Document

Jamie Yen, UW UserID: j6yen

Oct. 16th, 2022

I. Overview of Classes

Class 1: Deque

Description:
A type of queue data structure implemented using a doubly linked list. It is used to store internet browser history in the list using Node class along with some supporting functions to manipulate the lists, such as deleting, adding, and finding the URL.

Member variables:
1. Node type pointer p_head
2. Node type pointer p_tail
3. Integer size
4. Integer max_size

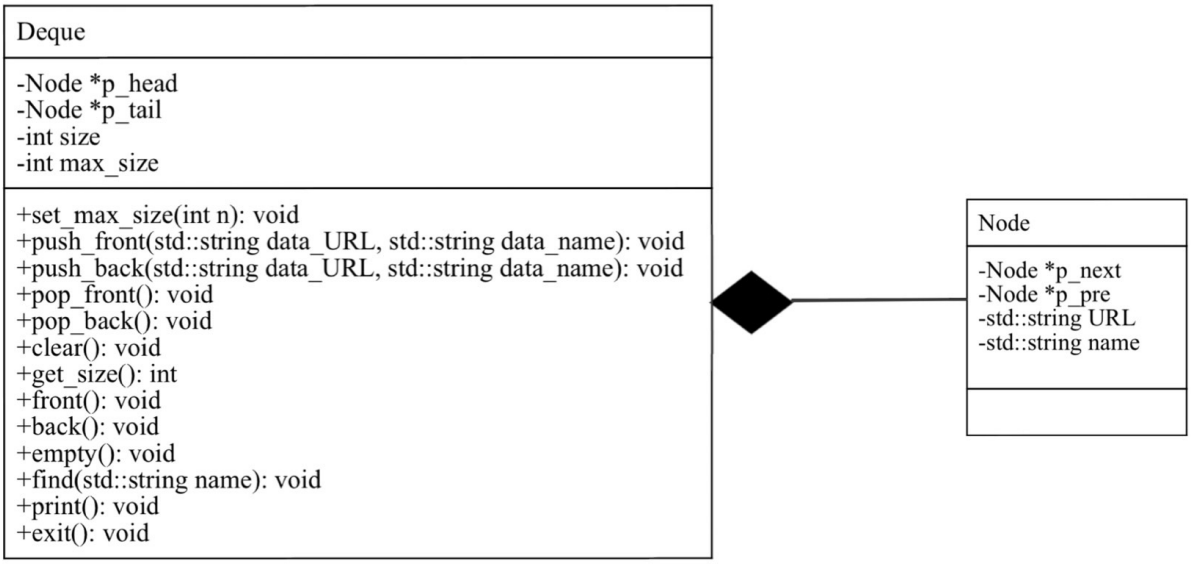
Member functions:
1. **set_max_size**: it sets the queue with maximum capacity n
2. **push_front**: inserts the name and URL as a node at the front of the doubly-linked list
3. **push_back**: inserts the name and URL as a node at the end of the doubly-linked list
4. **pop_front**: removes the first node of the doubly-linked list
5. **pop_back**: removes the last node of the doubly-linked list
6. **clear**: removes all the browsing history
7. **get_size**: returns the number of URLs currently in the list
8. **front**: print out the name and URL stored in the first node
9. **back**: print out the name and URL stored in the last node
10. **empty**: returns a bool telling whether the deque is empty
11. **find**: finds if the URL-name is stored in the list
12. **print**: prints all the browser history starting from the end of the linked list
13. **exit**: ends the input file

Class 2: Node

Description:
A class representing a node

Member variables:
1. Node type pointer p_next
2. Node type pointer p_pre
3. String URL
4. String name

II. UML Class Diagram



III. Constructor and Destructor

Class Node:

Constructor: initializes the data to be empty and both of p_next and p_pre pointing to null pointer

Destructor: does nothing

Class Deque:

Constructor: initializes the p_head and p_tail pointing to null pointer

Destructor: does nothing

IV. Test Cases

Test 1: Check if the doubly linked list can be created properly with a limited size that the user requested.

Test 2: Check if a browser history can be added to the front playlist and if the list can be printed out.

Test 3: Check if pop_front and pop_back can remove the nodes as desired and fail the remove the nodes if the list is empty

Test 4: Check if a URL-name can be found in the list, and not found when it's not in the list.

Test 5: Check if the size of the list gets modified correctly after a series of pop_front, pop_back, and inserting.

Test 6: Check if the element at the rear of the list got removed if the deque is full after calling push_front or push_back

Test 7: Check if all the elements are removed after clear is called, and the front and back function fails to return the first and last node since the list is now empty

V. Performance

The deque class is a doubly linked list that performs inserting, deleting, and finding the nodes in constant O(1) time. For the clear and print function, the performance is O(n) time.