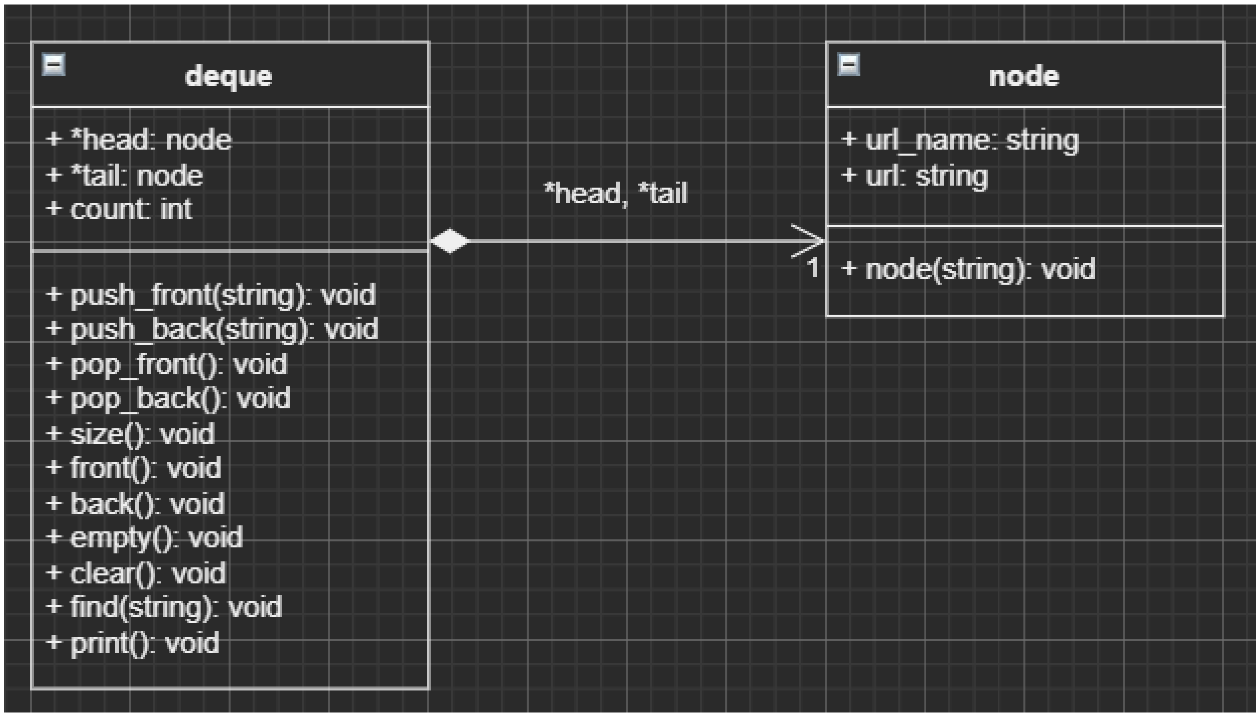
Project 1 Design Document

Zhiyuan Zhang z667zhan

1. Overview of classes

Two classes are implemented in this design, one for the deque and the other for the entries in the deque (nodes). Class ‘deque’ stores the head node and tail node of a deque, as well as the total number of nodes in that deque, with functions to initialize and destruct deque, to push and pop nodes, and to access various data stored in this deque. Class ‘node’ stores name and URL, and has pointers point to its previous and next nodes, with functions to initialize, set value on and destroy nodes.

2. UML class diagram



3. Design decisions:

<u>For deque class:</u>

For a double ended queue, when it is constructed the head and tail nodes will be NULL; to destruct this deque every node in it needs to be destroyed to free the memory. The size of deque is constantly changing so no ‘const’ will be used here.

<u>For node class:</u>

It has two strings, and the method is used to set these parameters only.

4. Test strategy:

For pushing nodes, it should always push a node at the beginning/end of the deque.

For popping nodes, it should always pop a node unless the deque is empty.

It should always return the correct size of deque; it should always return whether the deque is empty.

It should always return the first/last element in the deque unless the deque is empty.

For clearing, it should always remove all elements from the deque.

For finding nodes, it should always find the specific node if input URL-name is the same as said node; the search should not be case sensitive; it should not find a node if no URL-name in the deque is the same as input.

For printing deque, it should always print all entries in the deque starting at the back; it should not print anything if the deque is empty.

5. Performance analysis:

push_front(name, URL) & push_back(name, URL): O(1) since it only interacts with the first/last node regardless of the size of deque. It creates a new node, makes this node head/tail, increment count, and makes this node point to the previous head/tail; all in O(1) time.

pop_front() & pop_back(): O(1) since it only interacts with the first/last node regardless of the size of deque. It deletes the first/last node, makes the second node head/tail, and decrement count; all in O(1) time.

size() & empty(): O(1) since it only access the count value of deque.

front() & back(): O(1) since it only access the first/last node.

clear(): O(n) since it needs to go over the entire deque to delete all nodes.

find(name): O(n) since it needs to go over the entire deque to check whether there exists a URL-name that matches the input.

print(): O(n) since it needs to access all nodes to print all entries.