Harmon Mendelowitz

CS3410 Systems

HW00

Ring Size: Check if the ring parameter is null. If so, return -1, otherwise set a new int variable ‘s’ to the number of the ring’s write variable minus the ring’s read variable. If ‘s’ is negative, set it to s plus the ring’s total size to accommodate for the loop-around. Return ‘s’.

Ring Enqueue: Check if the ring parameter is null and return -1 if it is. Check if the ring buffer is full and return -1 if it is. This is done by checking if the ring’s write variable + 1 modulo the ring’s size is equal to the ring’s read variable. In other words, if the write integer is one place behind the read integer, accounting for the looping of the ring. If the ring is not full, then set the ring’s data at the write location to the data entered in the function’s parameter and increment the write variable by setting it equal to the current number plus 1, modulo the ring’s size. This function essentially moves the write value forward once. Return 0.

Ring Dequeue: If either the ring or the pointer from the parameter are null, return -1. Otherwise, check if the buffer is empty by comparing the values of the ring’s write and read variables. If they are equal, then the ring is empty and -1 is returned. If the ring is not empty, then set the data pointer’s value to the value at the ring’s read location and increment the read variable by setting it equal to read plus 1, modulo the ring’s size. Return 0.

Ring Apply FIR: Check if the input ring is null and returns 0 if it is. Checks if the buffer is full using the method described above and returns 0 if the ring is not full. Otherwise, make a new integer, ‘r’, and set it equal to the ring’s read value and a double used to calculate a sum. Loop a number of times equal to the ring’s size minus one and within the loop do two things: First, add the ring’s data at the location of ‘r’ multiplied by the data at the corresponding location of the array input to the function, to the double used for summation. Next, increment ‘r’ using the formula of setting ‘r’ equal to ‘r’ plus 1, modulo the size of the ring. After the loop is finished, return the value of the sum.

Time Complexity:

Ring Size: O(1) since in order to get the size of the buffer, only the ring’s write and read variables are compared one time, so the time complexity does not depend on the size or length of the ring.

Ring Enqueue: O(1). The operation contains no loops and only references variables contained in the ring object without accessing any values within the ring other than the one vale designated by the write variable. This interaction takes the same time regardless of total size since the write variable will always refer to the intended location.

Ring Dequeue: O(1). Again, data is stored in the ring at the index of the ring’s read variable and read’s value is incremented one index forward. This action does not depend on the ring’s size or the number of values currently stored in the ring.

Ring Apply FIR: This method is O(n) since it contains a loop which is run ring’s size minus one times and thus the runtime of the method is dependent on the total size of the ring buffer.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedPlots:

Analysis: By changing the ring buffer size, more values could be stored at any given time, though proper management of space would work as well. The drawback of an increased ring buffer size would be that the apply filter method would take longer to execute. In addition, changes to the filter weights would have a drastic effect on the filtered graph due to uneven weightings of some elements over others.

While listening to the three generated audio files, the formation of a distinct pattern could be heard from the corresponding “filtered” wave where there existed none in the “noisy” wave. The constant ringing sound was in the background of a large amount of noise, but in comparison to the “clean” wave, one could observe a similar frequency and pattern of sound.

By comparing the graphs above, an obvious improvement can be seen between the noisy and filtered plots. While the change between the two is less profound auditorily, the difference between the noisy and filtered graphs is much further apart than the difference between the filtered and clean graphs. The filtering program did make the intended sound recognizable, but it certainly could be improved upon. Overall, the filter was effective at reducing the provided noise, though it could quickly become obsolete when faced with a more complex scenario.