Robert Berkstresser

Analysis of Playlist Implementations Using Reference and Array Based Lists

The three playlist classes implemented using a linked list, circular linked list, and an array based list where tested, using randomly generated Song objects, in four categories:

1. Insertion
   * 500,000 values where inserted in random indexes in the list
2. Retrieval
   * 500,000 values where retrieved from random indexes in the list.
3. Swap
   * 1,000,000 elements exchanged positions in the list.
4. Shift
   * 500,000 Shifts were done ranging from -1,000,000 to 1,000,000

The insertion test for circular and straight linked lists produced a run time of 327 minutes and 30 seconds and 317 minutes 00 seconds respectively. Both lists produced an O(n) run time with the circular linked lists showing a 3% improvement over the non-circular version due to the constant runtime of insertions at the end of the list and the nature of optimizations made to the circular version of the list. In addition duplicate checking may have slowed some of the circular linked lists optimizations and decreased the margin between the two run times.

The retrieval test for circular and straight linked lists produced a run time of 11 minutes 08 seconds and 11 minutes 04 seconds respectively. The linked list showed a negligible improvement over the circular version most likely due to interference from other processes on the machine. Both implementations require that the entire list be searched in the worst case so the run time is O(n) with no apparent impact from any optimizations available in the circular linked list.

The swap test for circular and straight linked lists produced a run time of 22 minutes 20 seconds and 22 minutes 06 seconds respectively. The circular linked list showed a negligible improvement over the non-circular version most likely due to interference from other processes on the machine. Both implementations require that the list be traversed n and n - 1 times in the worst case. Therefore the run time is 2n-1 or O(n) with no apparent impact from any optimizations available in the circular linked list.

The shift test for circular and straight linked lists produced a run time of 45 minutes 48 seconds and 44 minutes 55 seconds respectively. The circular linked list showed a negligible improvement over the non-circular version most likely due to interference from other processes on the machine. Both implementations require that the list be traversed n and n - 1 times in the worst case. Therefore the run time is 2n-1 or O(n) with no apparent impact from any optimizations available in the circular linked list.

At the time of writing this paper no working implementation of an Array based list has been committed to the project. Even so some conjectures can be made about the advantages and disadvantages of the Array based list against the referenced based implementations. Insertion into an array based list implementation is considered an O(n) operation, but with proper optimizations of the array growth and shrinkage algorithms this may in practice be closer to an O(1) run time. This “average case” constant run time is certainly less than the linked lists “average case” of O(n) and matches the circular linked lists time for tail insertions (O(1)). Also retrieval from an array based list implementation will always be O(1), and easily trumps the reference based versions O(n). Both the swap and the shift tests rely on both of the operations above and therefore also have some of the benefits over the reference version. Swap will always be a constant run time O(1) and shift will be O(n) because all values must be copied after their new positions are calculated.

In conclusion the advantages of a referenced based list in quick creation from the tail and no need to resize an internal array seem to be outweighed by the O(1) best and worst case access time and the “average case” O(1) of the optimized insert when the list is used as a random access linear data structure.