Albert Wu

CptS 223

PA 2 Report

**A: Problem statement**

The goal of this assignment was to compare the performance of lists versus vectors (resizing array) on a problem that required iterating through, reading, and deleting elements.

**B: Algorithm design**

For both vectors and lists, my high-level algorithm was the same. When initializing the circular list/vector class at the start of the game, I simply added Person objects, with incrementing IDs starting from 1 (representing position), up to a total of N elements. I set an iterator to start at the first element. For each elimination, I simply incremented the iterator M times, then deleted the element at the iterator (the library implementation and speed of which depends whether a list or vector was being used), then set the iterator to the next element. Whenever the iterator was set to past the end of the list, I immediately set it back to the first element.

**C. Experimental setup**

I used a MacAir with a 2.2 GHz Intel Core i7 CPU, with 8GB RAM. I used Mac OSX for testing and timing.

The results are only from 1 experiment, but to check for consistency I performed each experiment 3 times and got consistent results each time.

Tested on EECS servers with:

g++ VectorMyJosephus.cpp ListMyJosephus.cpp TestVectorMyJosephus.cpp TestListMyJosephus.cpp TestAll.cpp Person.cpp -o result -std=c++0x -lrt

**D. Experimental results and discussion**

Plots (in order from 1-4 for lists, then vectors)

\*graphs from the data in “plot.csv”

For the list implementation the total running time increases linearly as N increases.

For the vector implementation the total running time seems to be polynomial. This is consistent with theoretical expectations as the runtime for the vector implementation should be O(N^2), as each deletion on average takes O(N), while the runtime for the list implementation should be O(N), as each deletion takes constant time, and there are N-1 deletions.

Comparing the total running times while varying N (M=3), you can see that the implementations seem to be about the same speed. This at first seems inconsistent with theoretical expectations. However, we likely simply did not test on large enough input sizes for the polynomial growth difference in deletion time to overtake the difference in constants, as for each elimination we had to iterate through M=3 elements, which is likely significantly faster with a vector implementation than a list.

For both the list and vector implementation the total running time increases linearly as M increases. For the list implementation this is consistent with theoretical expectations as each elimination takes O(M + 1) time (and the number of eliminations is equal to N-1, which is kept constant). For the vector implementation, each elimination takes O(M+N) time, and as N is kept constant, we can observe a linear increase in runtime as M increases.

Comparing total running times while varying M (N=1024) for list versus vector implementations, you can see that the vector implementation is consistently faster. This is consistent with theoretical expectations. We have evidence from comparing total running times over N that N=1024 is not large enough for the difference in degree of growth of vector vs list deletion time to matter. However, each deletion we have to iterate through M elements, for which vectors are a constant time faster than lists.

The average elimination time for both vector and list implementation seemed to be constant as N was increased, and linear as M was increased. For the list implementation, this is consistent with our previous analysis of the complexity of each deletion. For the vector implementation, this adds further evidence to our previous hypothesis that the values of N we are using are too small for the polynomial growth of the deletion time to show its effects.