Joseph Lewis

9/27/2018

PA 3 Report

CS 223

**Problem Statement:**

The goal is to successfully code two algorithms that successfully will complete the Josephus problem. The running times of the two algorithms will be compared. The first algorithm uses a list to store the remaining players, while the second algorithm uses a vector.

**Algorithm design:**

We were given three classes that would support us while programming the assignment. We had to work off the given code. The first class was a called ‘Person’. This classes had the integer of the player’s position, which included a constructor, destructor, and a print function. I added two functions that were a getter and a setter for the integer position in the class.

The second and third class was called ‘ListMyJosephus’ and ‘VectorMyJosephus’. These classes included needed functions in header file that I needed to work of off using a different file. The classes also included the integers size, N, and M in the private parameters. Size was used to determine the amount of people who were still in the game. N is the value of the number of players at the beginning of the game. M is the value of the number of passes until a player is eliminated. I added an integer to the private parameter named ‘pot’, which is used to determine which player gets eliminated. The two classes are very similar, but one uses a list in the private parameter, while the other uses a vector. These two structures are used to store players that are still in the game. The structures names were called ‘circ’.

To be able to use both classes I needed to first create constructors and destructors in the right files. Then I coded the function ‘init’ which is used to create a new game by adding player to the game using a for loop and the data structure ‘circ’. The for loop use a Person to assign values to using a setter and then the Person value were pushed in the data structure circ. I also used ‘init’ to set M, N, and size. After that I created a function that would determine if ‘circ’ is empty. This is used to determine when the game should stop. The next function I needed to create was one that would remove a data entry on ‘circ’. The function ‘eliminateNext’ was used to eliminate a player in the Josephus game while returning the eliminated player. To complete this function, I assigned an integer named count to equal pot plus M. Then I created a while loop that would subtract count by size when count is greater than size. After that, I program that function to eliminate the player at the position equal to count. Then I would assign pot to count. After that, I created a function that printed every player in the structure circle. Finally I created two files that would test the Josephus game using the list and the vector.

**Experimental setup:**

I first had to code the program that Vectors, and Lists can have running times compared using the Josephus problem. To record running time, I used to Chrono library to record the time of each function in microseconds. The CPU I used for this experiment is a “Intel(R)Core(TM)i7-4710HQ”, which is a 64-bit operating system. The clock speed used for testing was 2.50 GHz. The RAM use for this experiment had a memory value of 8 gigabytes with a speed of 1600 MHz. I completed experiment one and two stated in the assignment handout, completing one trial to test both Josephus programs to compare running times. The running times recorded were in micro seconds. All the testing in this experiment was conducted in the EECS SSH server, which involved the Unix environment using g++. Finally, a plotted my results and compared the running times.

**Experimental Results and discussion:**

**Plot 1**

**Plot 2**

**Plot 3**

**Plot 4**

Discussion-

Overall my results did confirm my expectations. The vector preformed the best and the running time depended on N, but not M. My observations confirmed that the list results had a higher running time than the vector results. The Tested results did not depend on the inputs because the list had the highest running time in all the inputs in the trials. The reason why list performed the worst, because it had a larger running time complexity. The running time complexity for completing the game was theta(N\*X). N is the size of the players playing the game. X is the time complexity for the function that eliminates a player. The list data structure had a longer running time complexity for the function that eliminates a player than the vector. This was caused by the list having more commands required than the vector to operate the program. The vector preformed like a normal array, while the list required an iterator to complete most of the programing. My observations also showed that running time dependent on the value N and not the value M. In experiment one, the running times increased as n increased. In experiment two, the running times had not increase or decrease due to the change of M. This was caused by the running time complexity being equal to theta(N\*X) described in the previous sentences.