**Reflection on the importance and efficiency of the use of doubly linked lists in a problem situation of this nature**

The importance of sorting algorithms is that they allow the user to sort the lines in a text file, is important for optimizing the efficiency of algorithms (such as search and merge used in this code) that require input data to be in sorted lists. Sorting is also often useful for canonicalizing data and for producing human-readable output. The computational complexity of the merge is O(n log n)

The importance of searching algorithms is that they allow the user to search for specific lines in a text file in this case between start and date it helps to simplify the visualization of the output. The computational complexity is in the best case O(1) and in the worst O(n) so the average will be O(n/2)

Efficiency plays a significant role in applications in a high-execution-speed environment where performance and scalability are paramount. One of the recommended best practices in coding is to ensure good code efficiency. Well-developed programming codes should be able to handle complex algorithms.

Steps:

1. The program reads the file and saves all the lines in a vector

2. The program adds a zero if the day is only one digit3. The program sorts the lines with merge sort

4. The program saves the sorted lines in a new file5. The program asks the user for the IP to normalize

6. The program saves the lines that are between the start and end IP in a new file

7. The program prints the lines that are between the start and end IP

The importance on use a double linked list is that you can go back and forth in the list without having to go through the whole list again and again to find the previous node or the next node of the current node you are in the list; some examples of the use of a double linked list are the undo and redo functions in a text editor, the back and forward buttons in a web browser, the history of a web browser, the list of songs in a music player, the list of files in a file explorer, etc

Reflection: I think that the program works well, but it could be improved by using a binary search instead of a linear search to find the start and end index. I also think that the program could be improved by using a hash table to store the lines instead of a vector, because the hash table would be faster to search.

Time complexity: O(n log n) because the merge sort is O(n log n) and the linear search is O(n) and the program does both.