The discussion on the choice between ArrayList and LinkedList for list manipulation prompts a thoughtful reflection on the intricacies of data structures and their performance implications. The insight that LinkedLists outperforms ArrayLists in scenarios involving repeated additions and removals from the middle highlights the importance of aligning data structure choices with the specific requirements of the algorithm at hand. In my case, where the majority of list manipulations involve appending or prepending elements, both ArrayList and LinkedList exhibit efficiency, with LinkedList potentially holding a marginal advantage.

The consideration of time complexity for accessing elements by index sheds light on another dimension of the decision-making process. While ArrayList boasts a constant time complexity for such operations, LinkedList's linear complexity may pose a concern. As my algorithm primarily involves sequential access to elements, the efficiency gains from ArrayList become apparent. This prompts a deeper reflection on the nature of the operations performed within the algorithm and the overall implications for runtime performance.

In conclusion, the analysis of ArrayList and LinkedList reveals that the choice between them hinges on the specific operations undertaken in the algorithm. The potential slight edge of LinkedList in scenarios with frequent insertions or removals in the middle needs to be weighed against the generally superior sequential access efficiency of ArrayList. Applying this insight to the proposed algorithm in Unit 1 emphasizes the need for a meticulous examination of the operations involved, ensuring that the selected data structure aligns seamlessly with the algorithm's requirements. This reflective process underscores the importance of a nuanced and informed decision-making approach in optimizing algorithmic performance.

In delving into this project, I gained valuable insights into the nuanced implementation differences and considerations inherent in such a switch. The clarity and well-structured nature of the code greatly facilitated my understanding of the necessary modifications. However, I recognize the potential for deeper comprehension through more detailed explanations or comments elucidating the performance considerations and trade-offs specific to the project's context. Given more time, I would undertake comprehensive performance testing to quantify and analyze the efficiency disparities between ArrayList and LinkedList across diverse scenarios. Additionally, I would explore alternative algorithms or optimizations, aiming to further elevate the program's overall performance. This learning experience underscores the importance of not only code implementation but also a thorough understanding of the underlying data structures and their implications on program efficiency.

A graph with a line

Description automatically generated

|  |  |
| --- | --- |
| Number of Rounds | Time in ms |
| 10 | 232 |
| 100 | 1289 |
| 1000 | 15793 |
| 10000 | 200825 |