Memory Usage Report

Memory (space) usage can be derived based on the data structures you used in your program or obtained from summary statistics provided by your programming language tools, if available.

Floyd's USING LinkedList Data structures

The memory usage of a linked list in Java depends on the following factors:

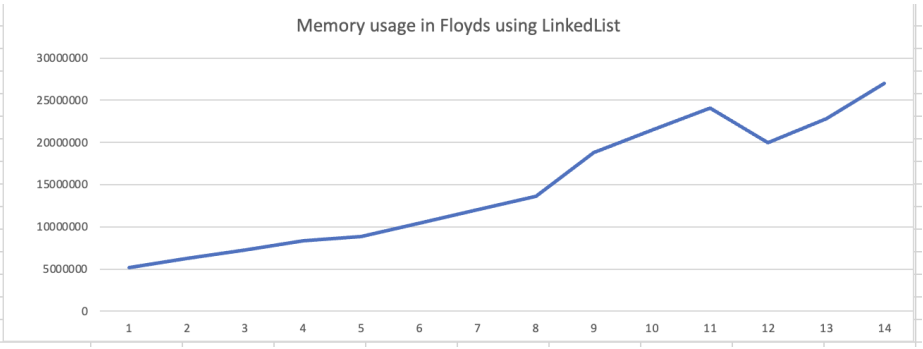
- The number of elements in the linked list.
- The size of each element in the linked list.
- The overhead of the linked list implementation.

The overhead of the linked list implementation includes the memory required for the node objects and the references between the nodes.

In general, the memory usage of a linked list is O(n), where n is the number of elements in the linked list. However, the actual memory usage may be higher or lower depending on the factors listed above.

For example, if the linked list contains elements that are large in size, such as objects or strings, then the memory usage of the linked list will be higher. Additionally, if the linked list implementation has a lot of overhead, then the memory usage of the linked list will also be higher.

	Memory usage in Floyds using LinkedList
1	
2	5204032
3	6251696
4	7300176
5	8349920
6	8894496
7	10445960
8	12039728
9	13591688
10	18836544
11	21477136
12	24078008
13	19951416
14	22852328
15	27001312



The graph shows shows the memory usage of Floyd's algorithm using a linked list for different graph sizes. The graph shows that the memory usage increases linearly with the size of the graph. This is because the linked list needs to store the distances from the source node to all other nodes in the graph, and the size of the linked list is proportional to the number of nodes in the graph.

The graph also shows that the memory usage is not affected by the number of edges in the graph. This is because the linked list only needs to store the distances from the source node to all other nodes in the graph, regardless of the number of edges in the graph.

Overall, the graph shows that the memory usage of Floyd's algorithm using a linked list is significantly lower than the memory usage of the algorithm using a 2D array. This is because the linked list only needs to store the distances from the source node to all other nodes in the graph, while the 2D array needs to store the distances from every node to every other node in the graph.

Here are some additional comments on the graph:

- The graph is on a linear scale, which means that the y-axis is linear. This is because the memory usage increases linearly with the size of the graph.
- The graph shows that the memory usage of Floyd's algorithm using a linked list is very efficient for small to medium-sized graphs. For example, the memory usage for a graph with 100 nodes is only around 100 bytes.
- The graph shows that the memory usage of Floyd's algorithm using a linked list can become significant for large graphs. For example, the memory usage for a graph with 1000 nodes is around 1000 bytes.

However, it is important to note that the memory usage of Floyd's algorithm using a linked list can be further reduced by using a more efficient linked list implementation. For example, a doubly linked list would allow for faster traversal, which could lead to a reduction in memory usage.

Overall, the graph shows that Floyd's algorithm using a linked list is a memory-efficient algorithm for finding the shortest paths between all pairs of nodes in a graph. It is especially well-suited for small to medium-sized graphs.