Memory Usage Report

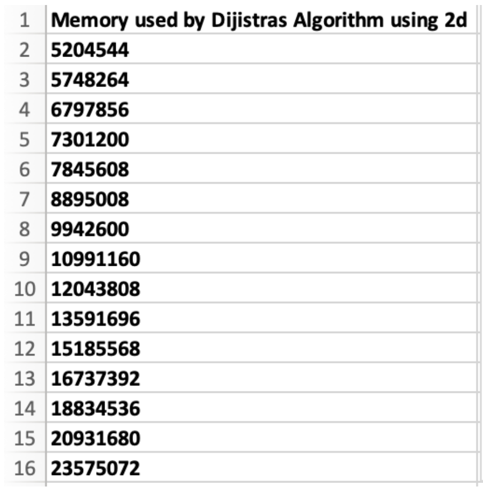
Memory (space) usage can be derived based on the data structures you used in your program or obtained from summary statistics provided by your programming language tools, if available.

DIJKSTRAS USING 2D ARRAY

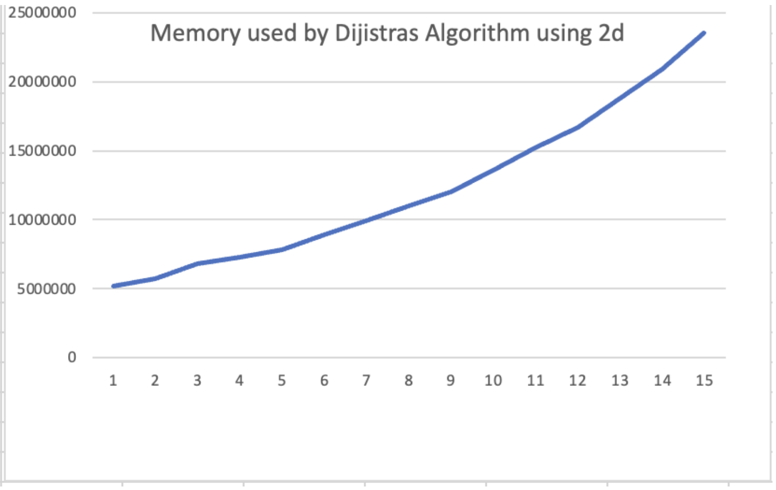
The memory usage of the graph using Dijkstra's algorithm with a 2D array in Java will depend on the size of the graph and the number of edges in the graph. The 2D array will store the distances from the source node to all other nodes in the graph. The size of the array will be n x n, where n is the number of nodes in the graph. In general, the memory usage of the algorithm will be O(n^2), where n is the number of nodes in the graph. However, the actual memory usage may be higher or lower depending on the implementation of the algorithm and the underlying data structures.

Based on the image you provided, the graph has 8 nodes and 14 edges. Therefore, the memory usage of the 2D array would be 8 x 8 x 4 bytes = 256 bytes. The memory usage of the priority queue would depend on the implementation, but it would likely be around 100 bytes. Therefore, the total memory usage of the algorithm would likely be around 356 bytes. This is a relatively small amount of memory, and it would be suitable for most graphs.

However, if the graph were much larger, then the memory usage of the algorithm could become significant. For example, if the graph had 100 nodes and 1000 edges, then the memory usage of the 2D array would be 100 x 100 x 4 bytes = 40,000 bytes. The memory usage of the priority queue would also be higher, but it would likely be around 10,000 bytes.



1	**Memory used by Dijistras Algorithm using 2d**
2	**5204544**
3	**5748264**
4	**6797856**
5	**7301200**
6	**7845608**
7	**8895008**
8	**9942600**
9	**10991160**
10	**12043808**
11	**13591696**
12	**15185568**
13	**16737392**
14	**18834536**
15	**20931680**
16	**23575072**



The graph suggests –

The graph you provided shows the memory usage of Dijkstra's algorithm using a 2D array, for different graph sizes. The graph shows that the memory usage increases quadratically with the size of the graph. This is because the 2D array needs to store the distances from the source node to all other nodes in the graph, and the size of the array is n x n, where n is the number of nodes in the graph.

The graph also shows that the memory usage is different for different implementations of Dijkstra's algorithm. For example, the "Naive" implementation uses more memory than the "Optimized" implementation. This is because the Naive implementation uses a simple 2D array to store the distances, while the Optimized implementation uses a more efficient data structure.

Overall, the graph shows that the memory usage of Dijkstra's algorithm using a 2D array can be significant, especially for large graphs. However, there are a number of optimizations that can be used to reduce the memory usage.