Making Friends Report

Bao Trung Hoang

April 24, 2025

1 Results

Briefly comment the results, did the script say all your solutions were correct? Approximately how long time does it take for the program to run on the largest input? What takes the majority of the time?

The solution passes all the test case, including the largest one. The program takes about 1 minute to run on the largest input. Most of the time is spent on the largest test case, which is the last one.

The process tasks most of the time would be the adjacency list traversal. For each newly visited node, we need to check all the neighbors of that node. This is costly because it has the time complexity of $O(n)$, which will be came significant for the largest input, containing $3 * 10^6$ nodes.

2 Implementation details

How did you implement the solution? Which data structures were used? Which modifications to these data structures were used? What is the overall running time? Why?

The solution is implemented in Python 3.12.3, adapted to the given data. The following data structures are used for the solution:

- The adjacency list is used to present the graph.
  - The index position of the list represents the node number.
  - The value at that index is a list of all neighbors of that node.
  - A neighbor is presented as a tuple of the neighbor node number and the weight of the edge connecting them.
- The priority queue is used to store the edges with the smallest weights being processed.
  - An empty list is used for this purpose.
  - The **heapq** module is used to transform the list into the priority queue.

  - The **heapq.heappush()** function is used to add a new edge to the queue, which ensures that the edge with the smallest weights is retrieved fist when using the **heapq.heappop()** function.
  - The edges connected to a defined first node will be added to the queue. The data structure stored in the queue is **(weight, (source_node, destination_node))**, which is a tuple.

- The **visited** list is used to keep track of the nodes that have been visited.
- The **nodes** list used to store all the node numbers avalilable in the graph.

The following modifications are made to the data structure for the given data:

- The node number is shifted by 1 to match the index position of the adjacency list.
- The priority queue is organized based on the value of the first element, which is the weight of the edge.

The time complexity of the solution is as follows: