Task-05: Comparison

* Adjacency List (Input)

Initialising the adjacency list takes $O(v)$ time. For my adjacency list, I used a dictionary in which the nodes were the keys and the neighbors were the values of the keys in the dictionary. The total complexity for the inputs will be $O(v) + O(E) = O(v+E)$

* Adjacency Matrix (Input)

For creating a matrix of $n \times n$, or, specifically, $v \times v$ size, we use a nested loop. So, the time complexity would be $O(v^2)$

* BFS using adjacency list

The visited BFS array takes $O(v)$ time. For the queue, enqueuing and dequeuing and setting up the visited list takes constant time. When the queue becomes empty, the complexity for this part becomes $O(v)$

Adding the nodes, the loop runs E number of
times. So, the overall time complexity ~~for this~~
for BFS, O(v) + O(v) + O(E) = O(v + E)

* DFS

Creating the visited array takes O(v) time.
The DFS function has been called recursively
for v times, i.e. O(v). So, the overall time
complexity could be, O(v) + O(v) + O(v) + O(E) = O(v+E

Better algorithm → BFS or DFS? (for reaching early)
DFS visits less nodes than that of BFS. BFS
searches like level order search of a tree,
i.e. it visits all the nodes. On the other hand,
DFS reaches deeper and deeper until it reaches
the dead end. So, we can find closer nodes faster
using BFS and farther nodes faster using DFS.
n the case of Ash and Gary, Gary will reach
victory road faster as he used DFS algorithm
and his destination was far from the start
point.