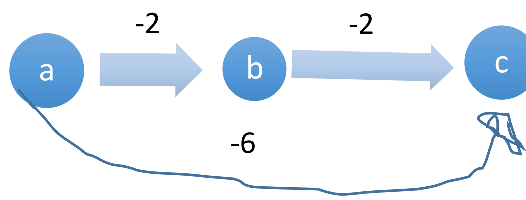
1. **Professor F. Lake suggests the following algorithm for finding the shortest path from node s to node t in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra's algorithm starting at node s, and return the shortest path found to node t. Is this a valid method? Either prove that it works correctly, or give a counterexample.**

   It doesn't work! Looking at the graph below, suppose we add 7 to make all the nodes positive, the a-b is now 5, b-c is also 5 and a-c is now 1. Dijkstra's would now interpret a-c as the shortest path, instead of a-b-c which is the actual shortest. This is because it interprets -6 to be smaller than -2 (logically true) but that only refers to direction usually, so we need to consider the absolute values.



2. **There is a network of roads G = (V, E) connecting a set of cities V. Each road in E has an associated length le. There is a proposal to add one new road to this network, and there is a list E 0 of pairs of cities between which the new road can be built. Each such potential road e 0 ∈ E 0 has an associated length. As a designer for the public works department you are asked to determine the road e 0 ∈ E 0 whose addition to the existing network G would result in the maximum decrease in the driving distance between two fixed cities s and t in the network. Give an efficient algorithm for solving this problem and analyze its time complexity.**

   This is actually quite similar to an airline algorithm that determine routes between hubs! We have two fixed cities, s and t. We run Dijkstra's algorithm on them towards any sufficiently far enough node to get the shortest paths from each. Given the shortest path values, we can iterate over E' (pairs of cities that the road can be built in) to find which path gives us the shortest paths from s to t with this new road e' of length $l_{e'}$in the middle of the two shortest paths:
   Algorithm: D(S) + D(T) + $l_{e'}$
   Our time complexity would be linear: O(|E'|) for iterating and Dijkstra's takes O(|E|log |V|)[1]. We disregard the constants for Dijkstra's and get:
   O(|E'|)+O(|E|log|V|)

[1] (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Running_time)

3. Results:

GO
AGO
AGIO
AGIOS
AMIGOS
GLIOMAS
ALGORISM
ALGORISMS
GLAMORISES
ALGOMETRIES
LEGITIMATORS
DERMATOLOGIES
DERMATOLOGISTS
MELODRAMATISING
IT
AIT
ADIT
ADITS
ADMITS
DIASTEM
ADAMSITE
ACETAMIDS
ACETAMIDES
ACIDIMETERS
ACIDIMETRIES
CIRCUMSTANCED
DISCOURAGEMENT
DISCOURAGEMENTS
TI
AIT
ADIT
ADITS
ADMITS
DIASTEM
ADAMSITE
ACETAMIDS
ACETAMIDES
ACIDIMETERS
ACIDIMETRIES
CIRCUMSTANCED
DISCOURAGEMENT
DISCOURAGEMENTS