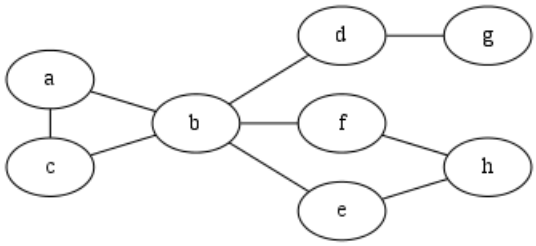
2. Graph Diameter

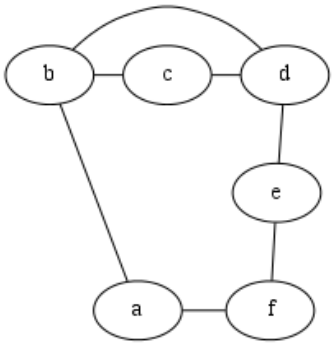
The *diameter* of a connected, undirected, unweighted graph is the largest possible value of the following quantity: the smallest number of edges on any path between two nodes. In other words, it's the largest number of steps required to get between two nodes in the graph. Your task is to design an efficient algorithm to find the diameter.

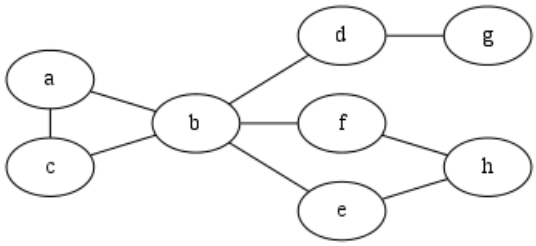
Step 1: Build intuition through examples.

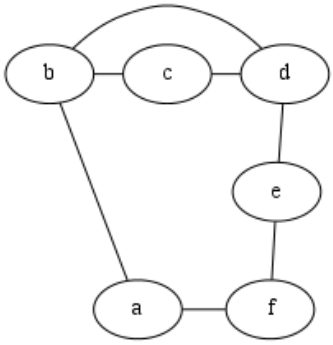
What is the diameter of the following graphs?



4







3

Step 2: Develop a formal problem specification

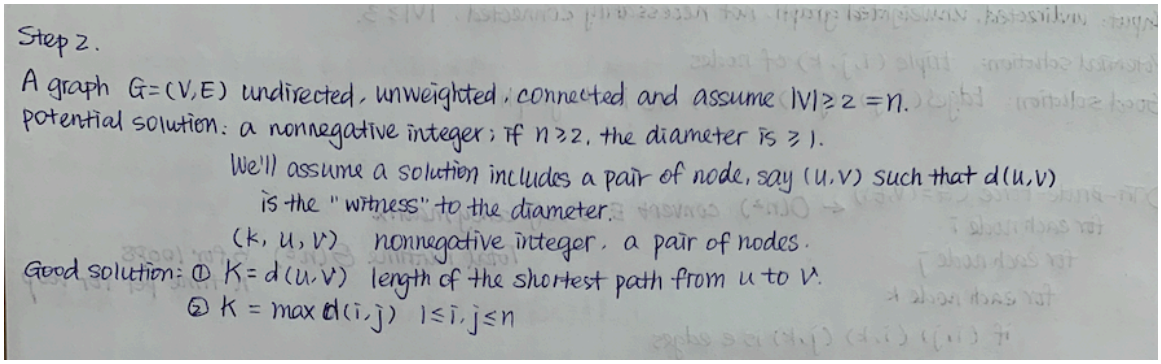
Develop notation for describing a problem instance, a potential solution, a good solution.

Describe a problem instance, a potential solution, a good solution.

empty graph $G=(\{\},\{\})$ ($\{vertex\}, \{edge\}$).
one-node graph $G=(\{x\},\{\})$
disconnected graph $G=(\{x,y,\cdots\},E)$ such that there's no path between x and y.
two-node connected graph $G=(\{x,y\},\{(x,y)\})$.

A—B—C $G=(\{A,B,C\},\{(A,B),(B,C)\})$

We describe a graph as a tuple $G=(V,E)$. V is the set of vertex, E is the set of edges.
We describe an edge as (u,v), $u,v \in V$.

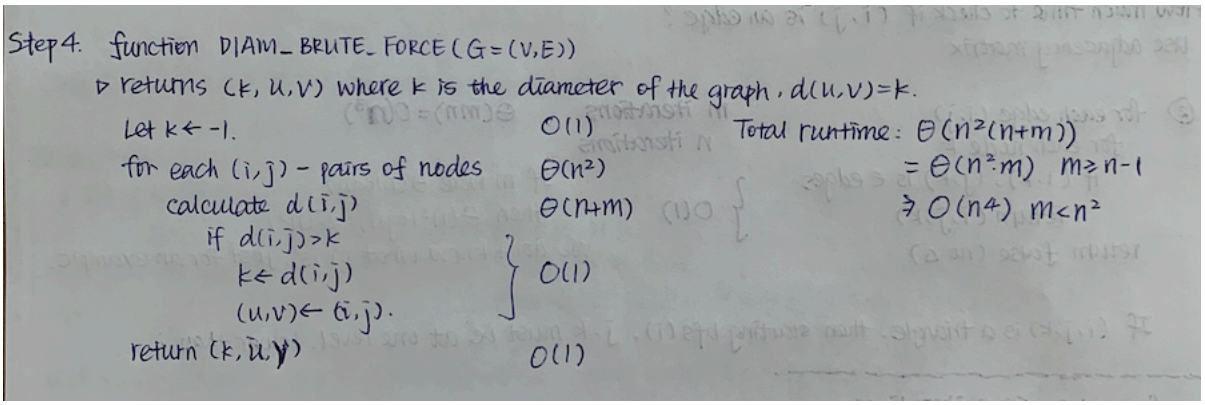


Step 3: Identify similar problems. What are the similarities?

BFS, DFS

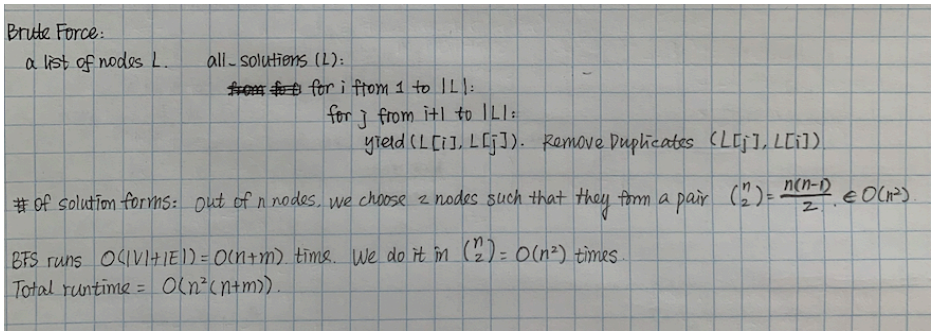
We can use BFS to calculate distance between node i and j.

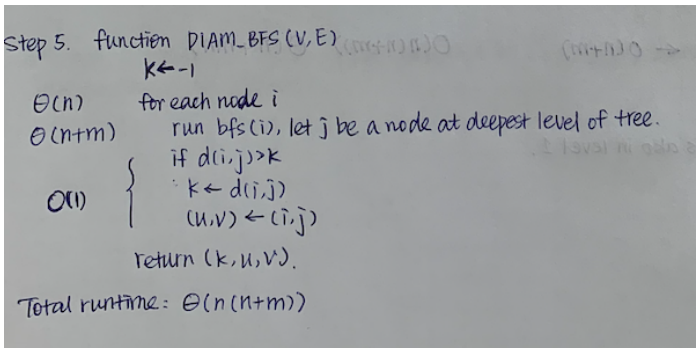
Step 4: Evaluate brute force.



Step 5: Design a better algorithm.

1. **Brainstorm some ideas, then sketch out an algorithm.**
   Try out your algorithm on some examples.





2. **Show that your algorithm is correct.**

See above

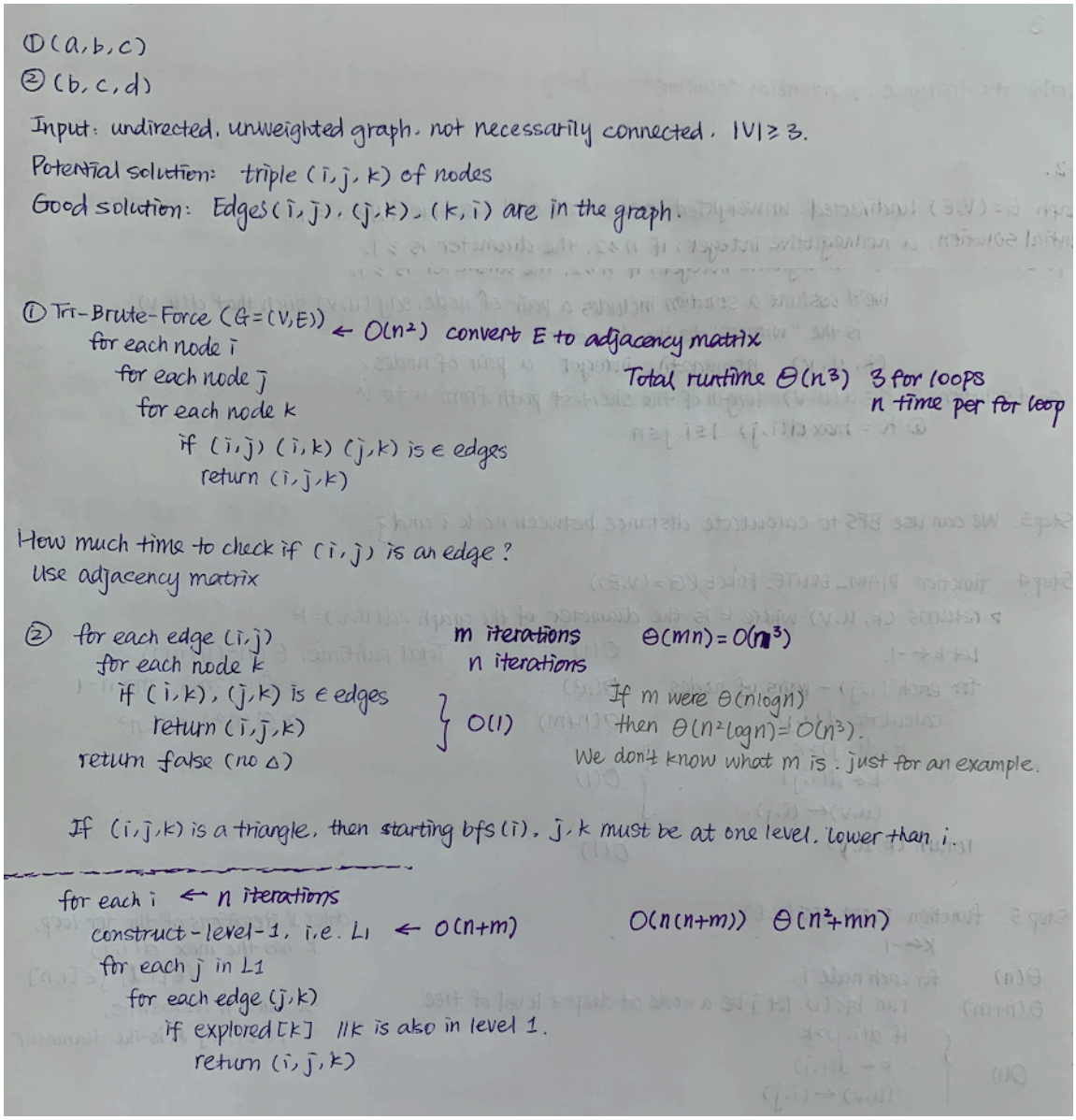
3. **Analyze the running time of your algorithm.**

See above

3. Graph Triangles

Find an efficient algorithm that takes as input an undirected graph G, and determines whether G contains a triangle. A graph $G = (V, E)$ contains a triangle if there are three distinct nodes i, j and k of V with an edge between each pair. **Here, unlike the previous parts, do not assume that G is connected**.[1]

Although we don't enumerate them here, the usual steps to design and analysis of algorithms will likely be helpful here. If you're unsure what to do, take it one step at a time.



[1] Finding triangles in a graph is an example of the more general problem of finding *motifs* in graphs. For example, there is much interest in finding motifs in biological networks, such as protein-protein interaction networks. A motif is typically described as a graph, say M, with a constant number of nodes, say numbered 1 to c. The motif appears in graph G if G has c distinct nodes, say n_1 to n_c, such that edge (n_i, n_j) is in G if and only if (i, j) is in M.