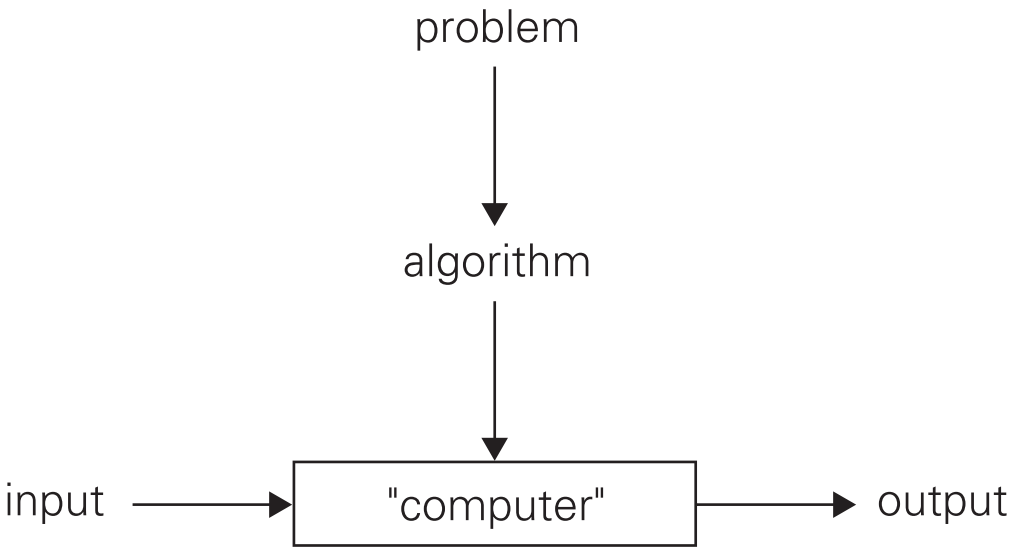
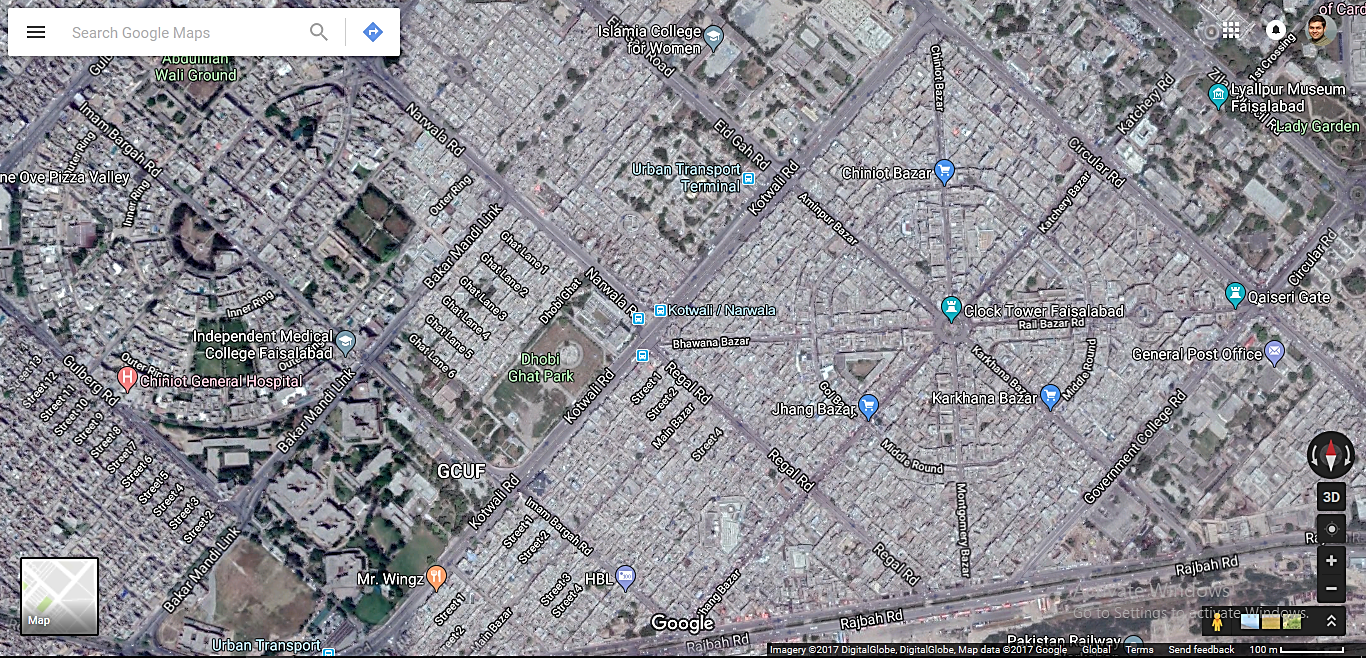
**What Is an Algorithm?**

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a ﬁnite amount of time.

* The reference to “instructions” in the deﬁnition implies that there is something or someone capable of understanding and following the instructions given.
* We call this a “computer,” “computers” are those ubiquitous electronic devices that have become indispensable in almost everything we do. 
* A simple way to define the word 'problem' is: a situation that needs solution.
* **A problem is defined by**
  + input domain e.g., all ordered pairs of positive integers.
  + output specification e.g., equivalent fraction in lowest terms
  + Method of approaching a solution e.g., Decision, Comparison, Computation etc.
* A program, on the other hand, is an algorithm that has been encoded into some programming language.

**Write down driving directions for going from GCUF to Clock Tower?**

****

1. On the picture, identify vertices and edges
2. What is the shortest path to reach destination
3. Looking at the clock tower, how many edges are connected to it ?
4. If blockage occur at kotwali road, how many other roads are available to reach destination
5. How will you give direction to a person to reach destination of he does not have Google map (intelligence of his own)

**Design an algorithm to ﬁnd all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5 and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively?**



**[Bound on: input , output]**



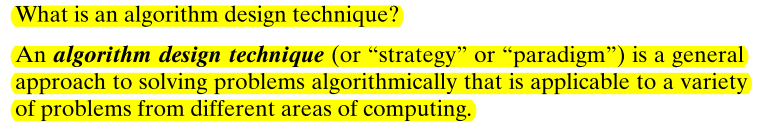




**Should you worry about the speed and amount of memory of a computer at your disposal?**









 **Pseudocode**



That is, you have to prove that the algorithm yields a required result for every legitimate input in a ﬁnite amount of time.

But in order to show that an algorithm is incorrect, you need just one instance of its input for which the algorithm fails.

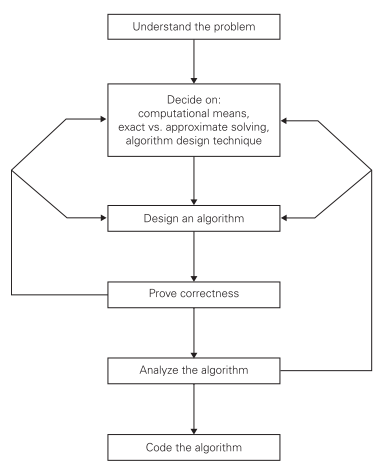
**For an approximation algorithm, we usually would like to be able to show that the error produced by the algorithm does not exceed a predeﬁned limit. Y**



**We usually want our algorithms to possess several qualities. After correctness, by far the most important is efﬁciency. In fact, there are two kinds of algorithm efﬁciency: time efﬁciency, indicating how fast the algorithm runs, and space efﬁciency, indicating how much extra memory it uses.**

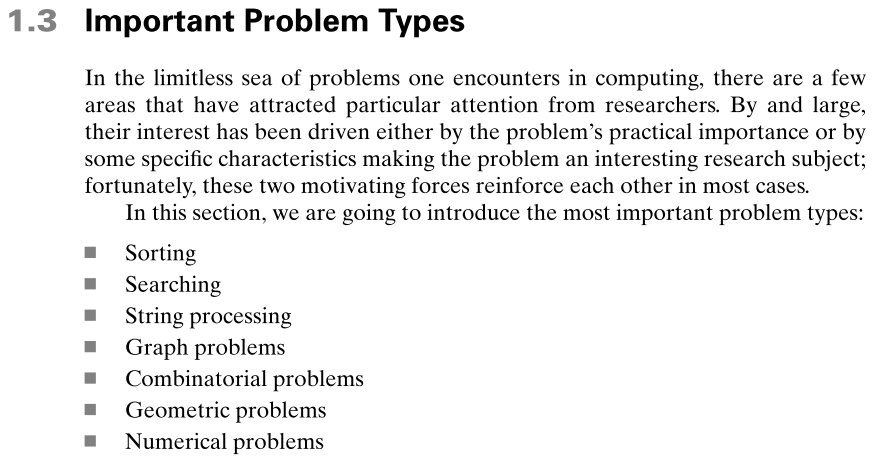


Still, you need to be aware of such standard tricks as computing a loop’s invariant (an expression that does not change its value) outside the loop, collecting common subexpressions, replacing expensive operations by cheap ones, and so on.



**Q. Old World puzzle A peasant ﬁnds himself on a riverbank with a wolf, a goat, and a head of cabbage. He needs to transport all three to the other side of the river in his boat. However, the boat has room for only the peasant himself and one other item (either the wolf, the goat, or the cabbage). In his absence, the wolf would eat the goat, and the goat would eat the cabbage. Solve this problem for the peasant or prove it has no solution. (Note: The peasant is a vegetarian but does not like cabbage and hence can eat neither the goat nor the cabbage to help him solve the problem. And it goes without saying that the wolf is a protected species.)**

**Inputs, Complexity**



**Sorting:** The sorting problem is to rearrange the items of a given list in non-decreasing order

**Searching:** The searching problem deals with ﬁnding a given value, called a search key, in a given set (or a multiset, which permits several elements to have the same value).

**Graph Problems:** One of the oldest and most interesting areas in algorithmics is graph algorithms. Informally, a graph can be thought of as a collection of points called vertices, some of which are connected by line segments called edges. Graphs are an interesting subject to study, for both theoretical and practical reasons. Graphs can be used for modeling a wide variety of applications, including transportation, communication, social and economic networks, project scheduling, and games.

**Combinatorial Problems:** From a more abstract perspective, the traveling salesman problem and the graphcoloring problem are examples of combinatorial problems. These are problems that ask, explicitly or implicitly, to ﬁnd a combinatorial object—such as a permutation, a combination, or a subset—that satisﬁes certain constraints. **A desired combinatorial object may also be required to have some additional property such as a maximum value or a minimum cost.**

**Geometric Problems:** Geometric algorithms deal with geometric objects such as points, lines, and polygons. The ancient Greeks were very much interested in developing procedures (they did not call them algorithms, of course) for solving a variety of geometric problems, including problems of constructing simple geometric shapes—triangles, circles, and so on—with an unmarked ruler and a compass. Then, for about 2000 years, intense interest in geometric algorithms disappeared, to be resurrected in the age of computers—no more rulers and compasses, just bits, bytes, and good old human ingenuity. Of course, today people are interested in geometric algorithms with quite different applications in mind, such as computer graphics, robotics, and tomography.

**Numerical Problems:** Numerical problems, another large special area of applications, are problems that involve mathematical objects of continuous nature: solving equations and systems of equations, computing deﬁnite integrals, evaluating functions, and so on. The majority of such mathematical problems can be solved only approximately.