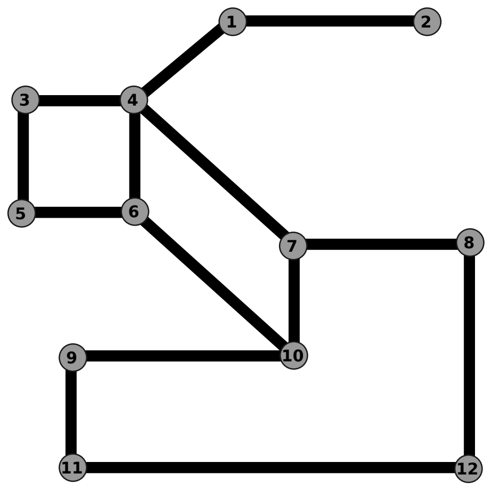
This relatively short homework is due in class on Wednesday, January 30. This is a written assignment, which should be turned in on paper. You can discuss this homework with other people in the class, but you should avoid discussing it with others. In any case, you should write up your own solutions in your own words.

1. You are in one node in an infinitely long doubly-linked list stretching off to the left and to the right. Exactly one node in the list contains a treasure, but you don't know which one. The only thing you can do is move left or right, one node at a time (and look in the current node for the treasure). Give an algorithm for finding the treasure. The algorithm must specify the sequence of left and right moves that you will make, and it must be guaranteed to find the treasure eventually. Try for an algorithm that is as efficient as you can make it. Also, say what you can about the number of moves that your algorithm makes while finding the treasure, if the treasure is N nodes away from your starting position.

2. The *vertex cover problem* is a well-known graph problem. A graph is a set of vertices and a set of edges, where each edge connects two vertices. Given a graph, the vertex cover problem asks you to find a subset of the vertices such that every edge has at least one endpoint in the subset, and the subset is as small as possible.

We can rephrase this as a "corridor guard" problem (keeping in mind that the network of corridors can be three-dimensional): Given a network of intersecting corridors, you want to post guards in selected intersections so that every corridor segment has a guard at at least one of its two endpoints. The problem is to do this using the minimum number of guards. Here is an example of such a network. The lines represent corridors (or edges) and the twelve numbered circles are their intersections (or vertices). Note, for example, that a guard in intersection number 6 can cover three corridor segments.



Note that there is a horribly inefficient "exhaustive search" algorithm that gives the correct solution: Consider all possible subsets of the set of intersections, throw out any subsets that don't cover all of the corridor segments, and then return a set of minimal size among those subsets that remain.

a) Find a reasonably efficient algorithm for selecting a subset of intersections where guards will be posted. It does not have to be guaranteed to give the correct, minimal solution—it just has to come up with some subset that seems likely to be pretty good. You should carefully state the steps in your algorithm. It should be clear how to apply your algorithm to **any** graph.

b) Show how your algorithm works for the example corridor network shown above. Don't just show the subset of intersections produced as the output of the algorithm; show how the steps specified in your algorithm produce that subset.

c) Find a counterexample that shows that your algorithm does not, in fact, give the correct solution in all cases. That is, find a corridor network (or graph) for which your algorithm produces a larger-than-necessary set of intersections. (Exhibit the network of corridors for your counterexample, the solution produced by your algorithm, and a smaller subset of intersections that covers all the intersections.)

3. (*Adapted from Exercise 2.1-3 in the textbook.*) Consider the **searching problem**:

Input: An array A of numbers and a number v.

Output: An index i such that $A[i]$ equals v, or the special value -1 if v is not in the array.

Write a pseudocode or Java algorithm for **linear search**, which scans through the array elements in order, looking for v. State a loop invariant for your algorithm, and use it to show that the algorithm is correct. You need to show that your loop invariant fulfills the three necessary properties (Initialization, Maintenance, and Termination).