1. Show the steps in finding the maximum flow in the following network:

![Network Diagram]

2. You are given a set of \(N\) sticks, which are lying on top of each other in some configuration. A stick may be picked up only if there is no stick on top of it. Explain in words and pseudo-code an algorithm to determine whether the sticks can all be picked up and to give a sequence of stick pickups that does this if possible. You may assume that you can access the sticks as in an array and that given two sticks \(A\) and \(B\) you can determine if \(A\) is on top of \(B\). Illustrate your algorithm on the following sticks: \(A, B, C, D, E, F, G\), where \(A > C\), \(G > C\), \(B > A\), \(B > G\), \(F > G\), \(F > E\), \(D > F\), \(D > E\) (where \(A > C\) means \(A\) is on top of \(C\)).

3. Consider the following greedy strategy for finding a shortest weighted path from vertex \(\text{start}\) to vertex \(\text{goal}\) in a given connected graph, then answer the question at the end.

   a. Initialize \(\text{path}\) to \(\text{start}\).
   b. Initialize \(\text{VisitedVertices}\) to \{\text{start}\}.
   c. If \(\text{start}==\text{goal}\), return \(\text{path}\) and exit. Otherwise continue.
   d. Find the edge \((\text{start}, v)\) of minimum weight such that \(v\) is adjacent to \(\text{start}\) and \(v\) is not in \(\text{VisitedVertices}\).
   e. Add \(v\) to \(\text{path}\) and add \(v\) to \(\text{VisitedVertices}\).
   f. Set \(\text{start}\) equal to \(v\) and go to step c.

   Does this greedy strategy always find a shortest path from \(\text{start}\) to \(\text{goal}\)? Either explain intuitively why it works or give a counter example (i.e., give an example of a weighted graph, a start vertex and a goal vertex, where the given algorithm does not give the shortest path, and explain briefly what goes wrong).

4. Explain how to solve the following problem using a shortest-path algorithm: The input is a two-dimensional array \(a[]\) of doubles indicating the exchange rates between currencies. So the entry \(a[i][j]\) is the amount of currency \(j\) obtained for one unit of currency \(i\). If \(i\) corresponds to the US Dollar and \(j\) corresponds to the Euro, then \(a[i][j]\) would be about .65 (one dollar buys roughly 0.65 euros) and \(a[j][i]\) would be about 1/.65 or 1.55. Determine if there is a sequence of exchanges that makes money. i.e., if \(a[0][1] = 2, a[1][2] = 2, a[2][0] = .3\), then exchanging 0's for 1's, then 1's for 2's, then 2's for 0's results in 1.2 0's for each 0 exchanged! You must describe how to set up the problem and how to solve it, including describing the algorithm you use to solve it.

5. Write a method that takes as input a binary search tree, \(T\), and two keys \(k_1\) and \(k_2\), which are ordered so that \(k_1 <= k_2\), and prints all elements \(X\) in the tree such that \(k_1 <= \text{key}(X) <= k_2\). You may assume that the keys are of type Comparable. Your program should run in time \(O(K + \log N)\), where \(K\) is the number of keys printed.

6. Write code for a class of general binary trees (not so specific as a binary search tree) and for a class of tree iterators. Now consider the following. Suppose that you are given two binary heaps \(A\) and \(B\). Assume that both are full, complete trees containing \(2^a - 1\) and \(2^b - 1\) nodes respectively.
   a. Give an \(O(\log N)\) algorithm to merge the two heaps if \(a = b\).
   b. Give an \(O(\log N)\) algorithm to merge the two heaps if \(|a - b| = 1\).
   c. Give an \(O(\log^2 N)\) algorithm to merge the two heaps regardless of \(a\) and \(b\).

7. If there’s a topic you’d prepped for but which wasn’t on this exam, give a short exposition of it with illustrative examples.