1. About complete binary tree implemented with a one-dimensional array T. Where is the left child or parent of T[i]?
   1. Parent would be T[(i – 1)/2]
   2. Left child is T[2i + 1]
   3. Right child is T[2i + 2]
2. Pre-order, in-order and post-order tree traversal. How to construct a binary tree given its in-order and post-order traversal results?
   1. Pre order: start from root and go down left subtree as far as you can, on unwind visit all right subtrees printing data as you go (root)(left subtree)(right subtree)
   2. In order: visit left most node of tree, print data, then visit right sub trees as you unwind (left subtree)(root)(right sub tree)
   3. Post order: visit each node after visiting the children (i.e., left subtree starting from last node, going from left to right. Then right sub tree does the same. Then root (left sub tree)(right sub tree)(root) (root of each sub tree is last printed out)
3. Algorithm analysis. Given a piece of code, please show me the growth rate function and the time complexity.
4. Max-heap definition and how to build a max-heap given an one-dimensional array? ( including the siftdown() procedure ), and How to add/remove a record in a max-heap?
   1. Max-heap: a complete binary tree whose nodes contain comparable objects and are organized from largest to smallest
   2. Sort from largest to smallest. The root of the heap is at index 0. Its left child is (2\*index + 1] and its right is [2\*index + 2]. For each index of array find left and right children is they exisit
   3. siftDown() : if root is not greater than both children, swap root with largest child. Follow down subtree that was modified and reapply logic until it is a leaf node or larger than both children
5. 5)Regarding Kruskal’s, Prim’s algorithm and the Dijkstra’s algorithm
   1. you should be able to explain step by step how the solution is computed with an example?
   2. The time complexity of the algorithm if using different auxiliary data structures, e.g. a regular queue vs. a priority queue.
   3. Why that algorithm works, i.e. the correctness proof we learned in the lecture notes.
6. The recursive solution of Topological sorting and its Time complexity.
7. Write a program to check whether a given directed graph is Acyclic, and then analyze the time complexity of your program.
   1. DFS will work.
   2. Also the UF structure works as we learned in the class.
8. What is greedy algorithm? And what is exhaustive search? what is divide-and-conquer? what is dynamic programming? and what is branch-and-bound?
9. How to solve the ChangeProblem using greedy algorithm, using recursive algorithm and using dynamic programming? You should be able to explain step-by-step with an example, and you are able to write the pseudo-code.
10. Apply dynamic programming in a problem.
11. Write a short Java program about a binary tree. Thinking about traversal, find a node at a particular level, total number of leaves, height of the tree, is the tree balanced or not? etc. The problem in the exam are not limited to these hints.
12. Prefix tree implementation, how to perform insertion and deletion in a prefix tree? what is its time complexity?