Starting with this homework, groups of up to three students may submit a single solution for each homework problem. Every student in the group receives the same grade.

---

1. Describe and analyze an algorithm to reconstruct a binary search tree T, given the sequence of keys visited by a postorder traversal of T (as in Quiz 0 problem 3).

   Assume that all the input keys are distinct. Don't worry about detecting invalid inputs; the input sequence is guaranteed to be the postorder traversal of some binary search tree.

2. An array $A[0..n-1]$ of n distinct numbers is ***bitonic*** if there are unique indices i and j such that $A[(i-1) \bmod n] < A[i] > A[(i+1) \bmod n]$ and $A[(j-1) \bmod n] > A[j] < A[(j+1) \bmod n]$. In other words, a bitonic sequence either consists of an increasing sequence followed by a decreasing sequence, or can be circularly shifted to become so. For example,

4	6	9	8	7	5	1	2	3

   is bitonic, but

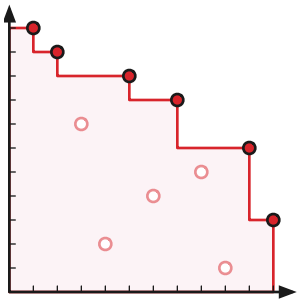
3	6	9	8	7	5	1	2	4

   is *not* bitonic.

   Describe and analyze an algorithm to find the *smallest* element in an n-element bitonic array in $O(\log n)$ time. You may assume that the numbers in the input array are distinct.

3. Let S be a set of n points in the plane. A point $p \in S$ ***maximal*** (or *Pareto-optimal*) if no point in S is both above and to the right of p. The maximal points in S intuitively define a *staircase* with all the other points in S below and to the left.

   Describe and analyze a divide-and-conquer algorithm to find all the maximal points in a given n-point set in $O(n \log n)$ time. You may assume all the input points have distinct x-coordinates and distinct y-coordinates.



Maximal points define a staircase.

*4. ***[Extra Credit]*** Describe and analyze an algorithm to find all the maximal points in a given n-point set in $\boldsymbol{O(n \log m)}$ time, where m is the number of maximal points. In particular, your algorithm should run in $O(n)$ time if the input set contains only one maximal point, and in $O(n \log n)$ time in the worst case. *[Hint: I know of at least two different ways to do this.]*