
Reminder: Add a signed statement that you have adhered to the collaboration policy for this class and that what you are presenting is your own work.

Problems

22. (Regular) You are given a set of \( n \) elements. However, the input has a large number of duplicates, and the number of unique elements in the list are only \( k \) (not a constant). Design a simple comparison-based sorting algorithm that sorts using \( o(n \log n) \) comparisons. Analyze its time complexity in terms of \( n \) and \( k \). **Hint:** Use red-black trees.

23. (Regular) Solve problem 9.3-1, page 192.

24. (Exercise) Solve problem 9.3-3, page 192. No details are needed. Couple of sentences should be sufficient.

25. (Regular) Solve 9.3-6, page 192. You may assume that \( k \) is a power of 2.


27. (Extra Credit) Solve 12.2-8, page 260.

28. (Exercise) Solve 13.3-2, page 287. Handdrawn trees are acceptable.


31. (Exercise) Solve 16.3-2, page 392.

32. (Exercise) Solve 15.4-1, page 355.

33. (Exercise) Read and understand the application described in Sections 15.1, 15.2, or 15.5.

34. (Regular) In class, we discussed greedy and dynamic programming algorithms to solve the activity selection problem. Consider the following modification to the problem. Assume that you are going to be paid a bonus of \( v_j \) dollars, if you scheduled activity \( a_i \). First show that a greedy algorithm will fail to find the optimal solution. Then design an algorithm to find a set of non-overlapping activities that would leave you with the largest possible bonus.