Algorithms
Problem Set 2
University of Virginia

Gabriel Robins

Please make all algorithms as efficient as you can, and state their time and space complexities.

1-46. Solve the following problems from the [Cormen, Third Edition, 2009] algorithms textbook:

p. 166: 6.5-9
p. 188: 7-4, 7-6
p. 197: 8.2-4
p. 200: 8.3-2, 8.3-4
p. 204: 8.4-2, 8.4-4
p. 206-207: 8-2, 8-3, 8-4
p. 215: 9.1-1, 9.1-2
p. 223-226: 9.3-1, 9.3-3, 9.3-4, 9.3-5, 9.3-6, 9.3-7, 9.3-8, 9.3-9
p. 236: 10.1-5
p. 240: 10.2-2, 10.2-3, 10.2-7, 10.2-8
p. 248: 10.4-3, 10.4-5
p. 255: 11.1-4
p. 289: 12.1-3
p. 299: 12.3-3, 12.3-4
p. 331-332: 13-1
p. 345: 12.1-8
p. 354-355: 14.3-7. 14-1
p. 397: 15.4-5, 15.4-6
p. 405-410: 15-2, 15-3, 15-5, 15-8
p. 422: 16.1-4
p. 428: 16.2-5
p. 437: 16.3-9
p. 446-447: 16-1

47. Give an algorithm that given a weighted graph, finds a spanning tree having the least possible product of its edge weights. Name a practical application of this problem.

48. True or false: if all edge weights of a graph are unique, then the MST is unique as well.

49. Give an efficient algorithm for finding the next-to-minimum spanning tree of a weighted graph.

50. The shortest path between two nodes in a weighted graph may be not unique. Give an algorithm to find a shortest path between two nodes with a minimum number of edges.

51. Prove whether there exist a data structure where the operations INSERT, DELETE, and MIN each requires O(1) worst-case time each.

52. Does there exist a data structure where add/delete/find require O(1) expected-time and O(log n) worst-case time?

53. A "probe" at a pair of nodes A and B in a tree T determines whether all edges along the path in T from A to B are "intact" (e.g., we are looking for "open faults" in an electrical circuit).

    a) What is the minimum # of probes (in terms of the # of nodes & leaves of the tree) required to completely test all edges in a given tree?

    b) Give an algorithm that finds such a minimum set of probes for an arbitrary tree.

54. We would like to make a height-balanced binary search tree **persistent**, in the following sense. At the end of an arbitrarily long mixed series of node ADD and/or DELETE operations, the state of the tree after each individual operation is still explicitly represented. After N such arbitrary ADD and/or DELETE operations are performed (starting with an empty tree), within O(1) time we can obtain a pointer to the complete tree as it was right after the i^{th} operation, for any given i. Similarly, we need to support FIND queries on each of the N past versions of the tree, without asymptotic time penalty over normal tree searches. How can such a scheme be implemented efficiently, without asymptotically slowing down the worst-case ADD and DELETE times? What is the space penalty (in terms of N) required to implement this scheme?

55. Give an efficient algorithm that given N points in the plane, determines whether any three of the points are collinear. What is the time complexity as a function of N?

56. Give an efficient algorithm that given N points in the plane, determines whether any three of the points are collinear and equally-spaced (along their containing line). What is the time complexity?

57. Give an efficient algorithm that given N points in the plane, determines a maximum (largest-cardinality) collinear subset. What is the time complexity?

58. Give an efficient algorithm that given N points in the plane, determines a maximum collinear equally-spaced subset. What is the time complexity?

59. Give an efficient algorithm that given N points in the plane, determines **all** maximal collinear subsets (a *maximal* collinear subset is one that is not properly contained in any larger collinear subset).

60. Give an **optimal** algorithm that given N points in the plane, determines all maximal collinear equally-spaced subsets. What is the time complexity? What is the time complexity as a function of N? Prove the optimality of your algorithm.