DSA Comprehensive Question Bank

Varunkumar Jayapaul

Last updated on 30^{th} November 2022

Syllabus

This question bank is based on the following syllabus and can be used to practice for the DSA compre exam

Data structures and algorithms

Common data structures: dictionaries, hashing, trees, search trees, heaps and graphs.

Program Performance: Time and space complexity, average and worst case analysis, asymptotic notation, recurrence equations and their solution.

Algorithmic techniques: Sorting algorithms – lower bound, sorting in linear time, Greedy algorithms (Huffman coding, knapsack), Divide and conquer - Master theorem, Dynamic programming (0/1 knapsack, Traveling salesman problem, matrix multiplication, all-pairs shortest paths)

Graph Algorithms: DFS and BFS, biconnectivity, spanning trees; Minimum cost spanning trees: Kruskals, Prims, and Sollins algorithms; Path finding and shortest path algorithms; Topological sorting; Matching, Network Flows; Bipartite graphs

Computational complexity: Introduction to problem classes: P, NP, NP-complete, NP-hard.

Textbooks

J. Kleinberg and E. Tardos, Algorithm Design, Pearson, 2006.
S. Sahni, Data Structures, Algorithms, and Applications in C++, Silicon Press, 2/e, 2005.

Reference books

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press, 3/e, 2009.
S. Dasgupta, C. H. Papadimitriou, U. V. Vazirani, Algorithms, McGraw-Hill, 2006.
S. S. Skiena, The Algorithm Design Manual, Springer, 2/e, 2008

1 First Portion

1. A binary tree is a rooted tree in which each node has at most two children. Show by induction that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

2. Show that if a node in a binary search tree has two children, then its successor has no left child and its predecessor has no right child.

3. We can sort a given set of n numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one) and then printing the numbers by an inorder tree walk. What are the worst-case and best-case running times for this sorting algorithm?

4. Show a way to build a max-heap from an unordered array in linear time.

5. Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.

6. Given two arrays of unordered numbers, how can we check whether both arrays have the same set of numbers in $O(n)$ time?

7. Give a nonrecursive algorithm that performs an inorder tree walk.

2 Second Portion

1. Which is asymptotically larger: $\lg(\lg^* n)$ or $\lg^*(\lg n)$?

2. Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_6$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots = \Omega(g_6)$.

   (a) $2^{\lg n}$ $\quad (\lg n)^{\lg n}$ $\quad e^n$ $\quad 4^{\lg n}$ $\quad (n+1)!$ $\quad \sqrt{\lg n}$

   (b) $(3/2)^n$ $\quad n^3$ $\quad \lg^2 n$ $\quad \lg(n!)$ $\quad (2^{2^n})$ $\quad n^{\frac{1}{\lg n}}$

3. Solve the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

   (a) $T(n) = 2T(\sqrt{n}) + \lg n$

   (b) $T(n) = T(n-2) + \frac{1}{\lg n}$

   (c) $T(n) = \sqrt{n}T(\sqrt{n}) + n$

   (d) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

   (e) $T(n) = 7T(n/3) + n^2$

4. Show that the solution to $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(n \lg n)$.

5. CLRS Chapter 5, Question 5.2 "Searching an unsorted array".

6. Suppose that we toss balls into b bins until some bin contains two balls. Each toss is independent, and each ball is equally likely to end up in any bin. What is the expected number of ball tosses?

7. Suppose that n balls are tossed into n bins, where each toss is independent and the ball is equally likely to end up in any bin. What is the expected number of empty bins? What is the expected number of bins with exactly one ball?

8. Suppose you flip a fair coin n times. What is the longest streak of consecutive heads that you expect to see?

3 Third Portion

1. Suppose that you are given a sequence of n elements to sort. The input sequence consists of n/k subsequences, each containing k elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length n is to sort the k elements in each of the n/k subsequences. Show an $\Omega(n \lg k)$ lower bound on the number of comparisons needed to solve this variant of the sorting problem.

2. Use induction to prove that radix sort works. Where does your proof need the assumption that the intermediate sort is stable?

3. Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

4. Suppose that we have a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.

5. Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

6. Describe an efficient algorithm that, given a set $\{x_1, x_2, \ldots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

7. Suppose that a data file contains a sequence of 8-bit characters such that all 256 characters are about equally common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case is no more efficient than using an ordinary 8-bit fixed-length code.

8. Consider the following closest-point heuristic for building an approximate traveling-salesman tour whose cost function satisfies the triangle inequality. Begin with a trivial cycle consisting of a single arbitrarily chosen vertex. At each step, identify the vertex u that is not on the cycle but whose distance to any vertex on the cycle is minimum. Suppose that the vertex on the cycle that is nearest u is vertex v. Extend the cycle to include u by inserting u just after v. Repeat until all vertices are on the cycle. Prove that this heuristic returns a tour whose total cost is not more than twice the cost of an optimal tour.

9. Suppose that the vertices for an instance of the traveling-salesman problem are points in the plane and that the cost $c(u, v)$ is the euclidean distance between points u and v. Show that an optimal tour never crosses itself.

10. Show how to express the single-source shortest-paths problem as a product of matrices and a vector. Describe how evaluating this product corresponds to a Bellman- Ford-like algorithm.

4 Fourth Portion

1. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose, we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.)

2. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.

3. Let (u, v) be a minimum-weight edge in a connected graph G. Show that (u, v) belongs to some minimum spanning tree of G.

4. Let T be a minimum spanning tree of a graph G, and let L be the sorted list of the edge weights of T . Show that for any other minimum spanning tree T' of G, the list L is also the sorted list of edge weights of T'.

5. Mention the Kruskal's and Prim's algorithm pseudocode. Explain their correctness and running time.

6. CLRS Chapter 24, Exercise 24-3 "Arbitrage".

7. Suppose that we are given a weighted, directed graph $G(V, E)$ in which edges that leave the source vertex s may have negative weights, all other

edge weights are nonnegative, and there are no negative-weight cycles. Argue that Dijkstra's algorithm correctly finds shortest paths from s in this graph.

8. Let $G(V, E)$ be a weighted, directed graph with nonnegative weight function $w : E \to \{0, 1, \ldots, W\}$ for some nonnegative integer W. Modify Dijkstra's algorithm to compute the shortest paths from a given source vertex s in $O(WV + E)$ time.

9. The edge connectivity of an undirected graph is the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and the edge connectivity of a cyclic chain of vertices is 2. Show how to determine the edge connectivity of an undirected graph $G(V, E)$ by running a maximum-flow algorithm on at most $|V|$ flow networks, each having $O(V)$ vertices and $O(E)$ edges.

10. Another way to perform topological sorting on a directed acyclic graph $G(V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $O(V+E)$. What happens to this algorithm if G has cycles?

11. Prove that a directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

5 Fifth Portion

1. If G is an undirected graph, a vertex cover C of G is a subset of the nodes where every edge of G touches one of the nodes in C. The vertex cover problem asks whether a graph contains a vertex cover of a specified size:

   VERTEX-COVER = $\{<G, k> | G$ is an undirected graph that has a k-node vertex cover $\}$.

   Show that VERTEX-COVER is NP-Complete.

2. A 2cnf-formula is an AND of clauses, where each clause is an OR of at most two literals. Let $2SAT = \{< \phi > | \phi$ is a satisfiable 2cnf-formula $\}$. Show that $2SAT \in P$.

3. A clique in an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E. In other words, a clique is a complete subgraph of G. The size of a clique is the number of vertices it contains. The clique problem is the optimization problem of finding a clique of maximum size in a graph. As a decision problem, we ask simply whether a clique of a given size k exists in the graph. Show that the clique problem is NP-complete.

4. Show that any language in NP can be decided by an algorithm running in time $2^{O(n^k)}$ for some constant k.

5. Prove that the class NP of languages is closed under union, intersection, concatenation, and Kleene star.

6. Show that the hamiltonian-path problem is NP-complete.

7. A subset of the nodes of a graph G is a dominating set if every other node of G is adjacent to some node in the subset. Let

   DOMINATING-SET $= \{< G, k > | G$ has a dominating set with k nodes$\}$.
   Show that it is NP-complete by giving a reduction from VERTEX-COVER.

8. Let MODEXP $= \{< a, b, c, p > | a, b, c,$ and p are positive binary integers such that $ab \equiv c (mod p)\}$. Show that MODEXP $\in P$.

9. Let RELPRIME be the problem of testing whether two numbers are relatively prime. Show that RELPRIME $\in P$.