Spring 2023

CSE 317: Design and Analysis of Algorithms

Instructions, topics, and practice questions for the Final Exam [All sections]

---

1 Instructions

1. **PLEASE READ THESE INSTRUCTIONS CAREFULLY.**
2. Use of calculators, mobile, or any other communication devices is strictly prohibited for the entirety of the exam period.
3. Please submit your devices in your bag at the front of the examination room.
4. You may keep writing material with you on your desk.
5. Read all questions carefully before you start answering.
6. Prioritize which questions to attempt first.
7. You may ask for extra sheets for your rough work, scribbling, future planning etc., **anything on extra sheet(s) will not be graded.**
8. Do as many questions as you can in the given time.
9. Please write your name and student ID on this question paper clearly.
10. Please do not use pencils or red/green/yellow pens for answers that you want to be graded.
11. Please write your answers in the provided space, anything on the extra sheet(s) will not be graded.
12. Instructors will not entertain any queries during the exam.
13. Please attach all extra sheets with your answer book.
14. Please write clearly and legibly and avoid overwriting.
15. When writing an algorithm, a clear description in English will suffice. Pseudocode is not required. Be sure to argue that your algorithm is correct (a formal proof is not needed), and analyze the asymptotic running time of your algorithm. Even if your algorithm does not meet a requested bound, you may receive partial credit for inefficient solutions that are correct.
16. Pay attention to the instructions for each question.
17. Acquisition of answers through unfair means will automatically cancel your exam.
18. Keep track of the time.
19. **Your answer(s) will not be graded if unreadable (by the instructors).**
20. **Your exam will not be graded if you don't follow these instructions.**

2 Topics

For the final exam following topics are included:

1. Asymptotic Notations
2. Recurrences
3. Divide and Conquer Algorithms
4. Greedy Algorithms
5. Dynamic Programming
6. Randomized Algorithms
7. Computational Complexity

Please note that our emphasis will be mostly on the topics covered after the midterm exam however you are expected to know the topics covered before the midterm exam as well.

3 Practice Questions

Following are some practice questions.

1. [**Analysis of Randomized Binary Search.**] Consider a variation of binary search algorithm in which instead of choosing the middle value in the current search window $[l, r]$, the algorithm select an element uniformly at random in each iteration. (i.e., instead of $m = (l + r)/2$, we choose m uniformly at random from $[l, r]$). Given that the probability of i-th value being selected as m during an execution of the algorithm is $1/|j - i + 1|$, where j is the index of target value, find the expected number of comparisons that the algorithm makes in the worst case.

   [**Hint:** *Express the number of comparisons as the sum of indicator random variables.*]

2. Prove following problems **NP**.

   (a) SUBSET-SUM: Given a set of integers S and an integer t, does there exist a subset $S' \subseteq S$ such that $\sum_{x \in S'} x = t$?

   (b) 2-PARTITION: Given a set of integers S such that $|S| = n$ and $\sum_{x \in S} x = 2B$, can S be partitioned into two subsets S_1 and S_2 such that $\sum_{x \in S_1} x = \sum_{x \in S_2} x = B$?

   (c) 3-PARTITION: Given a set of integers S such that $|S| = 3n$ and $\sum_{x \in S} x = 3B$, can S be partitioned into n subsets $S_1, S_2, \ldots, S_n$ such that $\sum_{x \in S_i} x = B$ for all i?

   (d) 3-DIMENSIONAL-MATCHING: Given three disjoint sets X, Y, and Z, each of size n, and a set $T \subseteq X \times Y \times Z$, is there a subset $T' \subseteq T$ such that $|T'| = n$ and no two elements of T' agree in any coordinate?

   (e) VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and every edge in E is incident on at least one vertex in V'?

   (f) HAMILTONIANCYCLE: Given an undirected graph $G = (V, E)$, the *hamiltonian cycle* problem asks whether there is a cycle that visits every vertex exactly once.

3. Consider the following algorithm that takes as input a positive integer n and outputs a random prime number in the range $[2, n]$. Compute the expected running time of this algorithm.

   Algorithm: RANDOM-PRIME

   Input: A positive integer $n \geq 2$.

   Output: A random prime number in the range $[2, n]$.

   1. Let $P = \{2, 3, 5, 7, 11, \ldots\}$ *// P is the set of all prime numbers*
   2. Let $S = \{2, 3, 4, \ldots, n\}$.
   3. **while** $S \neq \emptyset$
   4. $\quad x = \text{RANDOM}(S)$ *// pick a random element x from S*
   5. $\quad$ **if** $x \in P$ **then return** x
   6. $\quad$ **else** remove x from S
   7. **return** "failure".

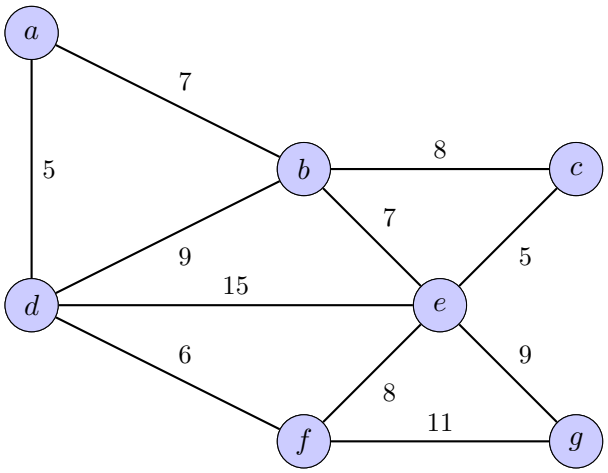
4. Let $A[1..n]$ be an array of n distinct positive numbers and $k \leq n$ be some positive integer. Give a randomized algorithm that return a random sample of k elements from A. The algorithm should run in $O(n)$ time. Prove that your algorithm is correct and analyze its running time.

5. Let $A[1..n]$ be an array of n numbers such that not all numbers are unique. We say that $A[i]$ is a *majority element* if it occurs at least $\lceil n/2 \rceil$ times in A. Give a Monte Carlo randomized algorithm that finds a majority element in A. Prove that your algorithm is correct and analyze its running time.

6. Let $A[1..n_1]$, $B[1..n_2]$, and $C[1..n_3]$ be three arrays of numbers of sizes n_1, n_2, and n_3, respectively. All these arrays are sorted in ascending order and individually they contain unique elements, however they may share elements with other arrays i.e., intersection of any of these two arrays may not be empty. Give a *greedy algorithm* that merges these three arrays in descending order such that it minimizes the total number of comparisons required to merge. Prove that your algorithm is correct and analyze its running time.

   [**Fact:** You may use the MERGE algorithm from the MERGESORT that merges two sorted arrays of sizes m and n and uses $m + n - 1$ comparisons.]

7. Apply Prim's algorithm on the following graph starting from vertex a and show the order in which the edges are added to the minimum spanning tree. Also show the total weight of the minimum spanning tree.



8. Let $G = (V, E)$ be a directed graph with n vertices and m edges. Let s be a vertex in V and $w : E \to \mathbb{R}$ be a weight function on the edges. Let $d[v]$ be the length of the shortest path from s to v for all $v \in V$. Let $d[v] = \infty$ if there is no path from s to v. Let $d[s] = 0$. Let $p[v]$ be the predecessor of v in the shortest path from s to v for all $v \in V$. Let $p[v] =$ "NIL" if there is no path from s to v. Let $p[s] =$ "NIL". Let Q be a priority queue that stores the vertices of V and supports the following operations:

   - INSERT(Q, v): Inserts vertex v into Q.
   - DECREASE-KEY(Q, v, k): Decreases the key of vertex v to k.
   - EXTRACT-MIN(Q): Removes and returns the vertex with the minimum key from Q.
   - EMPTY(Q): Returns "true" if Q is empty, otherwise returns "false".
   - CONTAINS(Q, v): Returns "true" if Q contains vertex v, otherwise returns "false".
   - SIZE(Q): Returns the number of vertices in Q.
   - KEYS(Q): Returns the keys of all vertices in Q.
   - VERTICES(Q): Returns all vertices in Q.
   - BUILD-HEAP(Q): Builds a heap from the vertices in Q.
   - HEAPIFY(Q, i): Heapifies the subtree rooted at index i.
   - PARENT(i): Returns the index of the parent of the vertex at index i.
   - LEFT(i): Returns the index of the left child of the vertex at index i.
   - RIGHT(i): Returns the index of the right child of the vertex at index i.
   - SWAP(Q, i, j): Swaps the vertices at indices i and j.

   Give an algorithm that computes $d[v]$ and $p[v]$ for all $v \in V$ using the above operations. Prove that your algorithm is correct and analyze its running time.