AAD Question Bank - 1

Part II

August 2025

Introduction

This problem set is ungraded and meant purely for practice. The problems have been shuffled and do not include topic tags, so that you can approach them without a bias. If you encounter any doubts, feel free to reach out during office hours. In case of typos or unclear statements, please let us know.
In the exam, you might be asked to "give an algorithm" to solve a problem. Your write-up should take the form of a short essay. Start by defining the problem you are solving and stating what your results are. Then provide:

- A description of the algorithm in English and, if helpful, pseudo-code.
- A proof (or proof sketch) for the correctness of the algorithm.
- An analysis of the running time.

We will give full credit only for correct solutions that are described clearly.

Problem 1

Prove the correctness of Kruskal's algorithm using the cut property.

Problem 2

Prove correctness of Prim's algorithm using the greedy-choice property.

Problem 3

Prove that Dijkstra's algorithm is a greedy algorithm. State clearly the greedy-choice property and optimal substructure it uses.

Problem 4

Define the maximum-spacing k-clustering problem. Show how Kruskal's algorithm can be adapted to solve it.

Problem 5

A directed graph $G = (V, E)$ is strongly connected if and only if every pair of vertices is strongly connected.

(1) Give an algorithm that computes all strongly connected components of a directed graph and runs in time at most $O(|V| \cdot |E|)$.

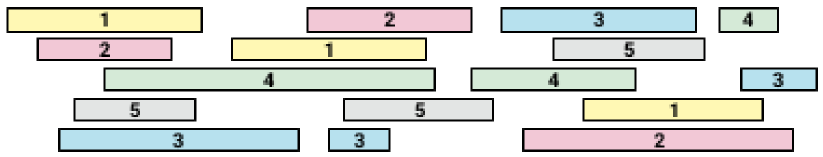
(2) Can this be improved to $O(|V| + |E|)$ by considering G^{rev} (the graph obtained by reversing all edge directions)? Explain your answer.

Problem 6

In a city there are N houses, each of which is in need of a water supply. It costs w_i dollars to build a well at house i, and it costs $c_{i,j}$ to build a pipe between houses i and j.
A house can receive water if either a well is built there or there exists some path of pipes to a house with a well.
Design an algorithm to find the minimum amount of money needed to supply every house with water. Proof of correctness is not required.

Problem 7

Let X be a set of n intervals on the real line. A *proper coloring* of X assigns a color to each interval so that any two overlapping intervals are assigned different colors.
Describe and analyze an efficient algorithm to compute the minimum number of colors needed to properly color X. Assume the input consists of two arrays $L[1..n]$ and $R[1..n]$, representing the left and right endpoints of the intervals in X.



A proper coloring of a set of intervals using five colors.

Problem 8

In a large university, there are many guest lectures scheduled throughout the day. Each lecture has a start time and finish time, and no two lectures can be held in the same hall at the same time.

(a) The university administration wants to maximize the number of lectures that can be held without overlap. Design a greedy algorithm that selects the maximum number of lectures and prove its correctness.

(b) A junior scheduler suggests a different greedy strategy: always choose the lecture with the shortest duration (smallest difference between finish and start time). Give a counterexample to show that this approach does not always yield an optimal schedule.

(c) Suppose that due to limited staff, the administration decides that at most k lectures can be scheduled for the day. Does the "earliest finish time" greedy algorithm still guarantee an optimal solution? Justify your answer.

Problem 9

An airline maintenance facility has a list of n planes that each require a maintenance job. Each job takes exactly 1 unit of time, and each job has a deadline by which it should be completed.

(a) Maximizing On-Time Jobs: Design a greedy algorithm to maximize the number of jobs completed before their deadlines. Prove correctness.

(b) Minimizing Maximum Lateness: Suppose the facility instead wants to minimize the maximum lateness across all jobs (lateness = completion time − deadline). Prove that scheduling jobs in order of Earliest Due Date (EDD) achieves an optimal schedule.

(c) Limitations of EDD: Now, suppose each job has a penalty if it is not completed by its deadline. Is Earliest Due Date still optimal when minimizing the total penalty? Give a counterexample and explain why not.

Problem 10

A tech company wants to schedule n computational tasks on a single server. Task i has processing time p_i and importance weight w_i.
The completion cost of a schedule is the weighted sum of completion times:

$$\text{Total Cost} = \sum_{i=1}^{n} w_i \cdot C_i$$

where C_i is the completion time of task i.

(a) Design a greedy algorithm that minimizes the total weighted completion time.

(b) Prove the correctness of your algorithm (using an exchange argument or greedy-choice property).

(c) Discuss whether the algorithm would still be optimal if tasks had release times (i.e., could only start after a given time).

Problem 11

A government agency is tasked with designing and maintaining a nationwide communication network connecting n cities using fiber cables. Each possible cable (u, v) has a distinct installation cost $c(u, v)$. To ensure connectivity at minimum cost, the engineers use a Minimum Spanning Tree (MST) as the backbone of the network.
While building and maintaining the network, they encounter several challenges:

(a) Since all edge weights are distinct, prove that the MST for the network is unique.

(b) The agency frequently receives requests to verify if a specific cable (x, y) must belong to some MST of the network. Design an efficient method to test this without reconstructing the MST from scratch.

(c) Suppose a new cable (u, v) with cost $c(u, v)$ is added after the MST T has already been built. Describe how to update the MST efficiently to account for this new cable. Prove correctness and analyze the runtime of your algorithm.

Problem 12

You are given a directed graph G where every edge has a negative weight, and a source vertex s. Your task is to find the shortest distances from s to every other vertex in G. Your algorithm should correctly handle the following cases:

(a) If a vertex t cannot be reached from s, report the distance $\text{dist}(t)$ as ∞.

(b) If there exists a cycle that is reachable from s, and vertex t can be reached from this cycle, then the shortest-path distance from s to t is not well-defined because there are paths with arbitrarily large negative lengths. In this case, report the distance $\text{dist}(t)$ as $-\infty$.

(c) If neither of the previous conditions applies, compute and report the correct shortest-path distance from s to t.

Problem 13

You are given a list of n jobs $J_1, J_2, \ldots, J_n$ with processing times $p_1, p_2, \ldots, p_n$ and weights $w_1, w_2, \ldots, w_n$. Starting from $t = 0$, the cost for completing a job J_i is defined as

$$\text{Cost}(J_i) = w_i \times (\text{total time from } t = 0 \text{ to the completion of job } J_i).$$

(a) Design an algorithm to determine the order in which to perform the jobs so that the total cost

$$\sum_{i=1}^{n} w_i \cdot C_i$$

(where C_i is the completion time of job J_i) is minimized.

(b) Prove the correctness of your algorithm.

(c) Analyze the time complexity of your algorithm.

Problem 14

Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E')$ be a spanning tree of G; we define the *bottleneck edge* of T to be the edge of T with the greatest cost.
A spanning tree T of G is a *minimum-bottleneck spanning tree* (MBST) if there is no spanning tree T' of G with a strictly cheaper bottleneck edge.

(a) Is every minimum-bottleneck tree of G a minimum spanning tree (MST) of G? Prove your answer or give a counterexample.

(b) Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove your answer or give a counterexample.

Problem 15

In a country named Valoria, there are n cities numbered from 1 to n. City 1 is the capital of Valoria. There are m undirected roads; the i-th road connects cities a_i and b_i and has length x_i. Additionally there are k train routes; the i-th train route connects the capital (city 1) and city s_i (bidirectional) and has length y_i.
It is guaranteed that every city can reach the capital via some sequence of roads and/or train routes. There may be multiple roads between the same pair of cities and there may be multiple train routes to the same city.
You (the president) want to close as many train routes as possible while preserving the following invariant:

$$\text{For every city } v, \text{ the length of the shortest path from } v \text{ to the capital remains unchanged.}$$

(a) Formally state the decision or optimization problem derived from the description above.

(b) Design an algorithm that computes the maximum number of train routes that can be closed while maintaining the invariant. Describe the algorithm precisely.

(c) Prove the correctness of your algorithm.

(d) Analyze the time and space complexity of your algorithm in terms of n, m, and k.

(e) Describe how to output which specific train routes should be kept (and which may be closed) when multiple optimal choices exist.

Problem 16

Suppose we are given a set U of objects labeled $p_1, p_2, \ldots, p_n$. For each pair p_i and p_j we have a numerical distance $d(p_i, p_j)$ satisfying

$$d(p_i, p_i) = 0, \qquad d(p_i, p_j) = d(p_j, p_i) > 0 \ \text{ for } i \neq j.$$

For a given parameter $k \geq 1$, a *k-clustering* of U is a partition of U into k nonempty sets $C_1, \ldots, C_k$. The *spacing* of a k-clustering is the minimum distance between any pair of points that lie in different clusters:

$$\text{Spacing}(C_1, \ldots, C_k) \;=\; \min_{1 \leq u \neq v \leq k} \min\{\, d(p, p') \mid p \in C_u,\ p' \in C_v \,\}.$$

We seek the k-clustering with maximum possible spacing; i.e. find a partition of U into k nonempty sets that maximizes the spacing.

(a) Formally state the optimization problem (input, output, objective) for maximum-spacing k-clustering.

(b) Give an efficient algorithm to compute a k-clustering that maximizes the spacing. Clearly state any assumptions you make about how the distances are provided as input (for example, as an explicit list of all $\binom{n}{2}$ pairwise distances, or as an edge-weighted graph).

(c) Prove the correctness of your algorithm (i.e., show it always returns a k-clustering with maximum spacing).

(d) Analyze the time and space complexity of your algorithm in terms of n and the number of input distance entries m (or in terms of n if the input is the full distance matrix).

(e) Explain how your algorithm and correctness argument change (if at all) when the distances additionally satisfy the triangle inequality.

Problem 17

A national power grid company wants to connect n cities with electrical cables. Each possible cable between two cities has an installation cost, which is distinct for every cable. The goal is to ensure that all cities are connected while minimizing the total installation cost.

(a) Formally model this problem as a graph problem. Precisely state what the cities and cables correspond to in graph-theoretic terms, and define the objective in terms of the graph.

(b) Design an algorithm to select a set of cables such that (i) every city is connected (the resulting graph is connected) and (ii) the total installation cost is minimized. Write the algorithm clearly (you may present Kruskal's or Prim's algorithm or another correct method), and give pseudocode.

(c) Prove that your algorithm always produces a minimum-cost network. In your proof, clearly explain the greedy-choice property and the cut (or cycle) property you use, or give an exchange argument showing optimality.

(d) Suppose after the network (an MST) has been built, a new cable (edge) $e = (u, v)$ with cost $c(e)$ is proposed. Explain how to check efficiently whether including e could reduce the total installation cost, and if so, how to update the existing network to a new minimum-cost network. Analyze the runtime of your update procedure.

(e) Given that all edge weights are distinct, prove that the minimum spanning tree of the network is unique.

Problem 18

Suppose you're a consultant for the networking company CluNet, and they have the following problem. The network that they're currently working on is modeled by a connected graph $G = (V, E)$ with n nodes. Each edge e is a fiber-optic cable that is owned by one of two companies—creatively named X and Y—and leased to CluNet.
Their plan is to choose a spanning tree T of G and upgrade the links corresponding to the edges of T. Their business relations people have already concluded an agreement with companies X and Y stipulating a number k so that in the tree T that is chosen, k of the edges will be owned by X and $n - k - 1$ of the edges will be owned by Y.
CluNet management now faces the following problem. It is not at all clear to them whether there even exists a spanning tree T meeting these conditions, or how to find one if it exists. So this is the problem they put to you:
Give a polynomial-time algorithm that takes G, with each edge labeled X or Y, and either

(a) returns a spanning tree with exactly k edges labeled X, or

(b) reports correctly that no such tree exists.

Problem 19

You are given a string s consisting of lowercase English letters.
We want to partition s into a maximum number of non-empty substrings that satisfy the following conditions:

- **Non-overlapping:** The substrings do not overlap. Formally, for any two substrings $s[i..j]$ and $s[x..y]$, either $j < x$ or $i > y$.
- **Character completeness:** If a substring contains a certain character c, then it must contain *all* occurrences of c in the original string.

Find the maximum number of substrings satisfying the above conditions.
If multiple partitions achieve the same number of substrings, return the one with the minimum total length of all substrings. It can be shown that such a solution is unique.

Problem 20

Let $G = (V, E)$ be a connected graph on n vertices and m edges such that their edge weights are all distinct. Algorithm 1 presents a different algorithm than what we studied in our classes. Your task is to:

(a) analyse the running time, and

(b) prove the correctness of this algorithm.

Algorithm 1: Borůvka's Algorithm for Minimum Spanning Tree
Input: A weighted connected graph $G = (V, E)$ with unique edge weights.
Output: The minimum spanning tree T for G.

```
1. Let $T$ be a subgraph of $G$ initially containing just the vertices in $V$ (with no edges, just isolated vertices).
2. while $T$ has fewer than $|V| - 1$ edges do
3.    for each connected component $C_i$ of $T$ do
4.         Let $e = (u, v)$ be the smallest-weight edge in $E$ with $u \in C_i$ and $v \notin C_i$.
5.         Add $e$ to $T$ (unless $e$ is already in $T$).
6. return $T$.
```