National Technical University of Athens

School of Electrical and Computer Engineering

Fundamentals of Computer Science, 2025–26

2nd Exercise Set

(algorithmic techniques – graph algorithms – numerical algorithms)

Exercise 1. (Minimum Number of Refueling Stops)

We are given a directed graph with n vertices (the cities of a country) and m edges (roads). Not all cities are necessarily connected by roads. Between two cities, roads may exist in both directions, but this is not required. Edge weights represent the distances between cities (positive). A car starts at city s and wants to reach city t. The car has fuel autonomy k kilometers. Each city has exactly one gas station. Since refueling causes a significant delay, we want to find the minimum number of refueling stops required to reach the destination, and the corresponding route.

1. Give the most efficient possible algorithm for this problem. Justify correctness and compute its complexity.
2. Note that we do *not* want the route of minimum total distance, but the route with the *fewest refueling stops*. Give an example where these differ.
3. Suppose we additionally want, among all routes with the minimum number of stops, the one with minimum total distance. Give the most efficient possible algorithm, justify correctness, and compute its complexity.

Exercise 2. (Second Minimum Spanning Tree)

We consider an undirected connected graph $G(V, E, w)$ with positive edge weights, assuming $|E| \geq |V|$ and that all weights are distinct. Let T be the set of all spanning trees of G, and let $t \in T$ be a minimum spanning tree (MST). The Second Minimum Spanning Tree (2nd-MST) of G is a spanning tree $t' \in T$ such that

$$w(t') = \min_{t'' \in T \setminus \{t\}} w(t'').$$

In other words, t' has weight greater than or equal to $w(t)$ and less than or equal to the weight of every other spanning tree.

1. Show that the MST of G is unique, and that this does not necessarily hold for the 2nd-MST.
2. Let t be the MST and t' the 2nd-MST. Show that t and t' differ by exactly one edge; i.e., there exist edges $e \in t$, $e' \notin t$, such that $t' = t \cup \{e'\} \setminus \{e\}$.
3. Let t be a spanning tree, and for any pair of vertices $u, v \in V$, let e_{uv}^{max} denote the maximum-weight edge along the unique u–v path in t. Give an $O(|V|^2)$ algorithm that computes e_{uv}^{max} for all pairs u, v.
4. Give an efficient algorithm that computes the 2nd-MST of G. Explain correctness and analyze its complexity.

Exercise 3. (Shortest Path With the Fewest Edges)
We consider a directed graph $G(V, E, w)$ with n vertices, m edges, and positive weights. Given a starting vertex s, design an efficient algorithm that, for all $u \in V$, computes a shortest s–u path that uses the *fewest edges among all shortest* s–u paths. What is the running time?

Exercise 4. (Distance Verification)
We consider a directed graph $G(V, E, \ell)$ with possibly negative edge lengths $\ell(e)$. We are given numbers $\delta_1, \ldots, \delta_n$, where each δ_k is supposed to equal $d(v_1, v_k)$, the distance from v_1 to v_k. Give an algorithm running in $\Theta(n + m)$ time that checks whether $\delta_k = d(v_1, v_k)$ for all k. If so, the algorithm must also return a shortest-path tree rooted at v_1 (without exceeding $\Theta(n + m)$ time).

Exercise 5. (Divide and Conquer Algorithms)
[exercise from the book *Algorithms* by Dasgupta, Papadimitriou, Vazirani]

A sequence $A[1], \ldots, A[n]$ is said to have a *majority element* if more than half its entries are identical. You may only test equality $A[i] = A[j]$.

1. Show how to solve this in $O(n \log n)$ time. (Hint: Divide A into A_1, A_2 of equal size. Do their majority elements help?)
2. Can you find a linear-time algorithm? (Hint: pair elements arbitrarily, discard unequal pairs, keep one from equal pairs. Show at most $n/2$ remain, and they contain a majority element if A had one.)

Exercise 6. (Repeated Squaring – Primality Tests)
(a) Write a program that tests primality using Fermat's test: If n is prime then for all a with $1 < a < n - 1$

$$a^{n-1} \bmod n = 1.$$

Explain how to detect compositeness and how repeated testing reduces error. Your program must handle numbers of thousands of digits. Test it on:

$$67280421310721, \quad 170141183460469231731687303715884105721, \quad 2^{2281} - 1.$$

(b) Some composite numbers (Carmichael numbers) pass Fermat's test for all a coprime to n. Test your code on large Carmichael numbers. What do you observe?
(c) Implement the Miller–Rabin test. Test it on Carmichael numbers. Do you observe anything strange? Explain.
(d) Write a program that finds all Mersenne primes $2^x - 1$ for $1 < x < 200$. Compare with `https://www.mersenne.org/primes/`.

Exercise 7. (Binary GCD)
Consider the following algorithm for the greatest common divisor (gcd), known as Binary GCD.

`bgcd`(a, b) (assume $a, b > 0$):

- If $a = b$, return a
- If a, b even, return $2 \cdot \mathsf{bgcd}(a/2, b/2)$
- If a even and b odd, return $\mathsf{bgcd}(a/2, b)$ (and vice versa)
- If a, b odd, return $\mathsf{bgcd}(\min(a, b), |a - b|/2)$

(a) Prove correctness. (b) What is its complexity and why? (c) Implement it and compare its performance with Euclid's algorithm on at least 10 pairs of very large numbers.

Exercise 8. (Dynamic Programming)

A frog starts on lily pad 1 and wants to reach lily pad n. Its jump from i is given by Jump$[i]$, meaning it may jump to any j with $i < j \leq i + \text{Jump}[i]$. Assume Jump$[i] \geq 1$ for all i.

Goal: find the minimum number of jumps from 1 to n.

(a) Give a counterexample showing that greedily taking the maximum possible jump is not optimal. (b) Define the DP subproblem. (c) Give the recurrence. (d) Provide pseudocode.

Submission instructions. Answers must be submitted by December 28, 2025 at 23:59, electronically on Helios (try to keep total size <5MB). Accepted formats: pdf, png, jpg, gif, zip/gz containing these.

Email submissions will not be graded.

Work must be strictly individual and include references to all sources used.

You must also briefly present your solutions orally at a later date (to be announced). For questions: `focs@corelab.ntua.gr`.