Fundamentals in Computer Science
Exercise Sheet – TD 3 (4 heures)

ENS DE LYON

Fall 2024
Prof. Thomassé

Teaching assistant: Malory Marin (LIP, MC2), malory.marin@ens-lyon.fr.
Some exercises are based on the book *A guide to algorithm design*.

Algorithm design (2h)

The goal of this set of exercises is to explore the different classical strategies to design an algorithm.

Exercise 1. ***Greedy algorithm : Interval cover***

We are given a set $X = \{x_1, ..., x_n\}$ of n points on a line.

(a) Design and implement a greedy algorithm that determines the smallest set of closed intervals of length 1 that contains all the points.

(b) Prove the optimality of the algorithm and give its complexity.

Exercise 2. ***Divide-and-Conquer : Maximum sum***

Let T be a table of n relative integers. We want to find the maximum sum of contiguous elements, namely, two indices i and j ($1 \leqslant i < j \leqslant n$) that maximize $\sum_{k=i}^{j} T[k]$.

(a) If $T = [2, 18, -22, 20, 8, -6, 10, -24, 13, 3]$, can you return the two indices corresponding to the maximum sum ?

(b) Design and implement an algorithm that returns the maximum sum of contiguous elements with a divide-and-conquer algorithm.

(c) (*Pour aller plus loin*) Design a linear-time algorithm that solves the problem through a single scan of the array. Illustrate the efficiency difference between this algorithm and the previous one by plotting both execution times versus the size of the array on random instances.

Exercise 3. ***Dynamic programming : Matrix chains***

Consider n matrices $A_1, ..., A_n$, where A_i is of size $p_{i-1} \times p_i$ ($1 \leqslant i \leqslant n$). We want to compute $A_1 \times A_2 \times ... \times A_n$. The problem is to find the order of the multiplications should be done and, therefore, to add parentheses to the expression, in order to minimize the total number of operations. Note that it costs $p_a \cdot p_b \cdot p_c$ to multiply a matrix of size $p_a \times p_b$ by a matrix of size $p_b \times p_c$.

Propose a dynamic-programming algorithm to solve the problem and give its complexity.

NP-complete problems (2h)

As discussed in the lecture, an NP-complete problem is one for which no polynomial-time algorithm is known to exist (unless P = NP). One of the most prominent NP-complete problems is 3-SAT, which involves a Boolean formula expressed in conjunctive normal form, such as

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge \dots$$

The objective is to determine a truth assignment for the variables such that the entire formula evaluates to true.

When dealing with a computational problem where no polynomial-time solution seems to exist, it's quite likely that the problem is NP-complete. To prevent the possibility of someone finding a polynomial-time algorithm in the future, it's advisable to formally prove that the problem is indeed NP-complete.

Exercise 4. ***Around clique and clusterings***

(a) Prove that the following decision problem, called CLIQUE, is NP-complete: Given a graph G and an integer k, does G have a *clique* of size k? That is, a set of k distinct vertices $u_1, \dots, u_k$ such that all the edges $u_i u_j$ (for $1 \leqslant i < j \leqslant k$) are present in G. *Hint:* Use a reduction from 3-SAT.

(b) Prove that the following decision problem, called DOMINATING SET, is NP-complete: Given a graph G and an integer k, does G have a *dominating set* of size k? That is, a set K of k vertices such that for every vertex $u \in V(G)$, either $u \in K$ or there exists some $v \in K$ such that $uv \in E(G)$. *Hint:* Use a reduction from VERTEX COVER, and add one vertex per edge in the instance of VERTEX COVER.

VERTEX COVER
Input: A graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, and an integer k. In addition, each vertex from G is incident to at least one edge.
Goal: Find a subset $C \subseteq V$ such that $|C| \leq k$ and every edge in E is incident to at least one vertex in C. In other words,

$$\forall uv \in E, \quad u \in C \text{ or } v \in C.$$

(c) Let $G = (V, E)$ be a complete graph whose edges are weighted by a weight function w that satisfies the triangle inequality: $w(u, v) \leqslant w(u, x) + w(x, v)$ for any u, v, and x in V. For instance, G might represent n points in the plane, where the weight of an edge corresponds to the distance between the two points. Let k be a positive integer. For any subset $S \subseteq V$ and any vertex $v \notin S$, we define $d(v, S) = \min_{s \in S} w(v, s)$. A *k-center* is a subset S of V with cardinality at most k such that the *center*(S) $= \max_{v \in V \setminus S} d(v, S)$ is minimized. Prove that the problem k-CENTER, which asks whether, given a real c, there exists a *k-center* S such that *center*(S) $\leqslant c$, is NP-complete. *Hint:* Use a reduction from DOMINATING SET.

Exercise 5. ***Minimizing harmful interactions in biological networks***

(a) In a biological network (e.g., a gene regulation network or protein interaction network), cycles can represent harmful feedback loops that disrupt the normal functioning of the system. To stabilize the network, biologists often need to remove a minimal set of nodes (genes or proteins) to break all the cycles.

This problem can be formulated as the **Feedback Vertex Set Problem**:

- Let the biological network be modeled as a directed or undirected graph $G = (V, E)$, where V represents the set of nodes (genes or proteins), and E represents interactions between them.
- A **feedback vertex set** is a subset $F \subseteq V$ of vertices whose removal makes the graph acyclic (i.e., without cycles).

Show that finding the minimum feedback vertex set is NP-complete by reducing it from the Vertex Cover problem in an undirected graph.

(b) Many biological problems, such as drug target identification, can be framed similarly to the MFVS problem. In the context of drug discovery, explain how the identification of a small set of critical proteins (nodes) using MFVS could guide the development of treatments to control diseases.

Exercise 6. ***Optimizing diagnostic tests***

Consider a medical diagnosis problem where a person must select the minimum number of diagnostic tests to confirm or rule out a set of possible diseases. The problem can be represented as follows:

- Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of diseases, and $T = \{t_1, t_2, \ldots, t_m\}$ be a set of diagnostic tests.
- Each test $t_i \in T$ can confirm or eliminate one or more diseases in D.

The problem can be reformulated as the Set Cover problem, where the universe is the set of diseases, and each set corresponds to the diseases that can be confirmed or eliminated by a single test. This problem is known to be NP-complete.

Set Cover
Input: A universe $U = \{u_1, u_2, \ldots, u_n\}$ of elements, a collection of sets $S = \{S_1, S_2, \ldots, S_m\}$ such that $S_i \subseteq U$ for all $i = 1, 2, \ldots, m$ and an integer k.
Goal: Find a subcollection $C \subseteq S$ such that $|C| \leqslant k$ and the union of the sets in C covers all elements of U, i.e.,

$$\bigcup_{S_i \in C} S_i = U.$$

Given that solving this problem exactly is computationally intractable (NP-complete), explain how to approach this problem using a greedy algorithm. Analyze the worst-case performance of this greedy approach in terms of approximation ratio compared to the optimal solution (*Hint :* Prove that if an optimal solution uses k tests, the greedy algorithm takes at most $O(k \ln(n))$ tests).