# PROJET - 2025 -
# SITE WEB DE CODES CIRCULAIRES

PROFESSEUR CHRISTIAN J. MICHEL

*Département d'Informatique, Université de Strasbourg*

## 1. The graph theory of $n$-nucleotide circular codes

Throughout this section let $B = \{A, C, G, T\}$ be the set of nucleotide bases, where $A$ stands for *Adenine*, $C$ stands for *Cytosine*, $G$ stands for *Guanine*, and $T$ stands for *Thymine*. For $n \in \mathbb{N}$ with $n \geq 2$, an *$n$-nucleotide code* is a subset $X \subseteq B^n$. The following definition relates a directed graph to any $n$-nucleotide code. Recall from graph theory (Clark and Holton, 1991) that a *graph* $\mathcal{G}$ consists of a finite set of *vertices (nodes)* $V$ and a finite set of *edges* $E$. Here, an edge is a set $\{v, w\}$ of vertices from $V$. The graph is called *oriented* if the edges have an orientation, i.e. edges are considered to be ordered pairs $[v, w]$ in this case.

**Definition 1.** Let $X \subseteq B^n$ be an $n$-nucleotide code ($n \in \mathbb{N}$). We define a directed graph $\mathcal{G}(X) = (V(X), E(X))$ with set of vertices $V(X)$ and set of edges $E(X)$ as follows:

- $V(X) = \{N_1...N_i, N_{i+1}...N_n : N_1N_2N_3...N_n \in X, 1 \leq i \leq n-1\}$
- $E(X) = \{[N_1...N_i, N_{i+1}...N_n] : N_1N_2N_3...N_n \in X, 1 \leq i \leq n-1\}$

The graph $\mathcal{G}(X)$ is called *associated* with $X$.

Basically, the graph $\mathcal{G}(X)$ associated with a code $X$ interprets $n$-nucleotide words from $X$ in $(n-1)$ ways by pairs of $i$-nucleotides and $(n-i)$-nucleotides for $1 \leq i \leq n-1$. The following figures give examples of codes and their representing graphs in the case of $n = 2$ (dinucleotide code), $n = 3$ (trinucleotide code) and $n = 4$ (tetranucleotide code).

*Example* 1. In Figures 1-3, we show three examples of dinucleotide, trinucleotide and tetranucleotide codes and their representing graphs.

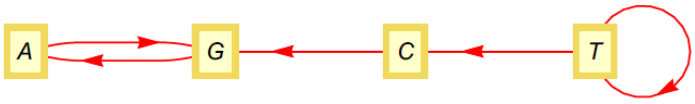


Figure 1. Graph representing the dinucleotide code $\{AG, CG, GA, TC, TT\}$

*E-mail addresses*: c.michel@unistra.fr.

*Date*: September 13, 2025.

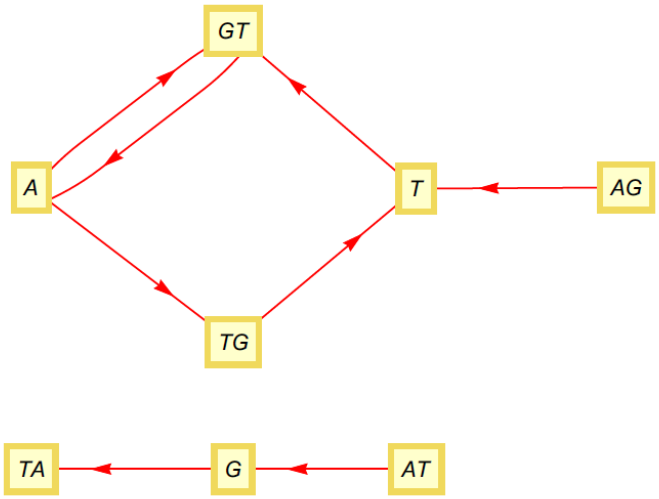


FIGURE 2. Graph representing the trinucleotide code $\{AGT, ATG, GTA, TGT\}$

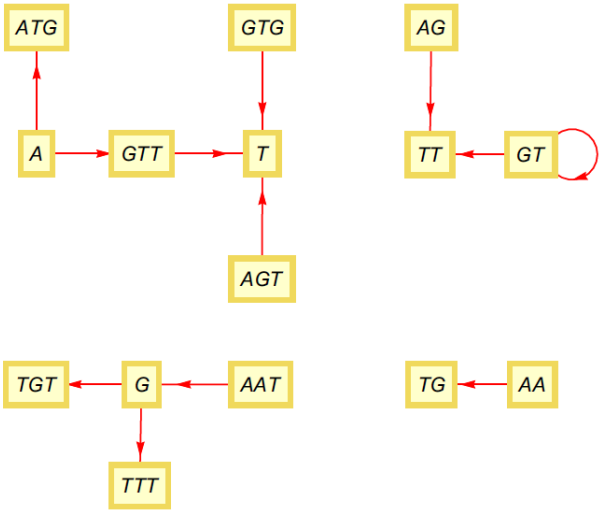


FIGURE 3. Graph representing the tetranucleotide code $\{AATG, AGTT, GTGT, GTTT\}$

As we can see the graph of the tetranucleotide code has four disjoint parts. However, note that two parts are built by vertices labeled with dinucleotides and two parts are built by vertices labeled with nucleotides and trinucleotides. These parts are called *components* of $\mathcal{G}$. Recall that a subset $V'$ of the set of vertices $V$ is called *connected* if for any two nodes $v, w \in V'$ there is a path $[v, v_1][v_1, v_2] \cdots [v_{n-1}, v_n][v_n, w]$ of vertices from $V'$ connecting $v$ and $w$. Any graph decomposes uniquely into connected components which are pairwise disjoint. Recall also that a graph is *bipartite* if its set of vertices $V$ can be decomposed into two disjoint subsets $V'$ and $V''$ such that the edges of $\mathcal{G}$ connect only nodes from $V'$ with nodes from $V''$ and vice versa. Obviously, if $X$ is an $n$-nucleotide code, then the components of $\mathcal{G}(X)$ are exactly the graphs

$$\mathcal{G}(X)_j = (V(X)_j, E(X)_j) \text{ for } 1 \leq j \leq n-1$$

with

$$V(X)_j = \{N_1...N_j, N_{j+1}...N_n, N_1...N_{n-j}, N_{n-j+1}...N_n : N_1N_2N_3...N_n \in X\}$$

and

$$E(X)_j = \{[N_1...N_j, N_{j+1}...N_n], [N_1...N_{n-j}, N_{n-j+1}...N_n] : N_1N_2N_3...N_n \in X\}.$$

These components do not have to be connected as we can see in Figure 3. However, quite often they are. In fact, $\mathcal{G}(X)_j$ consists exactly of the nodes (and their corresponding edges) that interpret the elements of $X$ in two ways: as a pair of a $j$-nucleotide and a $(n-j)$-nucleotide and as a pair of a $(n-j)$-nucleotide and a $j$-nucleotide. Note that by symmetry we have $\mathcal{G}(X)_j = \mathcal{G}(X)_{n-j}$ for all $j < n-1$. For instance, in Figure 3 above the two components of the graph associated to the tetranucleotide code are $\mathcal{G}(X)_1$ $(= \mathcal{G}(X)_3)$ and $\mathcal{G}(X)_2$. The next observation is obvious.

We now start to investigate our desired objects, namely $n$-nucleotide circular codes.

**Definition 2.** Let $X \subseteq B^n$ be a code. We say that $X$ is a *circular code* if for any concatenation $c_1 \cdots c_m$ of $n$-nucleotide words from $X$ there is only one partition into $n$-nucleotide words from $X$ when read on a circle.

The first observation shows that for circular codes the associated graph is already simple. Recall from graph theory (Clark and Holton, 1991) that an oriented graph is *simple* if it does not contain loops, i.e. edges between a node and itself, and does not have multiple edges with the same orientation between two nodes. Note that the orientation for the multiple edges do play a role, i.e. for a simple oriented graph $\mathcal{G}$, we can still have $[x, y] \in E(\mathcal{G})$ and $[y, x] \in E(\mathcal{G})$ which means that there is a cycle (circle) of length 2. However, for circular codes this structure is also excluded. Recall that a *cycle* in $\mathcal{G}$ is an oriented closed path in $\mathcal{G}$. A *circle* is a cycle that visits no node twice except for the starting node (which is the end node at the same time). For instance, in Figure 2 the sequence of vertices *T,GT,A,TG,T* is a circle while the sequence *T,GT,A,GT,A,TG,T* is a cycle that is not a circle.

**Lemma 1** (Fimmel, Michel, Strüngmann, 2015)**.** Let $X \subseteq B^n$ be a circular code. Then its representing graph is a simple oriented graph without circles of length 2.

We now state our first main theorem which proves the connection between circularity of codes and acyclicity of graphs. Recall from graph theory (Clark and Holton, 1991) that a graph is called *acyclic* if it does not contain cycles, i.e. oriented closed paths.

**Theorem 1** (Fimmel, Michel, Strüngmann, 2015)**.** *Given a code $X \subseteq B^n$ the following statements are equivalent:*

*(1) $X$ is circular*
*(2) $\mathcal{G}(X)$ is acyclic*

## 2. Projet: Création d'un site web de codes circulaires

**Objectif du projet**

Le but du projet est de développer un site web permettant de déterminer si un code est circulaire ou non.

**Contraintes générales**

Tous les affichages doivent être réalisés en anglais.

**Fonctionnalités principales**

- Acquisition d'un ensemble de mots, saisi interactivement ou importé depuis un fichier;
- Vérification que l'ensemble obtenu constitue bien un code, et que tous les mots ont la même longueur;
- Affichage du code, de son alphabet, du nombre de mots et de la longueur des mots;
- Construction et affichage de son graphe selon la Définition 1;
- Détermination et affichage de la propriété de circularité: code circulaire ou code non circulaire;
- Si le code est circulaire, affichage de son (ses) plus long(s) chemin(s) et de sa longueur:
  - si cette longueur est égale à 1, affichage de "code strong comma-free";
  - si cette longueur est égale à 2, affichage de "code comma-free".

**Consignes techniques**

Insérer l'instruction d'exécution dans un fichier HTML.
Le programme doit fonctionner avec le serveur dpt-info.di.unistra.fr.

## 3. Modalités du projet

Renvoyer par email à l'adresse c.michel@unistra.fr ou mettre sur un site avec un téléchargement simple de votre programme (source et exécutable).