Approche purement syntaxique (Denise Vella-Chemla, 12.1.2017)

1) Modéliser

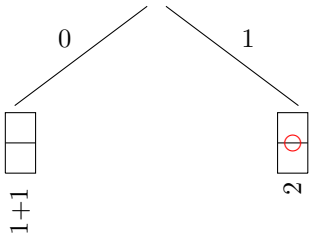
On cherche ce que les nombres premiers symétrisent.

On sait par exemple qu'un nombre premier p a exactement $\frac{p-1}{2}$ résidus quadratiques, tandis que ce n'est pas le cas d'un nombre composé. Si on considère x et $p-x$ inférieurs à p, soit tous 2 sont résidus quadratiques de p simultanément, soit si l'un l'est, l'autre ne l'est pas et inversement. On peut voir dans ces faits une forme de symétrie.

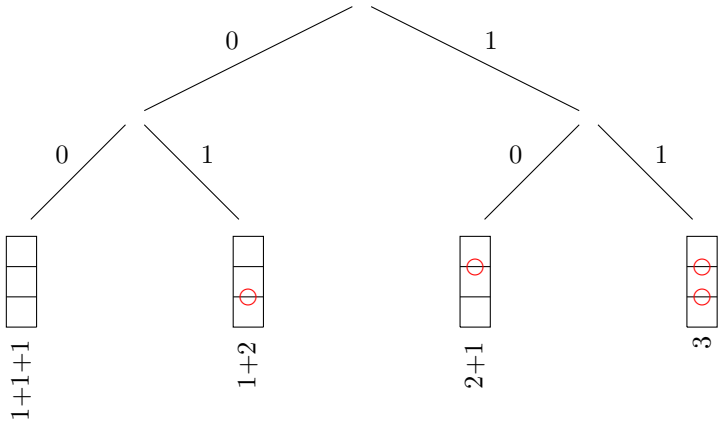
Cherchons d'autres formes de symétrie en étudiant les compositions des nombres. On prend comme référence l'article de wikipedia https://fr.wikipedia.org/wiki/Composition_(*combinatoire*).

Un nombre n a 2^{n-1} compositions différentes. On peut voir les compositions de n comme feuilles d'un arbre binaire de hauteur $n-1$. On voit que l'ordre des sommants importe, 1+1+2+2 et 1+2+1+2 sont des compositions différentes de 6 (alors qu'elles correspondent à la même partition). Il y a une correspondance bijective entre les mots booléens de $n-1$ caractères et les compositions de n. Par exemple, le mot de 5 lettres 10010 correspond à la décomposition $2+1+2+1$ de 6.

Arbre binaire des compositions de 2



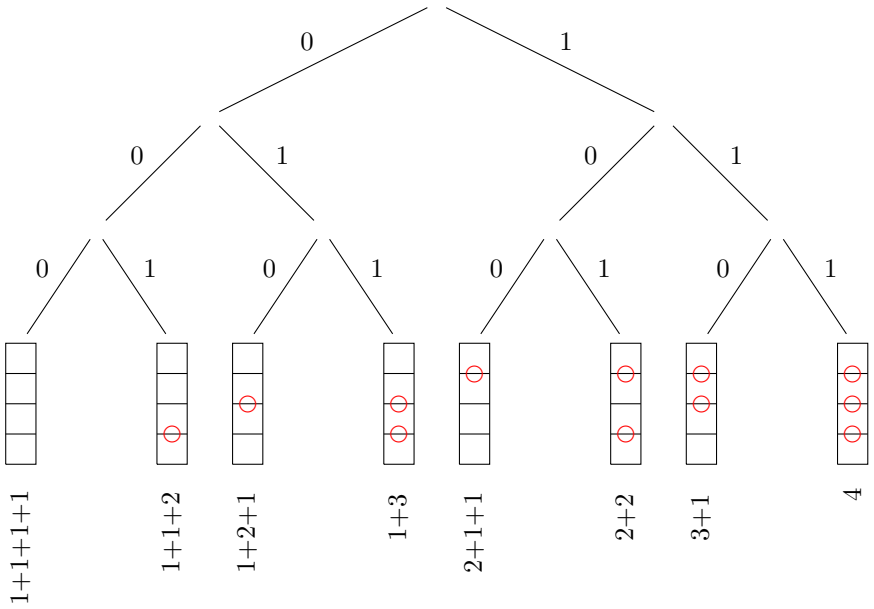
Arbre binaire des compositions de 3 (lire les dessins de haut en bas)



Pour obtenir une composition de $n+1$ à partir d'une composition de n, il suffit soit d'ajouter un sommant 1+ au début de la composition de n, soit de remplacer le premier sommant de la composition de n par son successeur, en obtenant à partir de la composition $n = s_0 + A$ la composition $n+1 = (1+s_0) + A$, le calcul entre parenthèses devant être effectivement effectué.

Voyons cela pour le passage des compositions de 3 à celles de 4 : de l'ensemble $\{1+1+1, 1+2, 2+1, 3\}$ des compositions de 3, on peut concaténer 1+ au début de chaque somme, ce qui permet d'obtenir l'ensemble de compositions de 4 $\{1+1+1+1, 1+1+2, 1+2+1, 1+3\}$; ou bien dans chaque composition de 3, on remplace le premier sommant par son successeur, ce qui permet d'obtenir l'ensemble de compositions de 4 $\{2+1+1, 2+2, 3+1, 4\}$. On réitère le processus pour passer de l'ensemble des compositions de 4 $\{1+1+1+1, 1+1+2, 1+2+1, 1+3, 2+1+1, 2+2, 3+1, 4\}$ à l'ensemble de compositions de 5 constitué de l'union des deux ensembles $\{1+1+1+1+1, 1+1+1+2, 1+1+2+1, 1+1+3, 1+2+1+1, 1+2+2, 1+3+1, 1+4\}$ et $\{2+1+1+1, 2+1+2, 2+2+1, 2+3, 3+1+1, 3+2, 4+1, 5\}$.

Arbre binaire des compositions de 4



On appelle compositions triviales les deux compositions constituées l'une uniquement de n caractères 1 séparés par des signes "+", de la forme $1+1+\ldots+1$, et l'autre de n seul. Un nombre composé est notamment caractérisé par le fait que l'une de ses compositions au moins, différente des deux compositions triviales, est telle que quelle que soit la permutation qu'on pourrait effectuer entre 2 de ses sommants, cette composition reste identique à elle-même. Il en est ainsi de la composition 2+2+2 de 6, ou 5+5+5+5+5 de 25. Un nombre premier n'a aucune de ses compositions qui est ainsi invariante par toute permutation de ses sommants. Cette caractérisation de la primalité est exclusivement syntaxique.

2) Essayer de formaliser

A chaque entier est associé un ensemble de parties de $\mathbb{N}$. Chaque partie de $\mathbb{N}$ est une application de $\mathbb{N}$ dans $\{0,1\}$ et on code par un booléen le fait qu'une partie soit image d'un entier ou pas.

Une partie de $\mathbb{N}$ étant codée par un mot infini, on complète chaque mot booléen de la section précédente par une infinité de 0 ; cette infinité de zéros à droite n'ajoute pas de sommant à l'écriture additive.

$$\begin{array}{l}\mathbb{N} \longrightarrow \{0,1\}^{\{0,1\}^{\mathbb{N}}} \\ n \longmapsto \{s \in \{0,1\}^{\mathbb{N}} / \forall i \geqslant n, s[i] = 0\} \subset \mathcal{P}(\mathbb{N})\end{array}$$

Voyons en quoi consiste le passage d'un entier au suivant selon cette formalisation (informatiquement, c'est trivial, ça consiste à concaténer une lettre 0 ou 1 à gauche des mots de n).

On a le diagramme suivant :

$$\begin{array}{ccc}\mathbb{N} & \xrightarrow{d_n} & \{0,1\}^{\{0,1\}^{\mathbb{N}}} \\ \downarrow{\scriptstyle +1} & & \downarrow{\scriptstyle d_{n+1}} \\ \mathbb{N} & \xrightarrow{d_n} & \{0,1\}^{\{0,1\}^{\mathbb{N}}}\end{array}$$

avec $d_n : \mathbb{N} \longrightarrow \{0,1\}$ et

$$\begin{array}{rccl} d_{n+1} : & k & \longmapsto & d_n(k-1), \forall k \geqslant 1 \\ & 0 & \longmapsto & 0 \\ & & \longmapsto & 1 \end{array}$$

La condition correspondant au fait d'être composé pour un nombre entier qui consiste à avoir une composition (différente des deux compositions triviales) de la forme $x+x+\ldots+x+x$ se traduit très simplement sur les mots booléens : elle consiste en l'apparition dans le mot d'une puissance au moins carrée d'un sous-mot non-nul (le sous-mot en question est appelé période en théorie des langages rationnels) ; par

exemple, la composition $2+2+2$ de 6, codée par le mot $1010100000\ldots$ contient le sous-mot 10 répété trois fois, elle s'écrit $(10)^3 0^\infty$.

Annexe 1 : les 32 compositions de 6 et leur mot booléen correspondant

$1+1+1+1+1+1$ (00000)	$2+1+1+1+1$ (10000)
$1+1+1+1+2$ (00001)	$2+1+1+2$ (10001)
$1+1+1+2+1$ (00010)	$2+1+2+1$ (10010)
$1+1+1+3$ (00011)	$2+1+3$ (10011)
$1+1+2+1+1$ (00100)	$2+2+1+1$ (10100)
$1+1+2+2$ (00101)	$2+2+2$ (10101)
$1+1+3+1$ (00110)	$2+3+1$ (10110)
$1+1+4$ (00111)	$2+4$ (10111)
$1+2+1+1+1$ (01000)	$3+1+1+1$ (11000)
$1+2+1+2$ (01001)	$3+1+2$ (11001)
$1+2+2+1$ (01010)	$3+2+1$ (11010)
$1+2+3$ (01011)	$3+3$ (11011)
$1+3+1+1$ (01100)	$4+1+1$ (11100)
$1+3+2$ (01101)	$4+2$ (11101)
$1+4+1$ (01110)	$5+1$ (11110)
$1+5$ (01111)	6 (11111)

Annexe 2 : Programme de passage des mots booléens aux compositions

```
#include <cstdlib>
#include <iostream>
#include <vector>

using std::cout;
using std::endl;
using std::atoi;

typedef std::vector<bool> bitset;

void print_bits(bitset &bits) {
    for (int i = 0; i < bits.size(); ++i)
        cout << (int) bits[i];
}

void print_deco(bitset &bits) {
    int s = 1;
    for (int i = 0; i < bits.size(); ++i) {
        if (bits[i]) {
            ++s;
        } else {
            cout << s << "+";
            s = 1;
        }
    }
    cout << s;
}
```

```
void generate(int n, bitset &bits) {
    if (n-- > 0) {
        bits[n] = 0; generate(n, bits);
        bits[n] = 1; generate(n, bits);
    } else {
        print_bits(bits);
        cout << " : ";
        print_deco(bits);
        cout << endl;
    }
}

int main(int argc, char* argv[]) {
    int n_max = 0;
    if (argc > 1) n_max = atoi(argv[1]);
    for (int n = 0; n < n_max; ++n) {
        cout << "n = " << n + 1 << " :" << endl;
        bitset bits(n);
        generate(n, bits);
        cout << endl;
    }
}
```