FACULTE DES SCIENCES DE LUMINY

Maths en Jeans 1

Les Digicodes

Virginie DUPRAT
Maëlys GARDEY
Julien TEBOUL

Table des Matières

Remerciements

Nous tenons à remercier dans un premier temps l'équipe pédagogique, Laurent Beddou et Julien Cassaigne, pour avoir assuré un suivi sur notre sujet, nous avoir conseillé et orienté.

Nous remercions également Monsieur Beffara, responsable de la formation MATHS DISCRETES 2 qui nous a apporté des connaissances nécessaires à l'évolution de ce sujet et Monsieur Van Caneghem, responsable de la formation FONDEMENTS DE L'INFORMATIQUE pour le logiciel qu'il nous a fourni.

Nous remercions plus particulièrement Messieurs Guieu et Tallot, collègues et amis, étudiant en L2 Informatique, pour l'aide évoquée dans ce rapport.

Historique du digicode

Le digicode (marque déposée par CDVI - PANTIN) a été inventé en 1970 par le français Bob Carrière.

C’est un type de serrure électronique qui s'ouvre en saisissant un code secret sur un pavé numérique.

Il est notamment employé sur les portes d'immeubles pour en limiter l'accès, et sert alors d'alternative ou de complément à l'interphone.

Les digicodes sont également utilisés à l'intérieur de bâtiments pour limiter l'accès d'une salle ou d'un secteur à une catégorie d'usagers. La salle des coffres d'une banque est l'exemple typique de cet usage.

Le digicode, au début simplement un pavé numérique avec des chiffres de 0 à 9, est devenu de plus en plus complexe avec de nos jours, l’ajout de lettre pour le code.

Un sketch de l'humoriste français Marc Jolivet met en scène un homme ivre souhaitant rentrer chez lui au milieu de la nuit en ayant oublié le code d'entrée, qui tente de parler au digicode comme il le faisait avec la concierge à l'époque où ce dispositif n'existait pas.

Définition du sujet

On a un digicode composé uniquement des nombres de 0 à 9, sans bouton « validé ». Dès que le code est tapé, le dispositif associé au digicode se déverrouille. Nous appelons cela la mémoire de frappe.
L’objectif est de trouver le plus rapidement possible le code et tout ce qui en déduit.

Plan de recherche

Afin de pouvoir avancer correctement et comprendre notre sujet dans son ensemble nous avons eu recourt à de nombreuses réductions et simplifications qui seront petit à petit enlevées.
Tout d'abord nous avons commencé nos recherches voulant éviter les répétitions d'un même chiffre dans un code. Cependant, à ce stade, cette simplification était inefficace et ne permettait pas de poursuivre les recherches.

Nous avons alors imaginé une façon plus radicale pour diminuer la difficulté. Nous désirons réduire la longueur des codes (code ≤ 4 chiffres) et le nombre de chiffres que l'on peut taper sur le digicode (chiffres de 0 à n, n ≤ 9).

Très vite confirmée par nos professeurs, cette hypothèse nous a amené à considérer des bases plus petites.

Dans un premier temps nous allons étudier les codes en base 2, en jouant sur la longueur des codes. On essaye alors de déterminer des propriétés et méthodes adéquates pour résoudre notre problème.

Ensuite, nous voulons augmenter un peu la difficulté et tester nos hypothèses, nous passons donc à la base 3. Nous avons pu faire de nouvelles conjectures sur le nombre de tapes et l'efficacité de notre méthode.

Enfin, nous avons reporté ces résultats en base 10. Nous en avons tiré une nouvelle approche, basée sur les réflexions dans les plus petites bases et dans un souci de simplicité avons rétabli l'impossibilité d'avoir des répétitions.

Journal de Bord

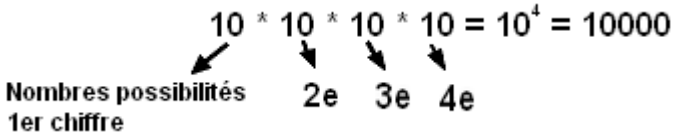
18 Janvier 2012 :

Ceci est notre première séance, une fois le groupe formé, nous commençons à parler du sujet.

On a analysé les termes du sujet et la problématique qui ont été données. Il en résulte cette compréhension du sujet :

On cherche à écrire une suite de chiffre la plus courte possible, comportant tous les codes apparaissant une seule fois.

On a calculé que pour un code dans un digicode traditionnel (sans les lettres), la chaine de nombres aura

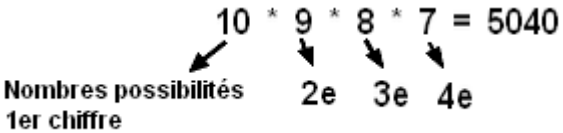


A partir de cette base, on a vite compris que pour traiter ce sujet, nous devons d'abord le simplifier.

Nous avons pensé aux simplifications suivantes :

On s'intéressera d'abord au cas où les codes ne comportent pas deux fois le même chiffre.

Avec cette simplification, il n'y aura plus que



Cette simplification n'est pas suffisante. Avant de s'intéresser aux codes à 4 chiffres avec des valeurs de 0 à 9, on doit explorer des pistes plus simples et plus utiles.

Nous nous sommes alors posé plusieurs questions :

- Quel serait le nombre de possibilités en rajoutant certains critères ? Sera-t-il inférieur ?
- Quelle est la méthode la plus adéquate pour trouver le code ?

Avec de légères modifications, on espère pouvoir transposer ces raisonnements à notre problématique générale.

Pour la prochaine fois, on va imprimer un tableau avec les nombres allant de 0000 à 9999 et pendant la semaine, nous travaillerons sur un code à 2 chiffres plutôt que 4.

25 Janvier 2012 :

Au début de la séance, nous arrivons avec notre tableau de nombres mais après réflexion, nous décidons de le mettre de côté.
Pendant la semaine, nous avons abordé le code à 2 chiffres de 0 à 9 et il en résulte qu'à part un net raccourcissement de la chaine de nombre, cette simplification n'est pas particulièrement utile et demeure sans intérêt dans la compréhension globale de notre sujet.

Nous décidons au cours de la séance de nous pencher sur un problème plus simple :
Un code à 2 chiffres de 0 à 2 avec mémoire et sans répétition (les deux chiffres sont différents).
On pense être arrivé à une simplification optimale.

Suite à notre cours sur la théorie des graphes enseigné au semestre 3 par M. Beffara dans l'unité Maths Discrètes 2, nous pensons pouvoir modéliser l'ensemble du problème sous forme de graphes.

Graphe : *Un graphe G est la donnée de deux ensembles finis S_G (les sommets) et A_G (les arêtes), et d'une fonction d'incidence sigma G*

Sommet : *Ils sont parfois appelés objets ou points. L'ensemble S est parfois noté V (anglais vertex).*

Arête : *Elles sont parfois appelées arcs ou parfois flèches. L'ensemble A est parfois noté E (anglais edge).*

Pour nous, un sommet doit correspondre à un code. Une arête illustre la connexion entre deux sommets qui peuvent se suivre dans la chaine de nombres que l'on cherche à établir.

Graphe orienté : *Un graphe orienté et un graphe auquel on rajoute une nouvelle composante aux arêtes, un sens.*

Pour concrétiser la suite de nombres, on a besoin de savoir dans quel ordre écrire les sommets. C'est pourquoi, on a besoin d'utiliser un graphe orienté.

Ainsi, pour former une telle suite, il suffit de sélectionner un sommet initial arbitrairement et de créer un chemin passant par tous les sommets.

Chemin : *Soit G un graphe, Un chemin dans G est une suite finie alternée* $p=(a_0,e_1,a_1,\ldots,e_n,a_n)$ *où les* a_i *sont des sommets et les* e_i *sont des arêtes.*

On se rend compte immédiatement que ce n'est pas suffisant. En effet, il peut exister entre deux sommets plusieurs arêtes, et dans différents sens. Il se peut que l'on passe plusieurs fois par chaque sommet donc la suite ne sera plus minimale et certains codes peuvent être répétés. On doit nécessairement introduire une nouvelle règle.

Circuit hamiltonien *: Chemin fermé qui passe une fois et une seule par chaque sommet d'un graphe*

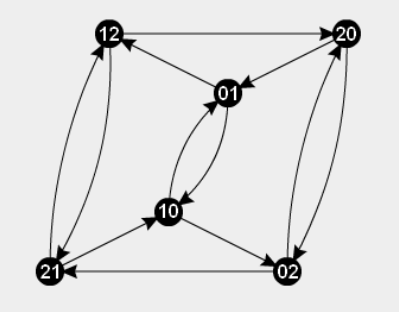
La nécessité de construire un circuit hamiltonien répond parfaitement aux critères exigés pour construire la suite.

La représentation des différents codes sous forme de tableau est :

	0	**1**	**2**
0		01	02
1	10		12
2	20	21	

Cherchons un circuit hamiltonien afin d'éviter les répétitions et le chemin le plus court.

La modélisation sous forme de graphe est :



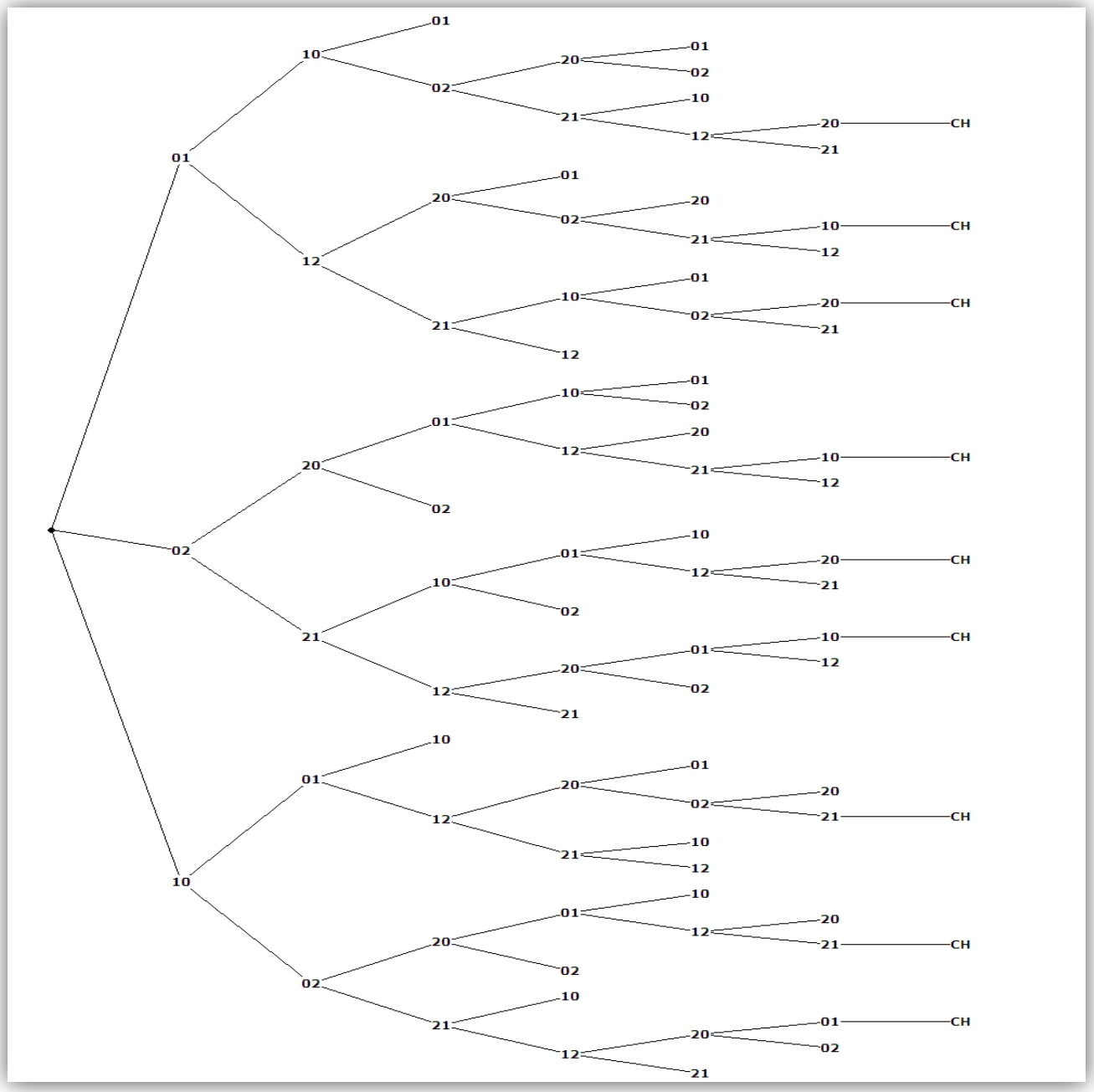
Nous trouvons par exemple le circuit hamiltonien suivant :

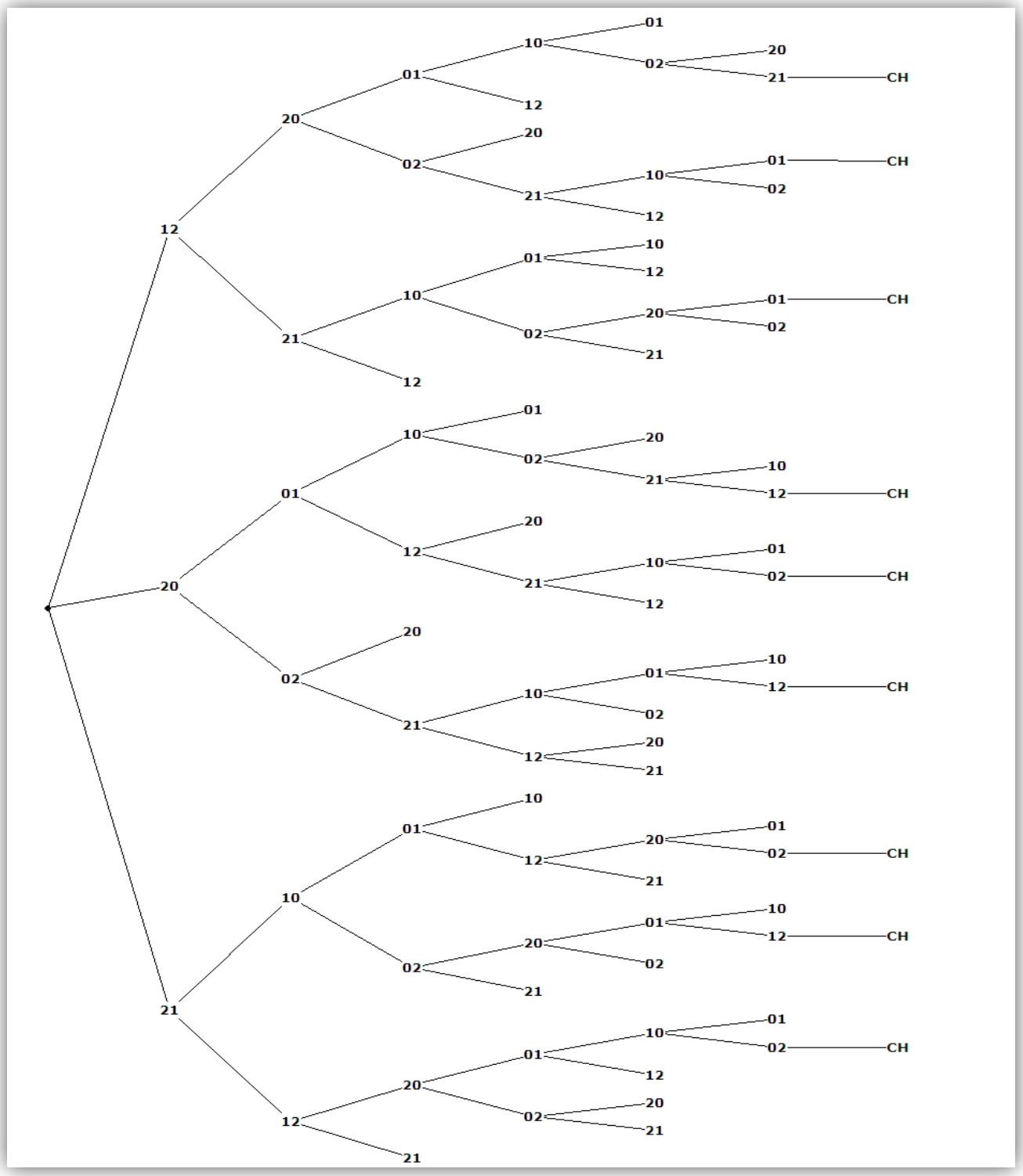
01 → 12 → 20 → 02 → 21 → 10 → ….

Ainsi, une des trois combinaisons, qui contient le code et qui commence par 01, une des plus courtes est 0120210.

Remarque : Si on n'exclut pas la répétition, on ajoute juste 3 chiffres. La combinaison précédente devient 0112002210.

La modélisation sous forme d'arbre est :





On veut savoir quel est le nombre de circuits hamiltonien possibles ?

Pour chaque départ, il y a 3 circuits.

Par exemple avec le départ 21, 21 → 10 → 01 → 12 → 20 → 02 →…

21 → 10 → 02 → 20 → 01 → 12→…

21 → 12 → 20 → 01 → 10 → 02→…

En sachant qu'il y a 6 départs possibles, il y a $3 \times 6 = 18$ circuits hamiltonien.

Les suites par exemple commençant par 21 sont : 2101202, 2102012 et 2120102.
Toutes les suites, quelque soit le départ, sont de longueurs 7.

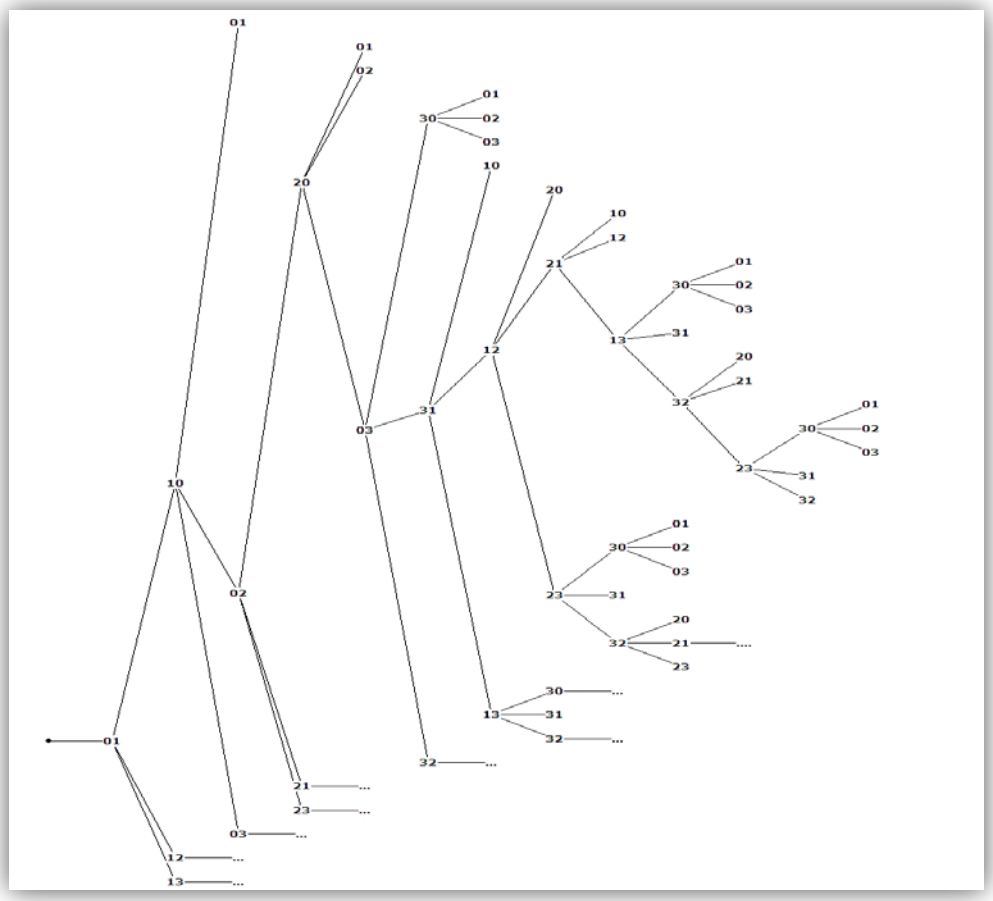
Remarque : Soit n le nombre de circuits hamiltoniens partant d'un sommet choisi
Alors le nombre total de circuits hamiltoniens dans le graphe s'obtient par :
$n \times (nombre\ de\ sommets\ du\ graphe)$.

Suivant la combinaison, le code apparait plus rapidement (même dans le pire cas possible).

Remarque : Il faut finir par le chiffre du début ce qui corrobore la théorie du circuit hamiltonien.

On a voulu passer aux chiffres de 0 à 3. Ce cas se révélant trop compliqué nous avons abandonné la construction de l'arbre et du graphe.

Le début de la modélisation sous forme d'arbre est :



Pour la prochaine fois, nous allons faire des recherches sur des algorithmes pour essayer de modéliser informatiquement les suites.

1 Février 2012 :

Après nos recherches au cours de la semaine, nous avons trouvé l'algorithme de Knuth-Morris-Pratt et celui de Boyer – Moore. Les deux sont des algorithmes de recherche de sous-chaîne.

L'algorithme de Knuth-Morris-Pratt (souvent abrégé par algorithme KMP) est un algorithme de recherche de sous-chaîne, permettant de trouver les occurrences d'une chaîne dans un texte. L'algorithme a été inventé par Knuth et Pratt, et indépendamment par J. H. Morris en 1975.

L'algorithme de Boyer-Moore est particulièrement efficace. Il a été développé par Robert S. Boyer et J Strother Moore en 1977.

En utilisant ces algorithmes, nous allons essayer de modéliser les résultats de nos recherches par un programme informatique afin de pouvoir augmenter la taille du code secret et le nombre de valeurs possibles pour un chiffre du code.

Notre programme doit :

1- Calculer le nombre de codes possibles en fonction de :
    a. Taille digicode
    b. Nombre de caractère possibles pour chaque chiffre
2- Etablir une chaine de caractère remplissant les conditions suivantes :
    a. Pas de répétitions pour un code
    b. Chaque code doit apparaitre dans la chaine une seule fois
    c. Avec un code initial :
        i. Soit aléatoire
        ii. Soit prédéfini par l'utilisateur

Les algorithmes KMP et Boyer-Moore se révèlent beaucoup trop compliqués à exploiter dans notre programme. On va essayer de créer notre propre algorithme.

Pour la prochaine fois, nous avons pour objectif de créer ce programme.

8 Février 2012 :

Pendant la semaine, nous avons réfléchi à l'élaboration du programme en lui-même.

Pour cela, on prend un nombre aléatoire entre 0 et carac (qui correspond au chiffre maximal que l'on peut choisir) : nbre

Avec la condition que nbre est différents des trois chiffres précédents et que la taille…

Voici quelques pistes explorées pour le programme :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TAILLE_MAX


int nbre_codes (int carac)
{
        int i, nbre = 1;
        for (i=0; i<3; i++)
        {
                nbre = carac*nbre;
                carac--;
        }

        return nbre;
}



/*void remplir_chaine (int carac, int init[])
{
        int i, j;
        int tt = nbre_codes(carac) + 3;
        int t[tt];
        int temp;
        int erreur =0 ;
        int x[4];
        for(i=0; i<4; i++)
                t[i] = init[i];

        i=4;

                do
                {
                        temp = rand() % carac;
                        if ((temp != t[i-1]) && (temp != t[i-2]) && (temp != t[i-3]))
                        {
                                x[0] = t[i-3];
                                x[1] = t[i-2];
                                x[2] = t[i-1];
                                x[3] = temp;

                                for (j =0; j<i-1; j++)
                                {
                                        if ( (x[0]==t[j]) && (x[1]==t[j+1]) && (x[2]==t[j+2]) && (x[3]==t[j+3]))
                                        {
                                                erreur = 1;
                                                break;
                                        }
                                }

                                if (erreur != 1)
                                {
                                        t[i] = temp;
                                        i++;
                                }
                        }

                }
                while ( i != tt);

                for (i=0; i< tt; i++)
                        printf ("%d", t[i]);


}
*/
```

```c
int main (void)
{
        int c;
        char code[3];

        //FILE* fichier = NULL;
        //fichier = fopen("chaine.txt","r");

        printf ("Quel est le nombre maximal sur le digicode ?\n");
        scanf("%d", &c);

        printf("Le nombres de codes possibles est:\n%d\n", nbre_codes(c+1));

        printf("Donnez un code à trois chiffres:\n");
        fgets(code, sizeof code, stdin);

        //while (fgets(tempchaine,nbre_codes(c+1)+2,fichier) != NULL
        //{



        /*printf("Code initial ? :\n");
        for(i=0;i<4; i++)
        {
                printf("Chiffre %d ? :\n", i);
                scanf("%d", &init[i]);
        }

        remplir_chaine(c,init);
        */
        return 0;
```

Ce programme ne nous parait pas très essentiel pour l'instant et extrêmement dur à réaliser. Nous n'avons pas les capacités informatiques et préférons l'abandonner pour l'instant. L'aspect graphique du problème est plus important.

Nous allons donc retour à cette idée.

Suite à une longue discussion avec M. Beddou, nous mettons de côté nos recherches actuelles et nos résultats pour nous intéresser aux nombres binaires. Nous reprendrons ces pistes plus tard mais s'intéresser aux changements de bases nous permettra de trouver des méthodes de résolution efficaces et adaptables aux bases supérieures.

Pour la semaine prochaine, nous réfléchirons à une mise en pratique de cette théorie.

15 Février 2012 :

Pendant la semaine, nous avons réfléchi au passage à la base 2. Par définition, les nombres binaires n'ont que des chiffres de 0 à 1 uniquement. Chaque nombre binaire correspond à un code.

Cette correspondance pour le binaire $a_4a_3a_2a_1$ est donné par la formule

$$a_4 \times 2^3 + a_3 \times 2^2 + a_2 \times 2^1 + a_1 \times 2^0$$

Par exemple 10 correspond au code $0 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 0 = 2$.

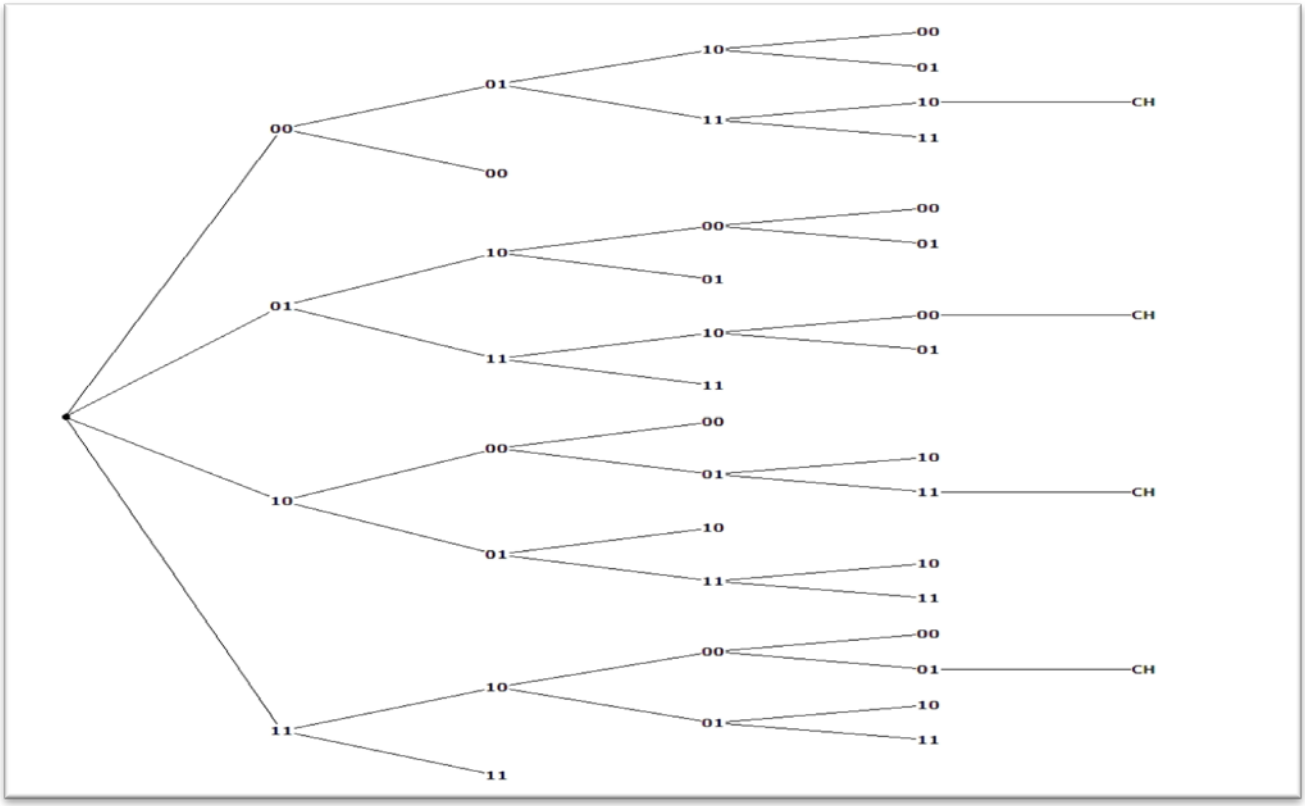
Dans cette base, nous aurons que le cas avec répétition car comme chaque nombre binaire correspond à un code et que dans les nombres binaires, il y a des répétitions.

Au cours de la séance nous allons évoluer du code à 2 chiffres jusqu'au code à 4 chiffres.
Nous avons séparé le travail en trois parties pour construire notre solution :

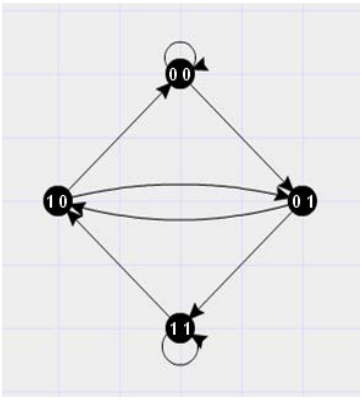
Un premier objectif est de créer le graphe correspondant à toutes les connexions possibles entre chaque sommet.
Par la suite nous avons transposé ces possibilités et avons tracé un arbre.
Enfin chaque solution peut être transposée sur le graphe initial, faisant nécessairement apparaitre un circuit hamiltonien. Quand il y a plusieurs possibilités, on en choisit une arbitrairement. En commençant par le sommet de notre choix, on construit la suite en parcourant un à un les somment dans l'ordre imposé.

<u>Code à 2 chiffres :</u>
Nous aurons quatre codes : 00, 01, 10, 11.

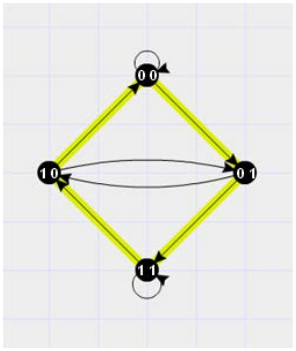
La modélisation sous forme d'arbre est :



La modélisation sous forme de graphe est :



Les circuits hamiltoniens pour les nombres binaires à 2 chiffres sont :



00 → 01 → 11 → 10 →…

10 → 00 → 01 → 11 →…

01 → 11 → 10 → 00 →…

11 → 10 → 00 → 01 →…

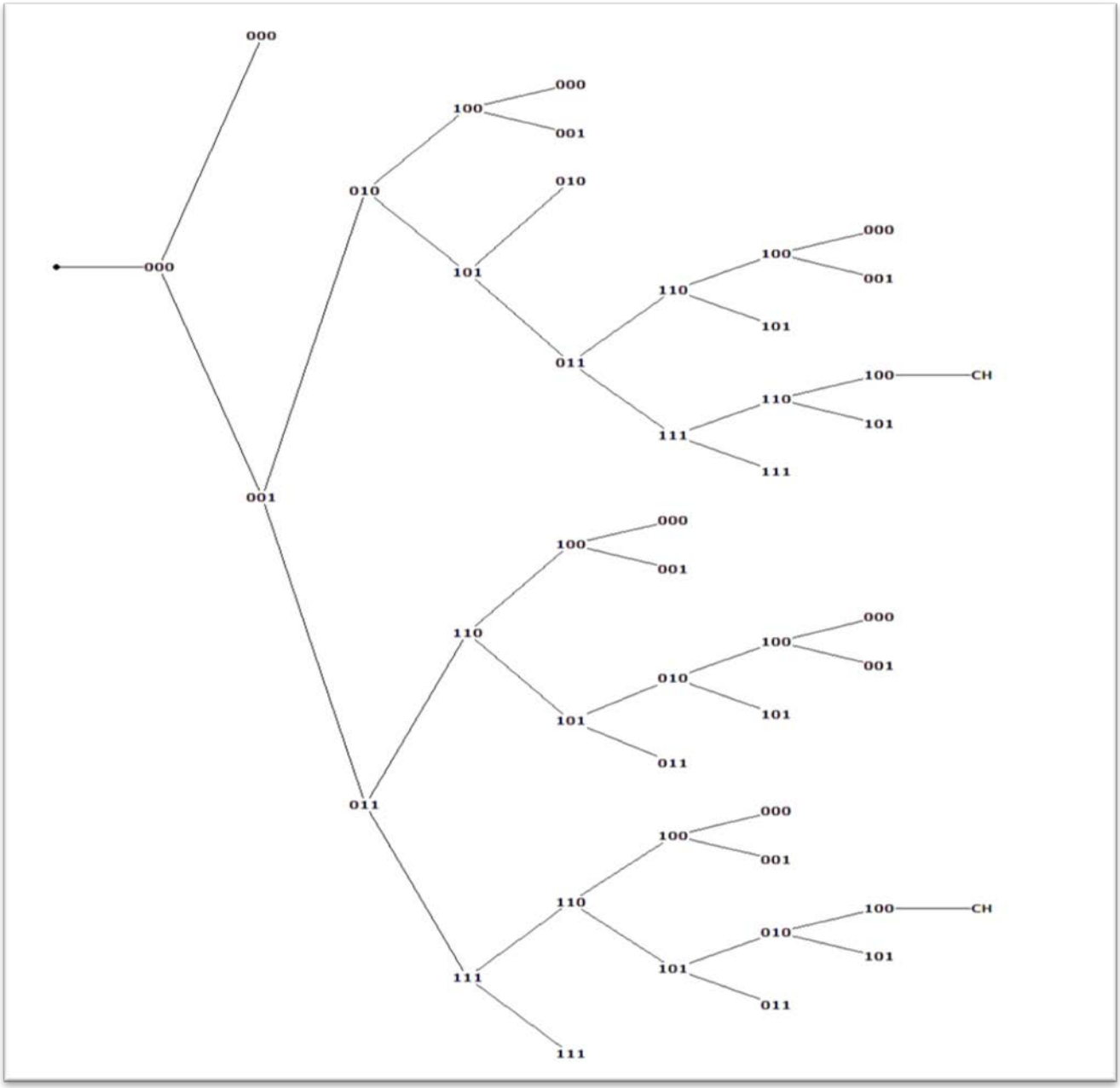
Il y a 4 circuits hamiltoniens. Les combinaisons sont de longueur 5, exemple 00110.
Il existe une unique possibilité pour chaque départ.

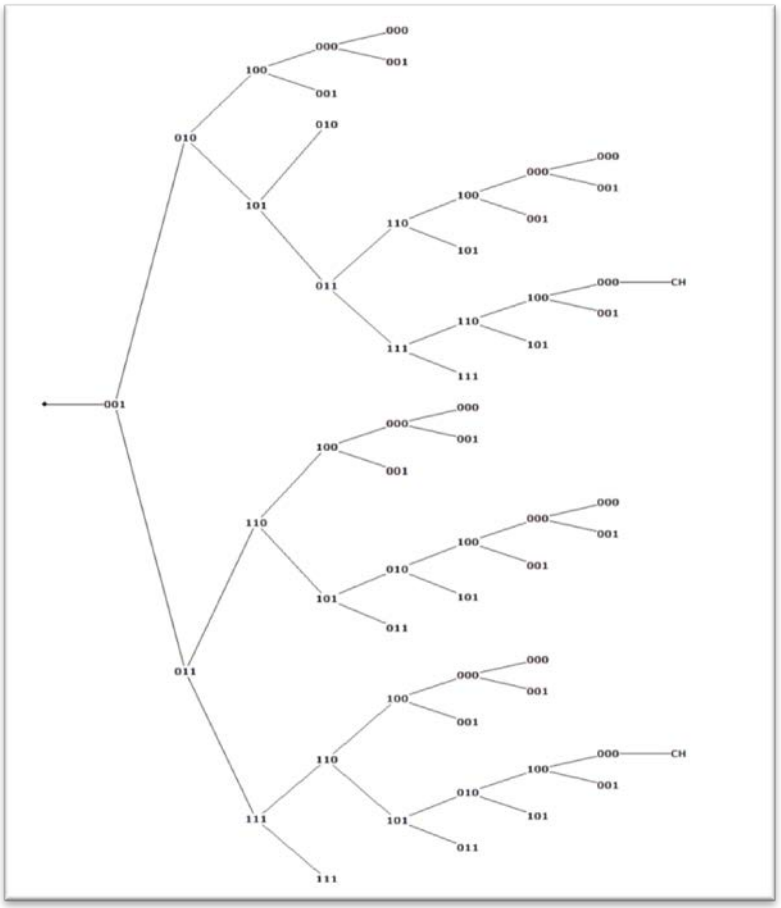
Codes à 3 chiffres :

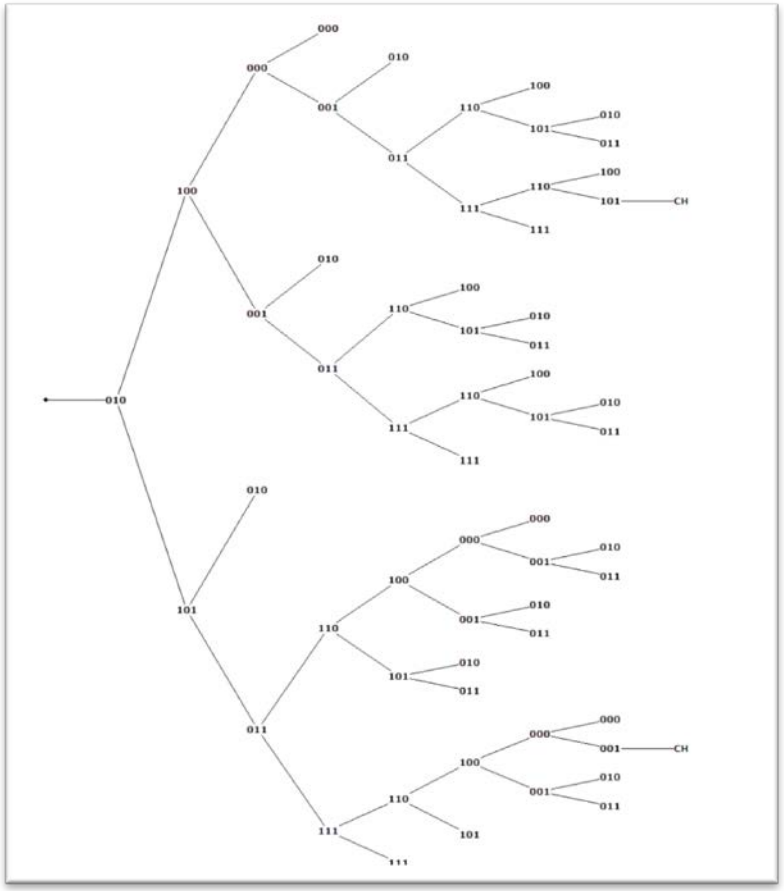
Nous aurons 8 codes : 000, 001, 010,011, 100, 101, 110, 111.

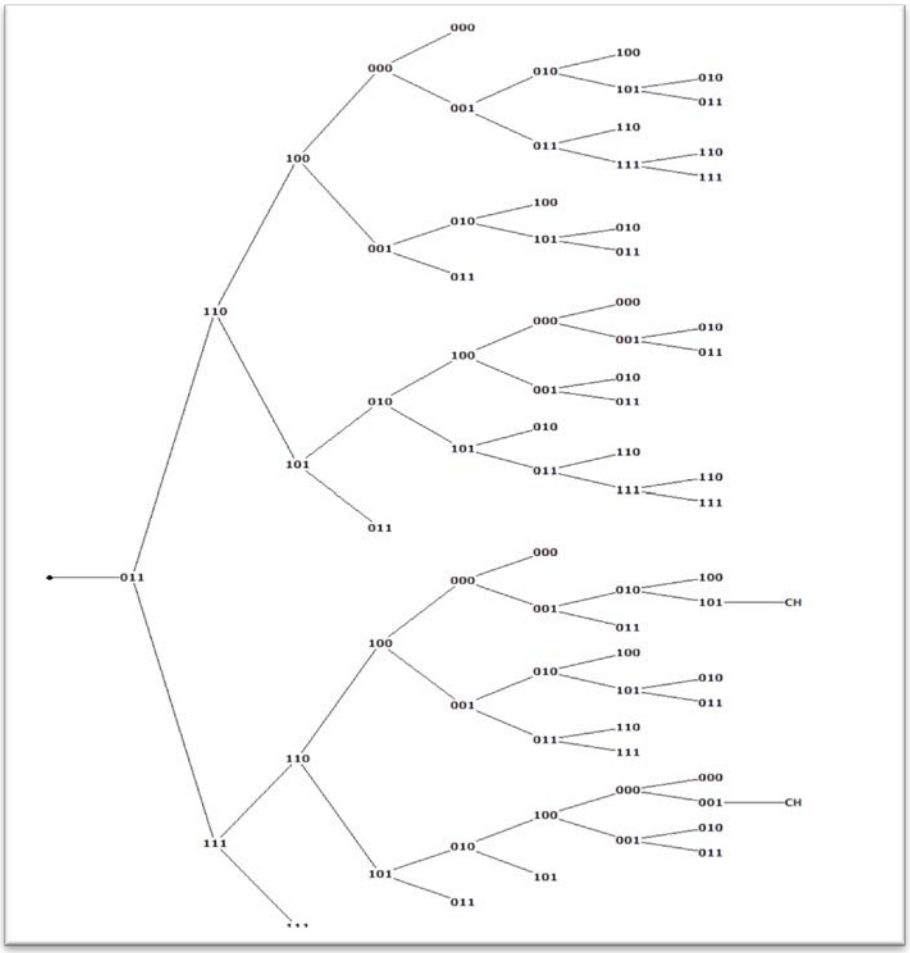
Vu que l'arbre devient complexe avec plus de branches qu'avant, nous avons cherché un logiciel qui modélise les arbres de façon clair et symétrique. Nous avons trouvé « Tree Diagram Generator ».

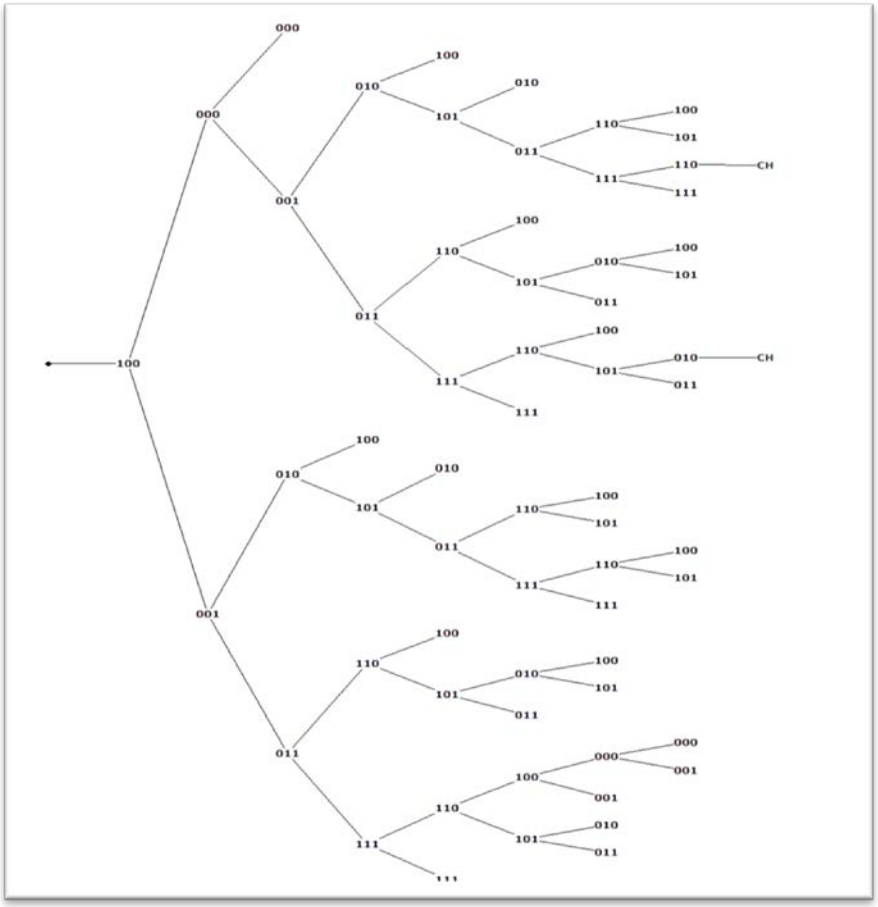
La modélisation sous forme d'arbre est :

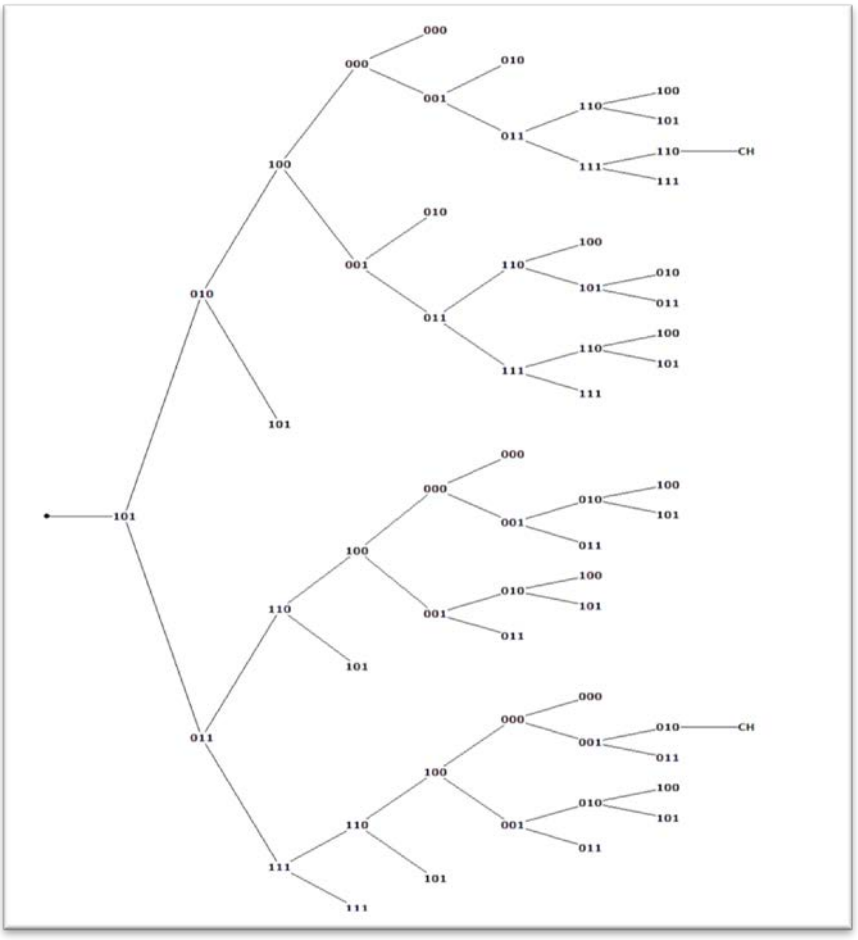


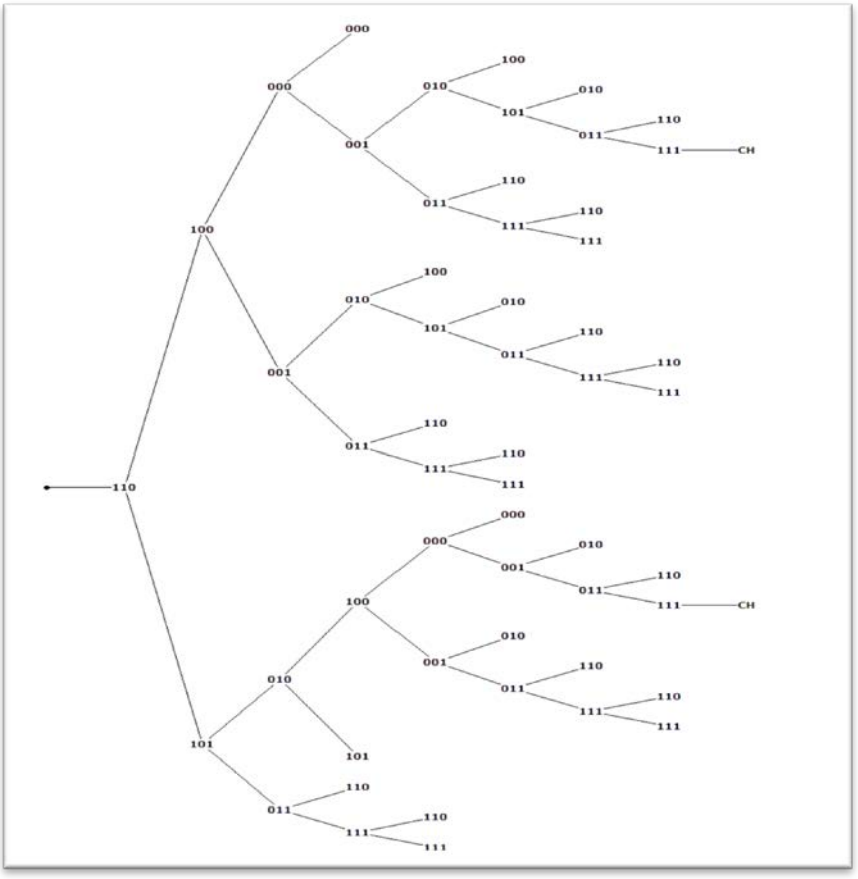

001
100
000
000
001
001
010
010
101
011
110
100
000
000
001
001
101
111
000
CH
100
001
110
101
111
110
000
000
001
001
100
010
101
101
011
011
000
000
001
001
100
110
010
000
000
001
CH
100
001
010
101
111
101
011
111


010
000
000
010
001
100
110
011
101
010
011
111
100
110
101
CH
111
010
001
100
110
010
011
101
011
111
010
101
010
000
000
001
010
011
100
001
010
011
101
010
011
011
000
000
001
CH
100
001
010
011
110
101
111
111


011
100
110
111
000
001
010
011
101
100
110
111
CH

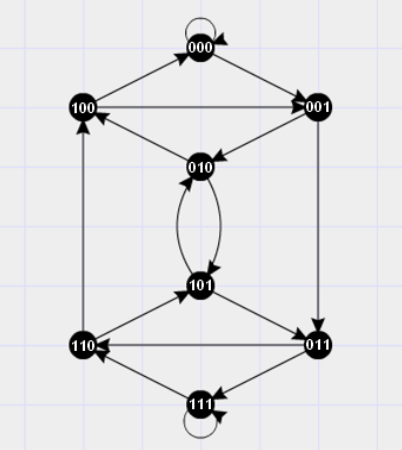

100
000
001
010
011
101
110
111
CH


101
010
100
000
000
001
010
011
100
101
110
111
001
010
011
100
101
110
111
101
011
100
000
000
001
010
011
100
101
001
010
011
100
101
110
101
111
000
000
001
010
CH
011
100
010
100
101
110
111
101
111

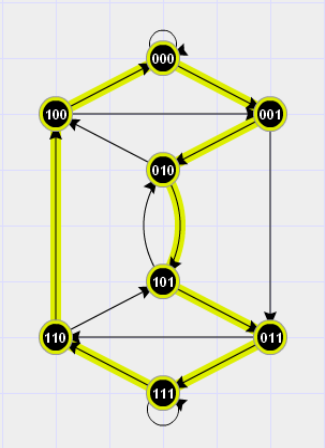

110
100
000
000
001
010
100
101
010
011
110
111
CH
011
110
111
110
111
001
010
100
101
010
011
110
111
110
011
110
111
110
111
101
010
100
000
000
001
010
011
110
111
CH
001
010
100
101
110
111
101
011
110
111
110
111

A partir de maintenant les arbres se feront avec ce logiciel.

La modélisation sous forme de graphe est :

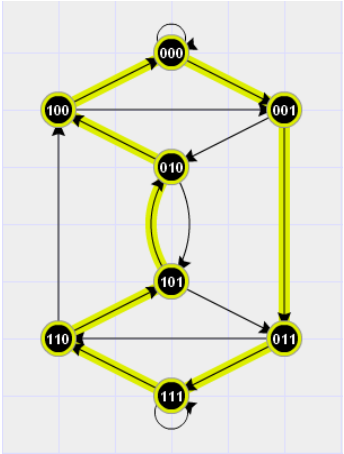


Les circuits hamiltoniens pour les nombres binaires à 3 chiffres commençant par 000 sont :



000 → 001 →
010 → 101 →
011 → 111 →
110 → 100 →…

000 → 001 →
011 → 111 →
110 → 101 →
010 → 100 →…



Il existe deux possibilités pour chaque départ.

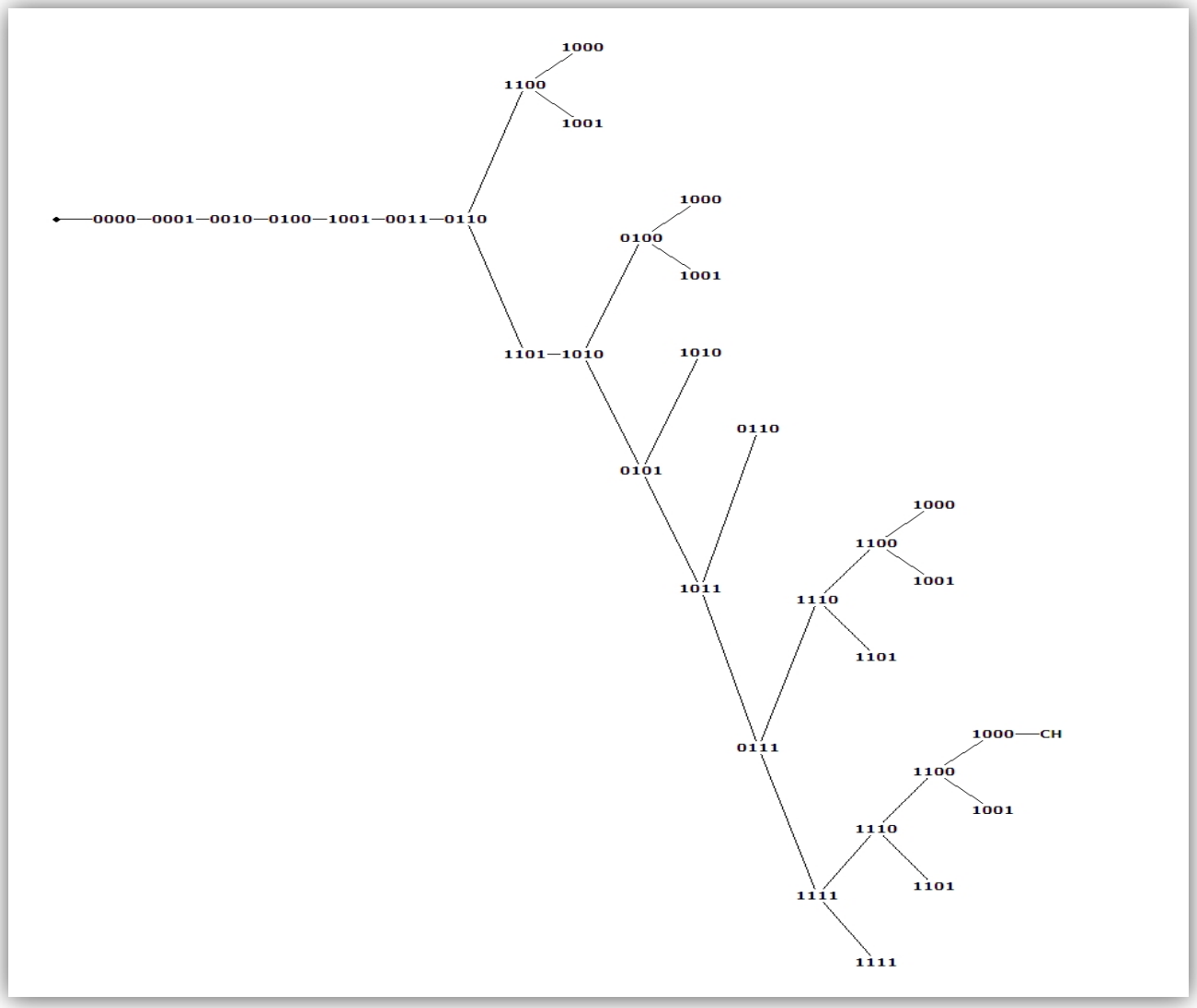
Il y a, pour ce cas, 2 circuits hamiltoniens pour départ 000, exemple 0001110100.
En tout, il y a $8 \times 2 = 16$ circuits hamiltoniens.

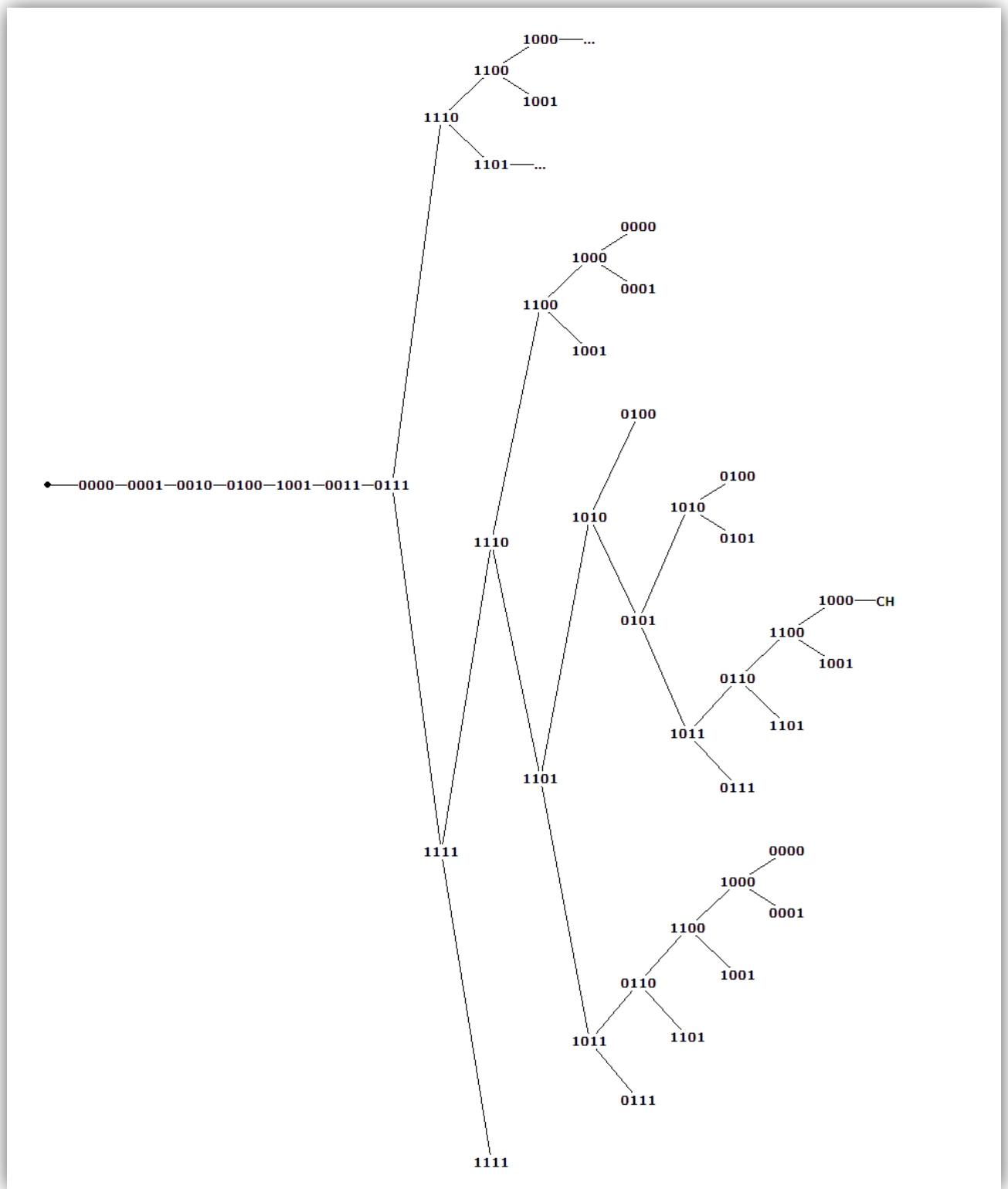
Les suites commençant par 000 sont : 0001011100 et 0001110100.
Toutes les suites, quelque soit le départ, ont une longueur de 10.

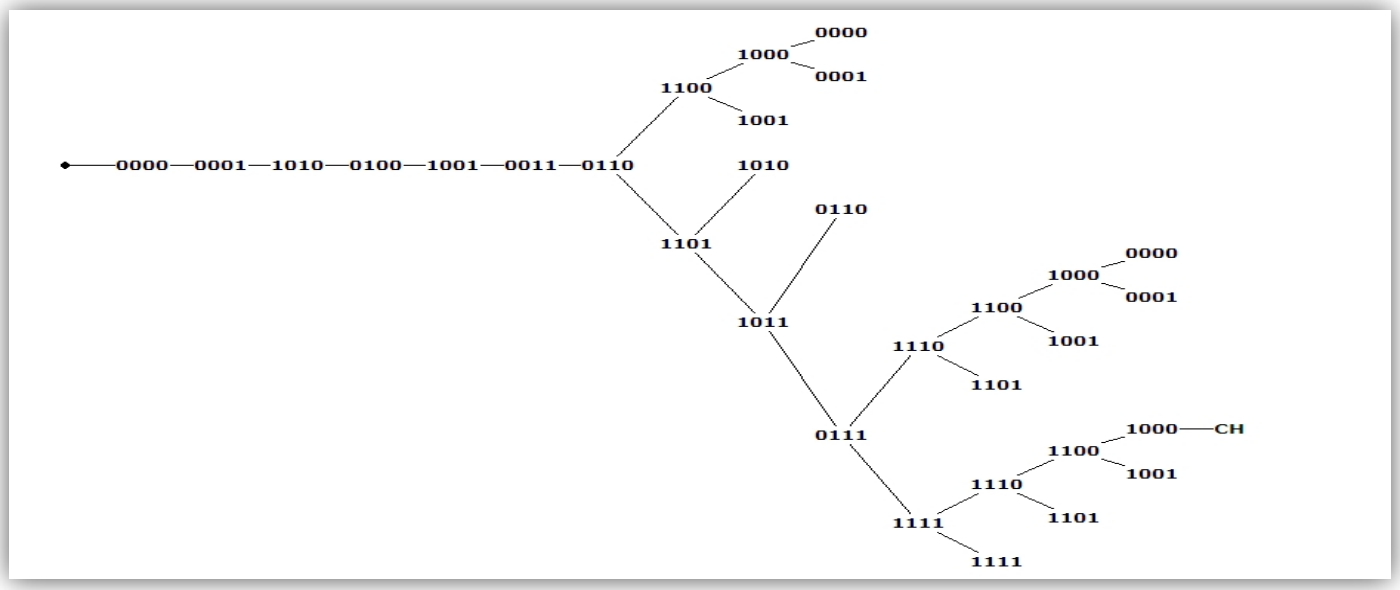
Code à 4 chiffres :

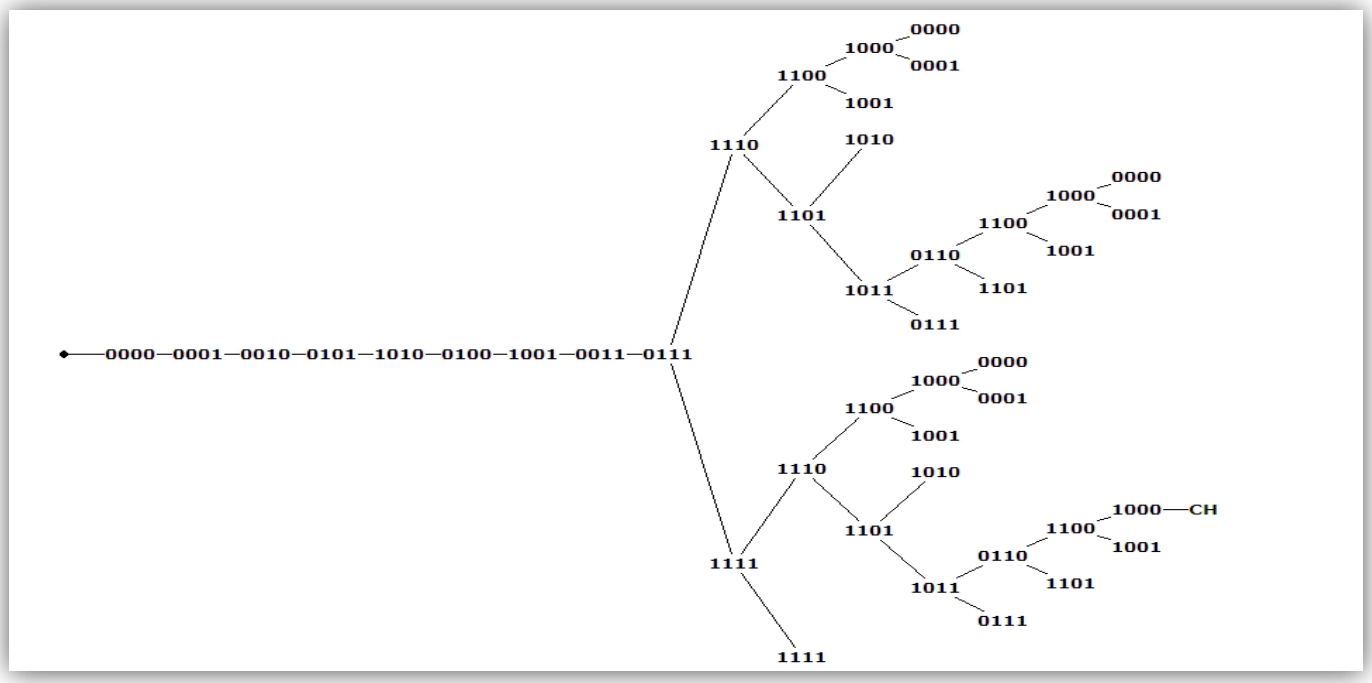
Nous aurons 16 codes : 0000, 0001, 0010, 0011, 0100,0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.

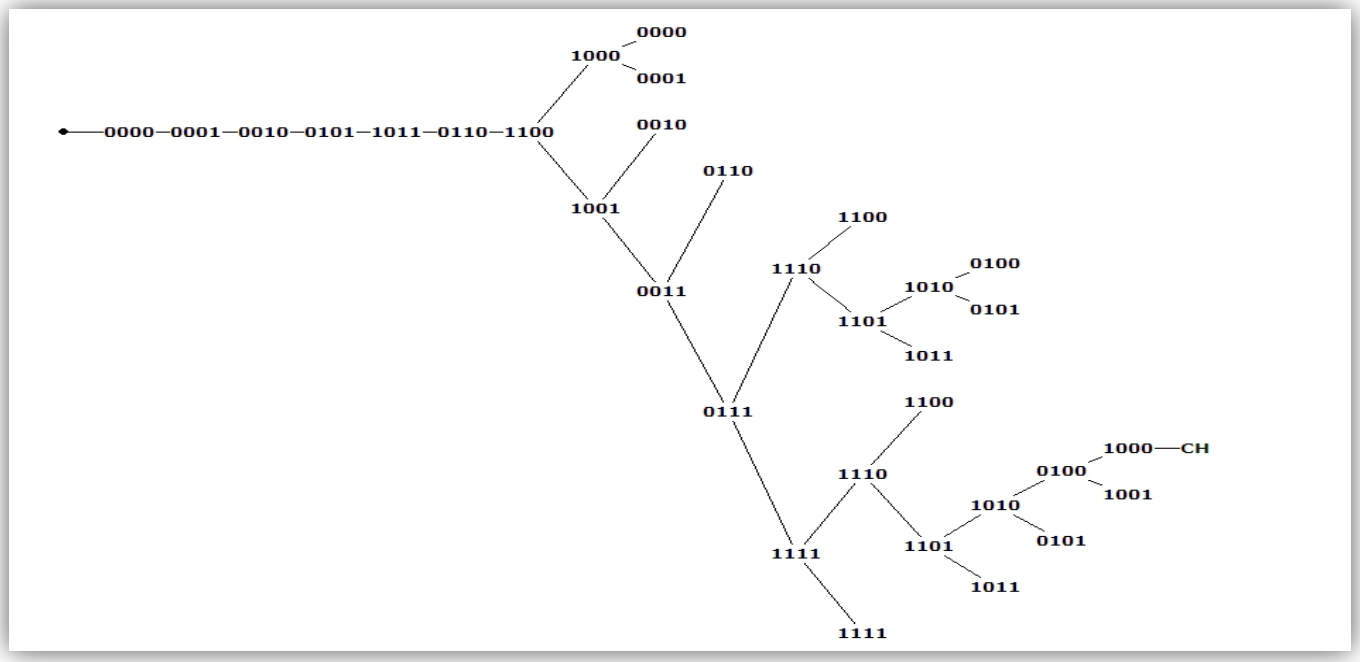
La modélisation sous forme d'arbre avec pour départ 0000 est :

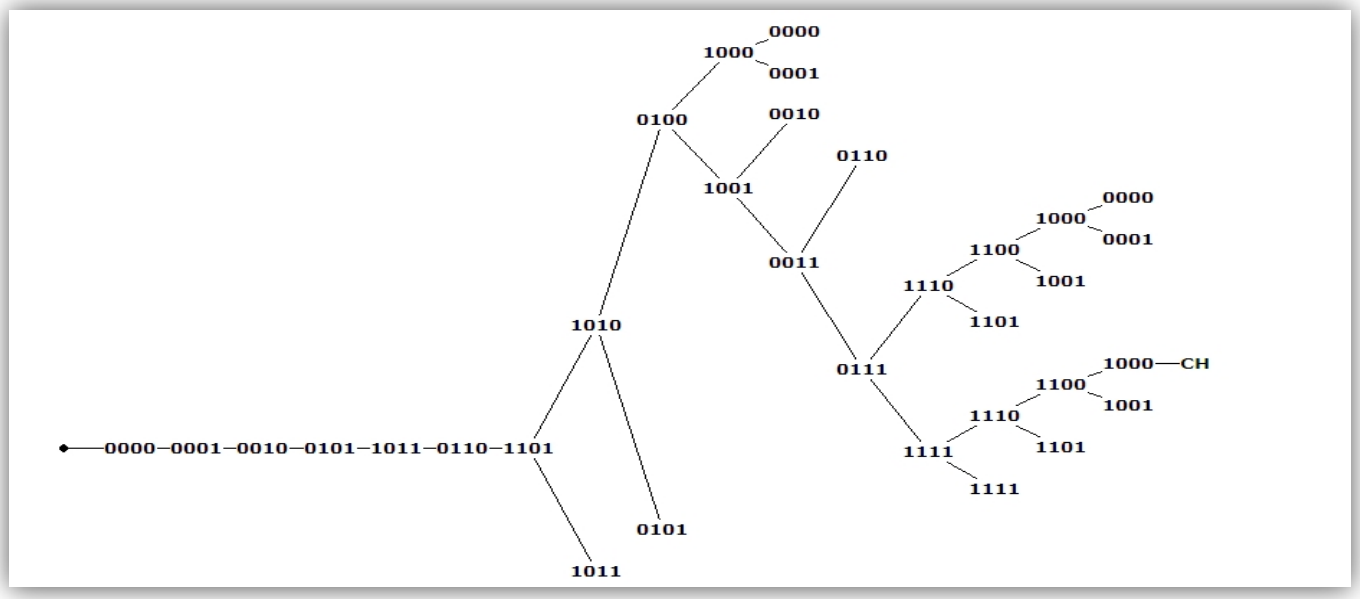


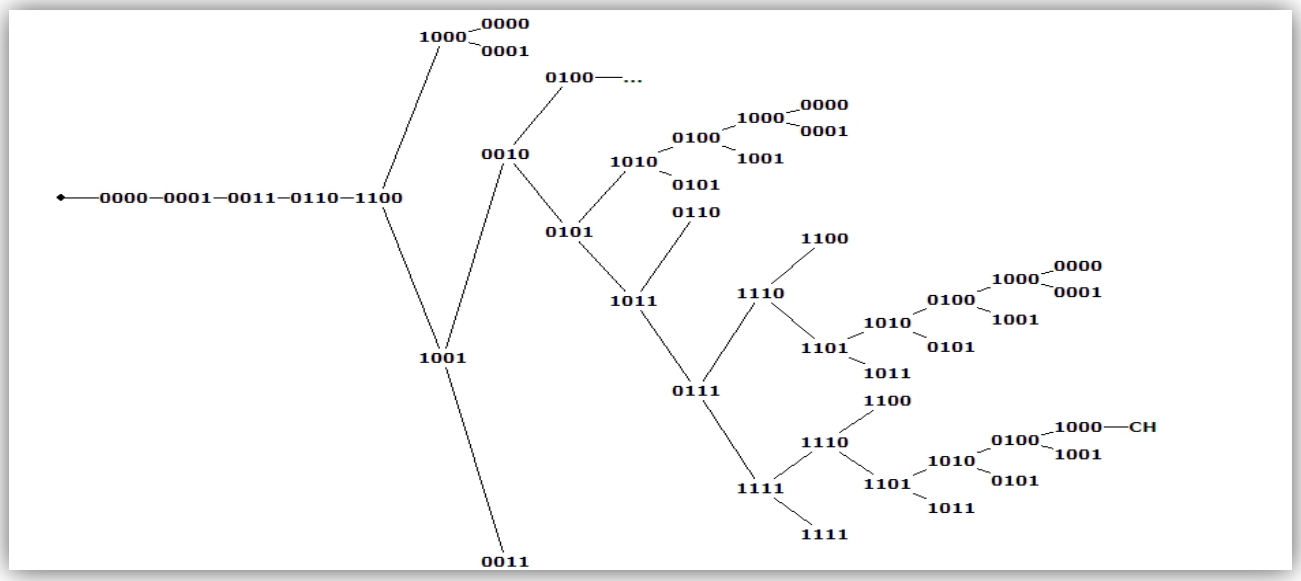

1000—...
1100
1001
1110
1101—...
0000
1000
0001
1100
1001
0100
1010
0100
1010
0101
1110
0101
1000—CH
1100
1001
0110
1101
1011
0111
1101
1111
0000
1000
0001
1100
1001
0110
1101
1011
0111
—0000—0001—0010—0100—1001—0011—0111
1111

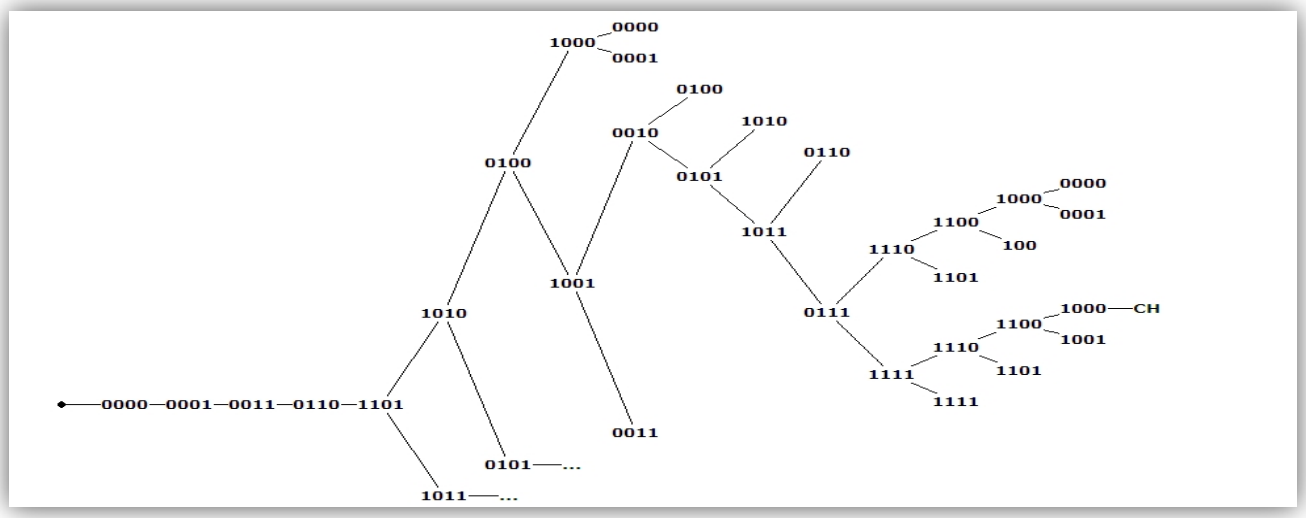

0000
1000
0001
1100
1001
0000—0001—1010—0100—1001—0011—0110
1010
0110
1101
1011
0000
1000
0001
1100
1001
1110
1101
0111
1000—CH
1100
1001
1110
1101
1111
1111

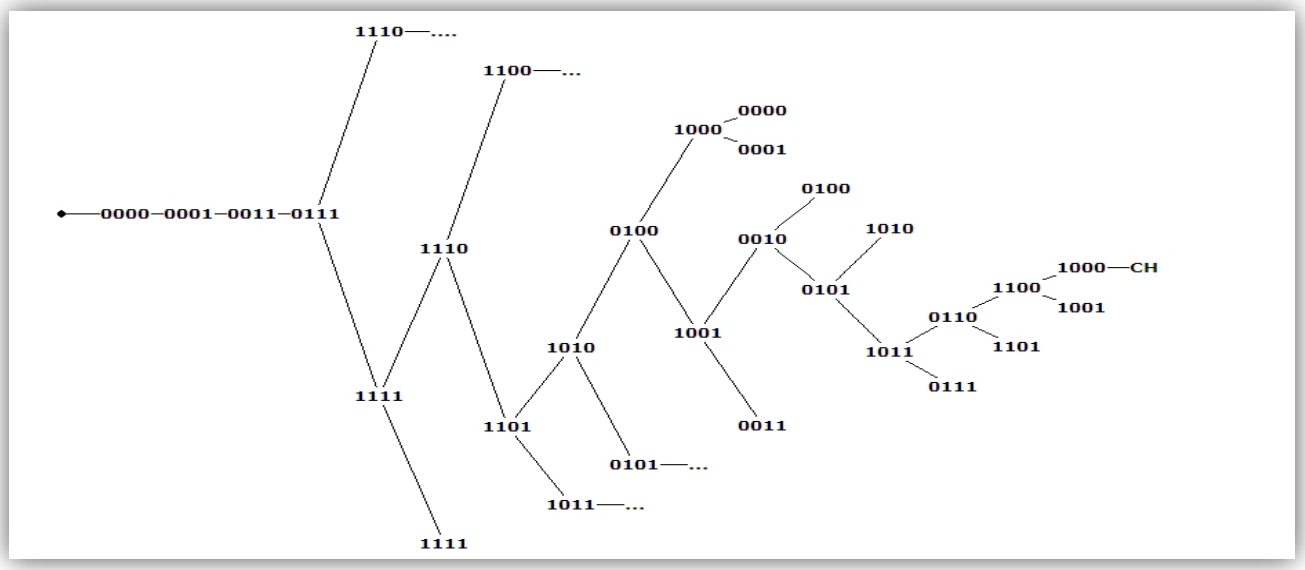

0000
1000
0001
1100
1001
1110
1010
1101
0000
1000
0001
1100
1001
0110
1101
1011
0111
0000—0001—0010—0101—1010—0100—1001—0011—0111
0000
1000
0001
1100
1001
1110
1010
1101
1000—CH
1100
1001
0110
1101
1111
1011
0111
1111


0000
1000
0001
0000–0001–0010–0101–1011–0110–1100
0010
1001
0110
1100
1110
0011
0100
1010
1101
0101
1011
0111
1100
1000—CH
1110
0100
1001
1010
0101
1101
1111
1011
1111


0000
1000
0001
0100
0010
0110
1001
0000
1000
0001
1100
0011
1001
1110
1010
1101
0111
1000—CH
1100
0000–0001–0010–0101–1011–0110–1101
1001
1110
1101
1111
1111
0101
1011


0000—0001—0011—0110—1100
1000
0000
0001
0100—...
0010
1010
0100
1000
0000
0001
1001
0101
0101
0110
1011
1100
1110
1101
1010
0100
1000
0000
0001
1001
0101
1011
0111
1100
1110
1101
1111
1010
0100
1000—CH
1001
0101
1011
1111
0011

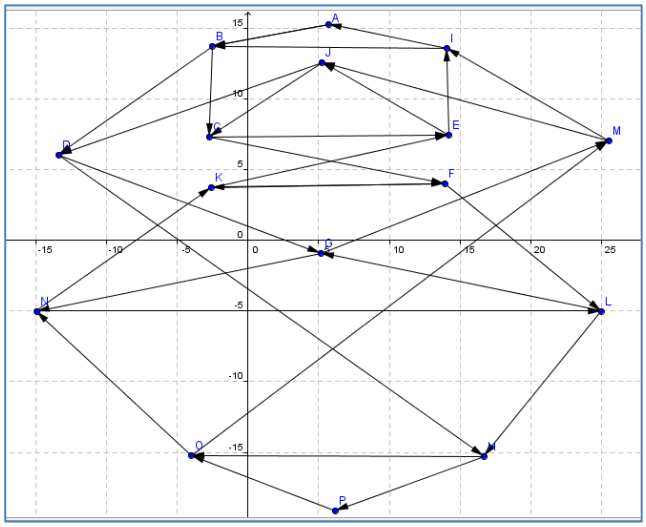

0000—0001—0011—0110—1101
1000
0000
0001
0100
0010
1010
0100
0101
0110
1001
1011
1010
1100
1000
0000
0001
1110
100
1101
0111
1100
1000—CH
1001
1110
1101
1111
0011
0101—...
1011—...


0000—0001—0011—0111
1110—....
1100—...
1000
0000
0001
0100
0100
0010
1010
1110
0101
1000—CH
1100
1001
0110
1001
1010
1011
1101
0111
1111
1101
0011
0101—...
1011—...
1111

A 4 chiffres, la complexité a bien augmenté. En effet, il y a $2^{16} = 65\,536$ chemins à vérifier. C'est impossible à la main. Les circuits hamiltoniens trouvés ne sont peut-être pas les seuls possibles.

Pour tracer les graphes, on a préféré les modéliser sur un support informatique. En effet on ne peut pas déplacer un sommet créé sur une feuille de papier et que toutes les arêtes reliées au sommet bougent en même temps.
Nous avons pensé à utiliser un logiciel simple de géométrie : « geogrebra ».

La modélisation sous forme de graphe est :



Les avantages du logiciel:

- Possibilité de sauvegarder les travaux en cours
- Interface agréable (zoom, déplacement)

Les inconvénients :

- Doubles flèches non distinctes
- Déplacement manuel des sommets pour la clarté du graphe
- Les boucles n'apparaissent pas sur le graphe initial
- Lettres pour les noms des sommets (au lieu des codes)
- Impossible de colorier une arête.

Nous continuerons à utiliser ce logiciel en attendant d'en trouver un plus adéquate.

Les circuits hamiltoniens trouvés pour les nombres binaires à 4 chiffres commençant par 0000 sont :

0000→0001→0010→0100→1001→0011→0110→1101→1010→0101→1011→0111→1111→1110→1100→1000→...
0000→0001→0010→0100→1001→0011→0111→1111→1110→1101→1010→0101→1011→0110→1100→1000→...
0000→0001→0010→0101→1010→0100→1001→0011→0110→1101→1011→0111→1111→1110→1100→1000→...
0000→0001→0010→0101→1010→0100→1001→0011→0111→1111→1110→1101→1011→0110→1100→1000→...
0000→0001→0010→0101→1011→0110→1100→1001→0011→0111→1111→1110→1101→1010→0100→1000→...
0000→0001→0010→0101→1011→0110→1101→1010→0100→1001→0011→0111→1111→1110→1100→1000→...
0000→0001→0011→0110→1100→1001→0010→0101→1011→0111→1111→1110→1101→1010→0100→1000→...
0000→0001→0011→0110→1101→1010→0100→1001→0010→0101→1011→0111→1111→1110→1100→1000→...
0000→0001→0011→0111→1111→1110→1101→1010→0100→1001→0010→0101→1011→0110→1100→1000→...

Il existe 9 possibilités pour chaque départ.
Il y a, pour ce cas, 9 circuits hamiltoniens.

La suite pour départ 0000, exemple 0000100110101111000 est de longueur 19.
Si il n'existe pas d'autres circuits hamiltoniens avec pour départ 0000, en tout, il y a $16 \times 9 = 144$ circuits hamiltoniens donc suites de longueur 19.

Pour la prochaine fois, il s'agira de passer à l'étape juste au-dessus : la base 3. Nous allons commencer à faire des recherches et tracer d'éventuels arbres.

22 Février 2012 :

Au cours de la semaine, nous avons trouvé un nouveau programme pour modéliser les graphes, « graph creator ». Celui est utilisable uniquement sur internet à l'adresse : *http://illuminations.nctm.org/ActivityDetail.aspx?id=20*.

Les avantages du logiciel :

- Représentation des graphes plus agréable
- Outils supplémentaires (surligneur, aide à la vérification de circuits hamiltoniens)
- Grille pour placer les sommets
- Boucles
- Doubles flèches distinctes.

Les inconvénients :

- Petite interface graphique
- Impossibilité de sauvegarder
- Connexion internet requise
- Déplacement manuel des sommets pour la clarté du graphe
- Lettres pour les noms des sommets (mais facilement modifiable avec éditeur d'images).

A partir de maintenant les représentations sous forme de graphes se feront avec ce logiciel.

Nous avons aussi pensé à la base 3 c'est-à-dire les nombres composés de 0, 1 et 2.

Chaque nombre écrit dans la base correspond à un code.

Cette correspondance pour le chiffre en base 3 $a_4a_3a_2a_1$ est donné par la formule

$$a_4 \times 3^3 + a_3 \times 3^2 + a_2 \times 3^1 + a_1 \times 3^0$$

par exemple 012 correspond au code $0 \times 3^3 + 0 \times 3^2 + 1 \times 3 + 2 = 5$.

Dans cette base, nous aurons que le cas avec répétition car comme chaque nombre écrit en base 3 correspond à un code et que dans les nombres écrits en base 3, il y a des répétitions.

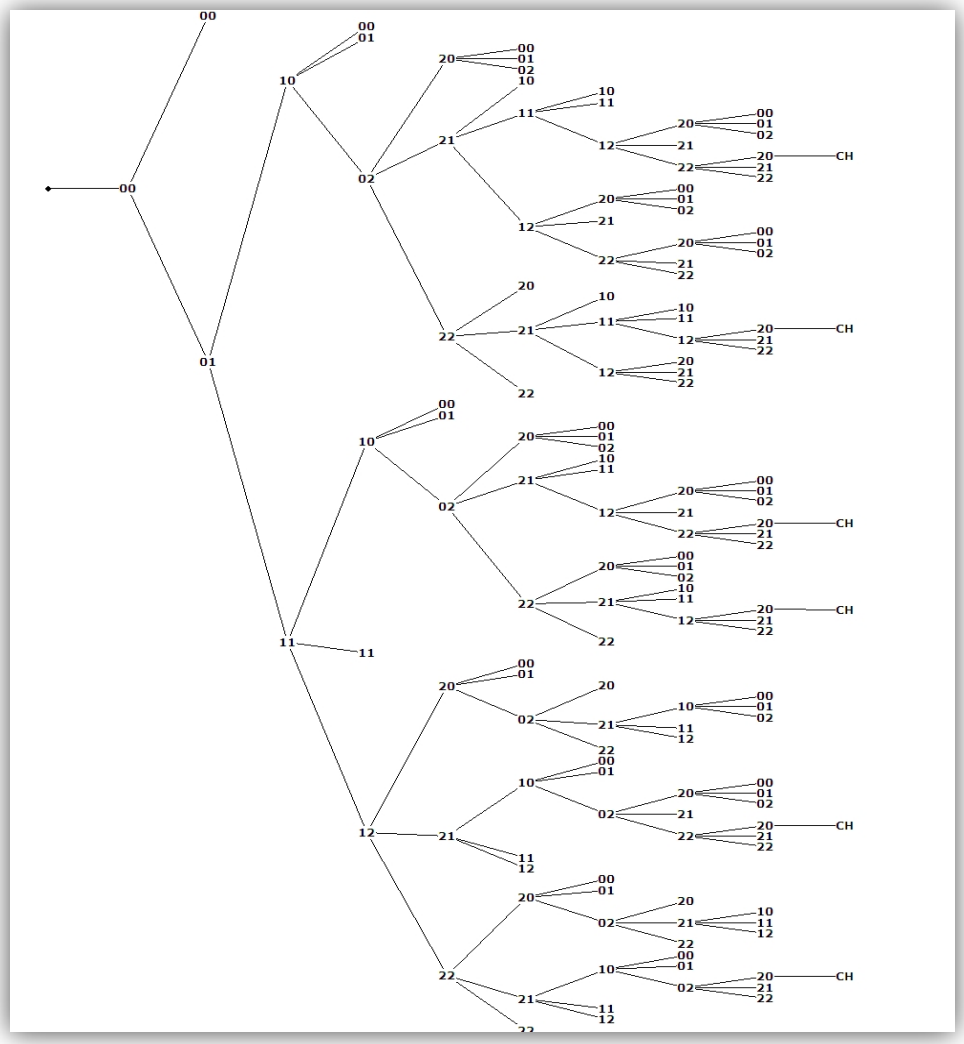
Nous avons commencé simplement avec un code à deux chiffres et nous étudierons un rapide aperçu du code à 3 chiffres pour la compréhension du problème avant de comprendre qu'il n'est pas nécessaire d'aller plus loin, le problème se complique trop et il ne nous apprendra rien de plus.

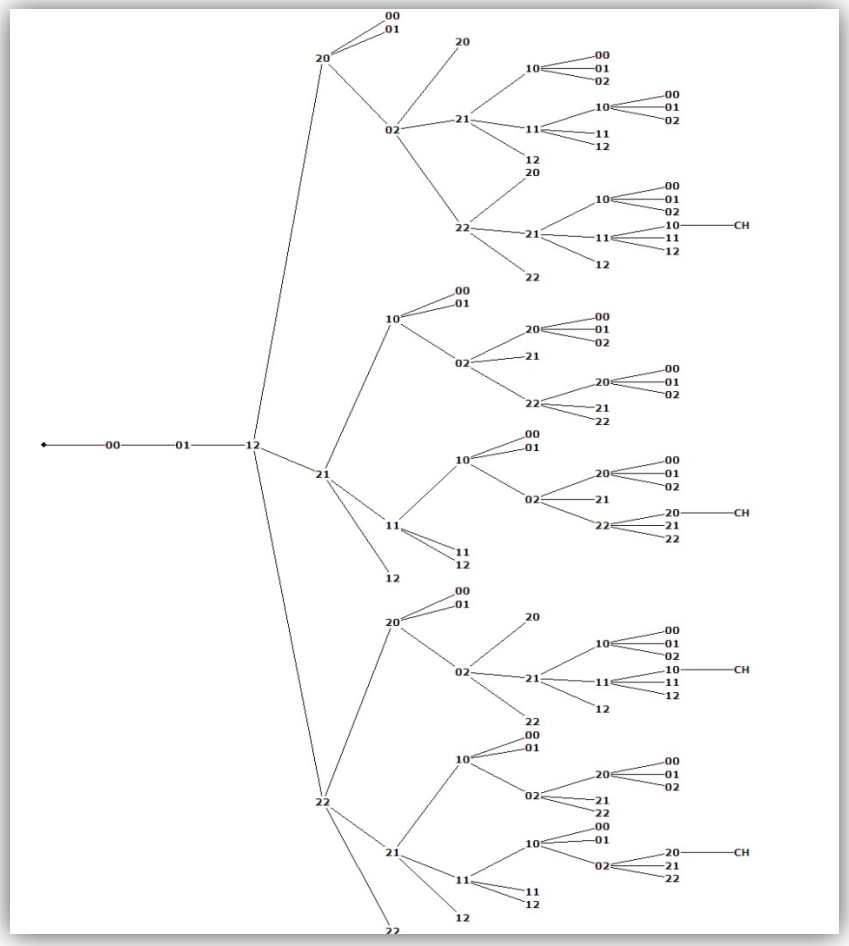
<u>Code à 2 chiffres :</u>

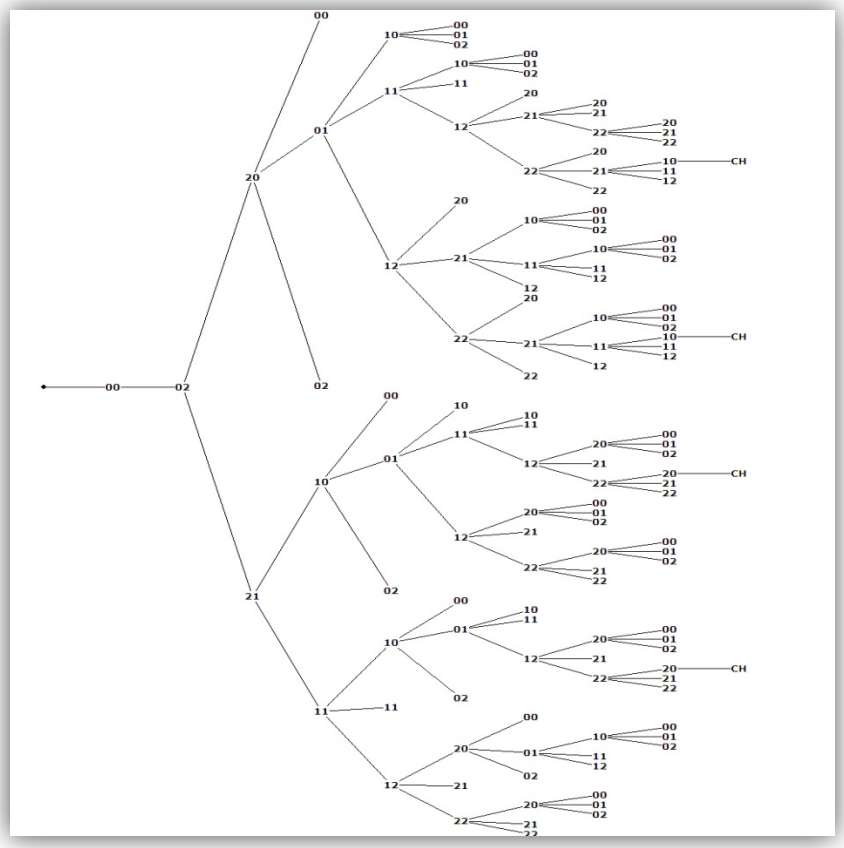
Nous aurons 9 codes : 00, 01, 10, 11, 02, 12, 20, 21, 22.

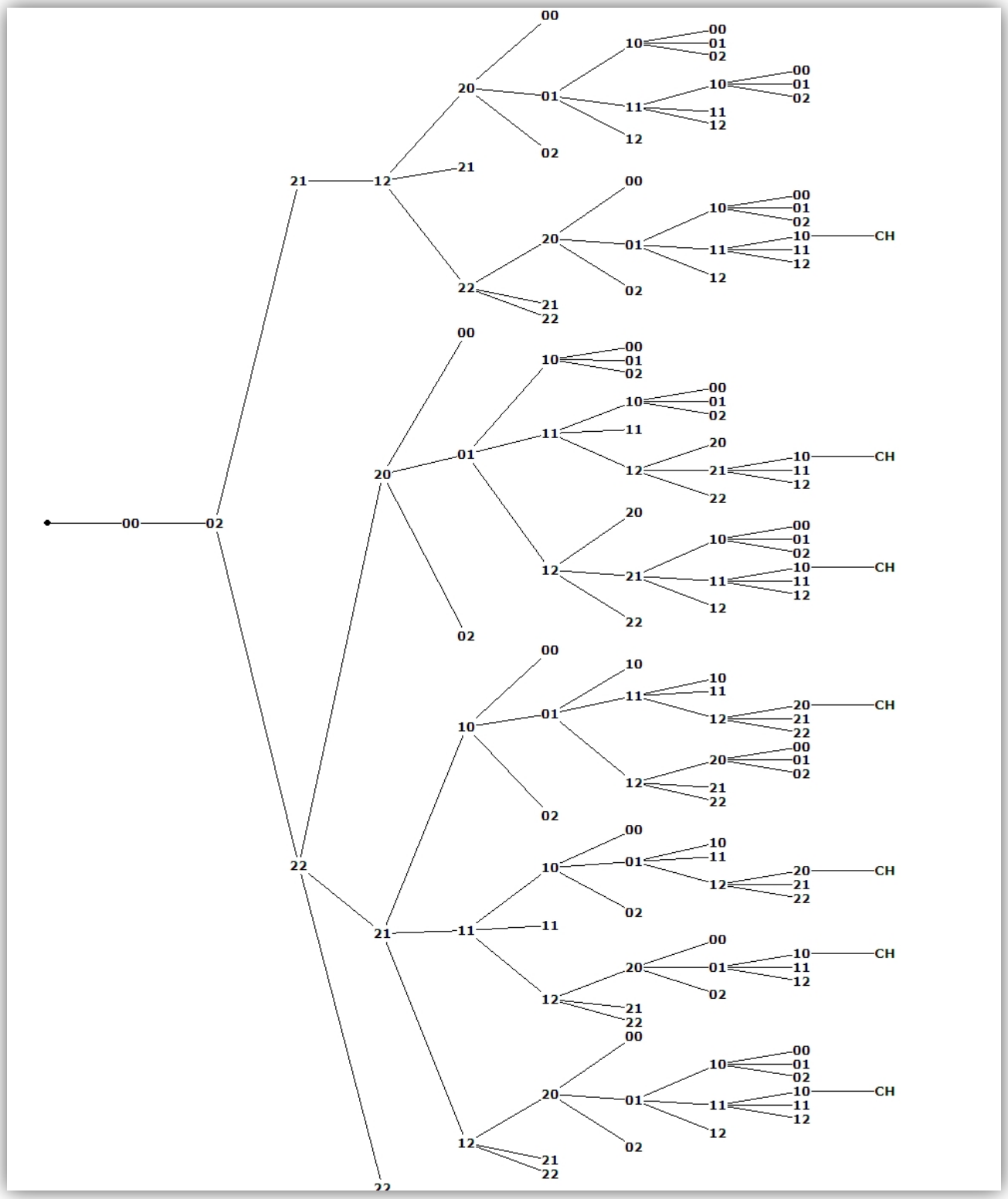
Grâce à la méthode employée en base 2 avec les arbres, on trouve l'ensemble des suites possibles.

Sa modélisation sous forme d'arbre est :

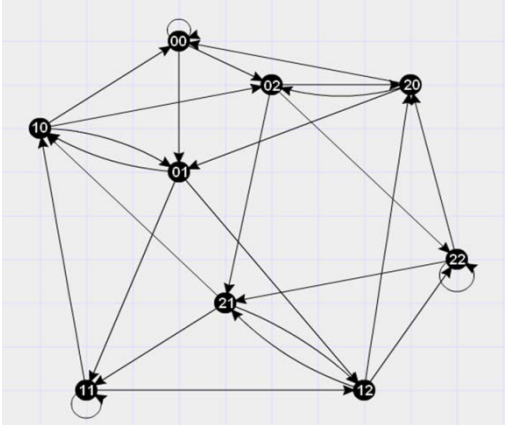



00
01
12
20
21
22
10
11
02
CH


00
02
20
21
01
10
11
12
CH


00
02
21
12
20
00
01
02
10
00
01
02
11
11
12
10
00
01
02
12
21
22
00
01
02
10
00
01
02
11
11
12
10
00
01
02
CH
12
21
22
00
01
02
10
00
01
02
11
11
12
20
21
22
CH
12
21
22
00
01
02
10
00
01
02
11
11
12
10
00
01
02
CH
12
21
22
22
00
01
02
10
00
01
02
11
11
12
20
21
22
CH
12
21
22
00
01
02
10
00
01
02
11
11
12
10
00
01
02
CH
12
21
22
22

Sa modélisation sous forme de graphe est :



On constate que le graphe n'est plus planaire.

Planaire : *Un graphe est dit planaire s'il a au moins une représentation*
Planaire, c'est-à-dire une façon de dessiner un graphe dans le plan de sorte que les arêtes ne se croisent pas.

L'utilisation de « graph creator » est justifiée par le besoin d'un graphe plus esthétique.

Les circuits hamiltoniens des nombres écrits dans la base 3 à 2 chiffres commençant par 00 sont :

00 → 01 → 10 → 02 → 21 → 11 → 12 → 22 → 20 →…
00 → 01 → 11 → 10 → 02 → 21 → 12 → 22 → 20 →…
00 → 01 → 11 → 12 → 20 → 02 → 22 → 21 → 10 →…
00 → 01 → 11 → 12 → 22 → 20 → 02 → 21 → 10 →…
00 → 01 → 12 → 21 → 11 → 10 → 02 → 22 → 20 →…
00 → 01 → 12 → 22 → 21 → 11 → 10 → 02 → 20 →…
00 → 02 → 20 → 01 → 12 → 22 → 21 → 11 → 10 →…
00 → 02 → 21 → 11 → 10 → 01 → 12 → 22 → 20 →…
00 → 02 → 22 → 20 → 01 → 11 → 12 → 21 → 10 →…
00 → 02 → 22 → 21 → 10 → 01 → 11 → 12 → 20 →…
00 → 02 → 22 → 21 → 11 → 12 → 20 → 01 → 10 →…

00 → 01 → 10 → 02 → 22 → 21 → 11 → 12 → 20 →…
00 → 01 → 11 → 10 → 02 → 22 → 21 → 12 → 20 →…
00 → 01 → 11 → 12 → 21 → 10 → 02 → 22 → 20 →…
00 → 01 → 12 → 20 → 02 → 22 → 21 → 11 → 10 →…
00 → 01 → 12 → 22 → 20 → 02 → 21 → 11 → 10 →…
00 → 02 → 20 → 01 → 11 → 12 → 22 → 21 → 10 →…
00 → 02 → 21 → 10 → 01 → 11 → 12 → 22 → 20 →…
00 → 02 → 21 → 12 → 22 → 20 → 01 → 11 → 10 →…
00 → 02 → 22 → 20 → 01 → 12 → 21 → 11 → 10 →…
00 → 02 → 22 → 21 → 11 → 10 → 01 → 12 → 20 →…
00 → 02 → 22 → 21 → 12 → 20 → 01 → 11 → 10 →…

Il y a pour ce cas 22 circuits hamiltoniens pour départ 00.
En tout, il y a $9 \times 22 = 198$ circuits hamiltoniens.

Les combinaisons associées sont :

0010211220	-	0010221120	-	0011021220	-
0011022120	-	0011202210	-	0011210220	-
0011220210	-	0012022110	-	0012110220	-
0012202110	-	0012211020	-	0020112210	-
0020122110	-	0021011220	-	0021101220	-
0021220110	-	0022011210	-	0022012110	-
0022101120	-	0022110120	-	0022112010	-
0022120110					

Toutes les combinaisons sont de longueur 10.

La semaine prochaine on passera enfin au niveau fixé, la base 10. On va réfléchir à une façon d'utiliser notre savoir en base 2 et en base 3 pour l'utiliser dans cette nouvelle base.

29 Février 2012 :

La Base 10 est le système de valeurs utilisé sur une grande majorité des digicodes si ce n'est la totalité.
La complexité de cette base est évidente dès le code à deux chiffres, c'est pourquoi nous avons réfléchi à une autre méthode de résolution. Les arbres sont trop compliqués, inutiles à ce stade, donc nous avons étudié une nouvelle approche : des tableaux.

Cette méthode, permet une résolution beaucoup plus rapide mais a des limites. En effet, on ne trouvera qu'une seule solution (c'est ce que l'on cherche) au lieu de la totalité.

Code à 2 chiffres :

Nous aurons 90 codes.

00		10	89	20	87	30	83	40	77	50	69	60	59	70	47	80	33	90	17
01	88	11		21	85	31	81	41	75	51	67	61	57	71	45	81	31	91	15
02	84	12	86	22		32	79	42	73	52	65	62	55	72	43	82	29	92	13
03	78	13	82	23	80	33		43	71	53	63	63	53	73	41	83	27	93	11
04	70	14	76	24	74	34	72	44		54	61	64	51	74	39	84	25	94	9
05	60	15	68	25	66	35	64	45	62	55		65	49	75	37	85	23	95	7
06	48	16	58	26	56	36	54	46	52	56	50	66		76	35	86	21	96	5
07	34	17	46	27	44	37	42	47	40	57	38	67	36	77		87	19	97	3
08	18	18	32	28	30	38	28	48	26	58	24	68	22	78	20	88		98	1
09	90	19	16	29	14	39	12	49	10	59	8	69	6	79	4	89	2	99	

Nombre noir : code

Nombre rouge : position dans la suite

Nous partons d'un nombre arbitraire, ici 98 il y a donc 9 possibilités pour celui qui suit, nous allons toujours prendre le dernier, ici 9 jusqu'à arriver à la suite 989796959493929190. Si nous suivons le raisonnement pris avant il faudrait prendre le 9 or tous les chiffres commençant par 9 sont déjà utilisé, nous allons donc prendre le suivant, le 8 et continuons en prenant le dernier chiffre qui ne soit pas déjà utilisé.

La suite que l'on obtient est :

989796959493929190878685848382818076757473727170656463626160545352515043424140323130212010 9

Code à 3 chiffres :

Nous commençons à réfléchir au problème et cela sera l'objectif de la prochaine séance.

Nous passons définitivement à la méthode des tableaux pour la base 10 et nous pourrions l'adapter aux petites bases, mais celles-ci ne servaient qu'à la compréhension du problème, nous ne voulons pas faire de retour en arrière.

Pour la semaine prochaine, nous répartissons le travail. Une personne se concentrera sur le code à 3 chiffres, les deux autres chercheront à coder un programme permettant de balayer un fichier contenant nos solutions et trouver le code au sein de cette chaine de nombres.

7 Mars 2012 :

Au cours de la semaine, nous avons travaillé sur ce code à trois chiffres pour finalement arriver à en sortir une suite résultat.

```
9879869859849839829819809789769759749739729719709689679659649639629619609589579569549539529519509489479469459439429419409389379369359349329319019309289279269259249239219209189179169159149139129108908768758748738728718708678658648638628618608578568548538528518508478468458438428418408378368358348328318018308278268258248238218208178168158148138128107907807657647637627617607567547537527517507467457437427417407367357347327317017307267257247237217207167157147137127106906806706546536526516506456436426416406356346326316016306256246236216206156146136126105905805705605435425415405345325315015305245235215205145135125104904804704604504324314014304234214204134124103903803703603503403213013203120123102902802702602502402302109898
```

Nous nous sommes restreints au problème sans répétitions pour limiter le nombre de codes à chercher et avoir une solution plus courte. On réfléchira tout de même à ce que serait le résultat avec des répétitions.

Le programme avait très peu avancé, mais grâce à une aide extérieure d'étudiants en informatique, il est prêt et en phase de test. Il faudra encore l'adapter pour prendre en compte les différentes variables comme le nombre de chiffres et quel fichier lire.
En effet nous voulons un programme compatible avec le code à 3 chiffres et le code à 4 chiffres. A terme ce programme pourrait fonctionner pour toutes les bases et quel que soit le nombre de chiffres du code.

Nous travaillons à la correction de ce programme pendant la séance de TD.

Après un bilan avec nos professeurs, nous cherchons à clarifier les résultats obtenus.
Nous voulons rendre compte de notre gain sous forme de chiffres. Nous allons alors faire des recherches sur l'efficacité de notre méthode de résolution.

14 Mars 2012 :

L'objectif de cette séance et de quantifier l'efficacité de notre résolution :
Qu'advient-il du nombre de tapes sans répétition / avec répétition ?
Nous allons établir la différence en notre méthode et la méthode dite « classique ».
La méthode « classique » consiste à taper tous les codes du plus petit au plus grand (par exemple : en base 10 à 2 chiffres : taper de 00 à 99).
Il y aura des répétitions vu qu'il y a une mémoire de frappe.

Voici le tableau pour les codes à n chiffres écrits en base 2 :

n=	2	3	4
Notre méthode	5 tapes	10 tapes	19 tapes
Méthode classique	8 tapes	24 tapes	64 tapes

Voici le tableau pour les codes à n chiffres écrits en base 3 :

n=	2
Notre méthode	10 tapes
Méthode classique	18 tapes

Voici le tableau pour les codes à n chiffres écrits en base 10 (sans répétition) :

n=	**2**	**3**	**4**	**5**	**10**
Notre méthode	91 tapes	722 tapes	5 043 tapes	30 244 tapes	**3 628 809** tapes
Méthode classique	180 tapes	2 160 tapes	20 160 tapes	151 200 tapes	**36 288 000** tapes

Voici le tableau pour les codes à n chiffres écrits en base 10 (avec répétition) :

n=	**2**	**3**	**4**	**5**	**10**
Notre méthode	101 tapes	1 002 tapes	10 003 tapes	100 004 tapes	**1 000 005** tapes
Méthode classique	200 tapes	3 000 tapes	40 000 tapes	500 000 tapes	**10 000 000** tapes

Nous remarquons que dans chaque base, plus le code est long, plus notre méthode gagne en efficacité.

A partir de ces résultats, nous avons décidé de donner les pourcentages de gain, c'est-à-dire le pourcentage de tapes en moins par rapport à la méthode classique.

Voici le tableau pour les codes à n chiffres écrits en base 2 :

n=	2	3	4
Pourcentage de gain	38 %	58 %	70 %

Voici le tableau pour les codes à n chiffres écrits en base 3 :

n=	2
Pourcentage de gain	44 %

Voici le tableau pour les codes à n chiffres écrits en base 10 (avec ET sans répétition):

n=	**2**	**3**	**4**	**5**	**10**
Pourcentage de gain	50 %	67 %	75 %	80 %	**90 %**

Nous constatons sans surprise qu’en base 10 les pourcentages de gain sont les mêmes avec et sans répétition. En effet il n’y a aucune raison que les valeurs changent, avec répétition, on a simplement plus de codes et la chaine est donc plus longue.

Après discussion avec M. Beddou, nous avons décidé qu’il serait judicieux de créer une maquette représentant un graphe dans l’espace où l’on peut y voir un circuit hamiltonien. Ce support nous aidera à illustrer nos propos lors de notre présentation finale.
Nous devons discuter du budget et du choix des matériaux.

Pour la semaine prochaine on va réfléchir à de nouvelles idées pour notre sujet.

21 Mars 2012 :

Depuis plusieurs semaines, nous avons essayé de trouver une solution en base10 pour un code à 4 chiffres. Nous avons enfin un résultat correspondant à une chaine de 5043 caractères :

98769875987498739872987198709867986598649863986298619860985798569854985 3
98529851985098479846984598439842984198409837983698359834983298319807980 6
98059804980398029801983098279826982598249823982198209817981698159814981 3
98129810978697859784978397829781978097689765976497639762976197609758975 6
97549753975297519750974897469745974397429741974097389736973597349732973 1
97089706970597049703970297019730972897269725972497239721972097189716971 5
97149713971297109687968596849683968296819680967896759674967396729671967 0
96589657965496539652965196509648964796459643964296419640963896379635963 4
96329631960896079605960496039602960196309628962796259624962396219620961 8
96179615961496139612961095879586958495839582958195809578957695749573957 2
95719570956895679564956395629561956095489547954695439542954195409538953 7
95369534953295319508950795069504950395029501953095289527952695249523952 1
95209518951795169514951395129510948794869485948394829481948094789476947 5
94739472947194709468946794659463946294619460945894579456945394529451945 0
94389437943694359432943194089407940694059403940294019430942894279426942 5
94239421942094189417941694159413941294109387938693859384938293819380937 8
93769375937493729371937093689367936593649362936193609358935793569354935 2
93519350934893479346934593429341934093289327932693259324932193089307930 6
93059304930291028912890281027912790278127802710269126902681268026712670 2
61025912590258125802571257025612560251024912490248124802471247024612460 2
45124502410239120912390219023812081238021802371207123702170236120612360 2
16023512051235021502341234023102930193209318931793169315931493129310928 7
92869285928492839281928092789276927592749273927192709268926792659264926 3
92619260925892579256925492539251925092489247924692459243924192409238923 7
92369235923492319208920792069205920492039201923092189217921692159214921 3
92108910879187908765876487638762876187608756875487538752875187508746874 5
87438742874187408736873587348732873187068705870487038702870187308726872 5
87248723872187208716871587148713871287108691869086758674867386728671867 0
86578654865386528651865086478645864386428641864086378635863486328631860 7
86058604860386028601863086278625862486238621862086178615861486138612861 0
85918590857685748573857285718570856785648563856285618560854785468543854 2
85418540853785368534853285318507850685048503850285018530852785268524852 3
85218520851785168514851385128510849184908476847584738472847184708467846 5
84638462846184608457845684538452845184508437843684358432843184078406840 5
84038402840184308427842684258423842184208417841684158413841284108391839 0
83768375837483728371837083678365836483628361836083578356835483528351835 0
83478346834583428341834083278326832583248321830783068305830483028301832 0
83178316831583148312831082918091829081908276827582748273827182708267826 5

826482638261826082578256825482538251825082478246824582438241824082378236
823582348231820782068205820482038201890182308217821682158214821382107910
789178907810769176907681768076547653765276517650764576437642764176407635
763476327631760576047603760276017630762576247623762176207615761476137612
761075917590758175807564756375627561756075467543754275417540753675347532
753175067504750375027501753075267524752375217520751675147513751275107491
749074817480746574637462746174607456745374527451745074367435743274317406
740574037402740174307426742574237421742074167415741374127410739173907381
738073657364736273617360735673547352735173507346734573427341734073267325
732473217306730573047302730173207316731573147312731072917091729071907281
708172807180726572647263726172607256725472537251725072467245724372417240
714072367235723472317206720572047203720179017801723072167215721472137210
691068916890681067916790678167806710659165906581658065716570654365426541
654065346532653165046503650265016530652465236521652065146513651265106491
649064816480647164706453645264516450643564326431640564036402640164306425
642364216420641564136412641063916390638160816380637163706354635263516350
634563426341634063256324632163056304630263016320631563146312631062916091
629061906281628061806271607162706170625462536251625062456243624162406235
623462316205620462036201690168016701623062156214621362105910589158905810
579157905781578057105691569056815680567156705610549154905481548054715470
546154605432543154031403540214025401340154305423542154205413041354120412
541053915390538153805371537053615360534253415340532453215304530213025301
532053145312031253105291509152905190528150815280518052715071527051705261
506152605160524352415240523452315204520352015901580157015601523052145213
521049104891489048104791479047814780471046914690468146804671467046104591
459045814580457145704561456045104391439043814380437143704361436043514350
432143024301432043124310429140914290419042814081428041804270417042614061
426041604251405142504150423142034201320149013901480138014701370146013601
450135014230123042103910389138903810379137903781378037103691369036813680
367136703610359135903581358035713570356135603510349134903481348034713470
346134603451342134503410329130913290129031803190328130813280128032714271
307132701270317032613061326012603160325130513250125031503241324012403210
987

Voici le programme :

```c
/*déclaration des bibliothèques*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/*Fonction préliminaire qui copie un code*/
void copie(char *a,char *b, int lg)
{
        int i;
        for(i=0;i<lg;i++)
                a[i]=b[i];
}

/*Fonction principale qui prend en argument du texte tapé dans le terminal*/
int main(int argc, char *argv[])
{
        /*Déclaration de variables*/
        FILE *f;
        char * buffer,*chaine,c;
        int position=1,i=0, lg=atoi(argv[2]);


        /*Vérification que la recherche est bien tapée dans le terminal*/
        if(argc>4)
        {
                printf("Erreur,le code à rentrer dans le terminal est du type : %s [nom fichier] [nbre chiffres code] [code recherché]",argv[0]);
                exit(EXIT_FAILURE);
        }

        /*Ouverture du Fichier sélectioné*/
        f = fopen(argv[1],"r");

        /*Vérification que le fichier existe et qu'il n'est pas vide*/
        if(f==NULL)
        {
                printf("Erreur lecture fichier.\n");
                exit(EXIT_FAILURE);
        }

        /*Déclaration d'un buffer assez grand pour prendre au maximum une chaine de 5045 caractères*/
        buffer = malloc(5045 * sizeof * malloc);


        /*Parcours de la chaine et impression dans le terminal*/
        c=fgetc(f);
        while(c!=EOF)
        {
                printf("%c",c);
                buffer[i] = c;
                i++;
                c=fgetc(f);
        }

        /*Déclaration d'un chaine*/
        chaine = malloc((lg+1) * sizeof * chaine);

        /*Recherche du code demandé dans la suite de nombres*/
        for(i=0;i<5041;i++)
        {
                copie(chaine,&buffer[i],lg);
                chaine[lg] = '\0';
                if(strcmp(chaine,argv[3])==0)
                {
                        printf("\n%s est le %déme code\n\n",chaine,position);//Affichage de la position du code.
                }
                position++;
        }

        /*Fermeture du fichier et fin du programme*/
        fclose(f);
        return 0;
}
```

Mode d'emploi du programme :

Le programme est intitulé « main.c ». Sont fournis deux fichiers contenant les suites de nombres pour les codes sans répétition :

- Code à 3 chiffres : « 123.txt »
- Code à 4 chiffres : « 1234.txt »

Le programme se lance dans un terminal.

Le code à rentrer est du type :

. /main [nom du fichier à ouvrir] [longueur code] [code à rechercher]

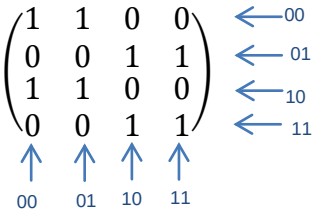
Exemple: *./main 1234.txt 4 2415*

De nouvelles idées sont abordées pour voir si notre sujet peut être traité différemment.
Dans la modélisation sous forme de graphe, nous avons pensé utiliser les poids des arêtes et les matrices d'adjacences.

Poids des arêtes (graphe pondéré) : *C'est un graphe orienté où les arêtes sont étiquetées ou pondérées par les éléments d'un ensemble R. L'élément p(e) de R associé à l'arête e est appelé l'étiquette ou le poids de e.*

Matrice d'adjacence : *Soit G un graphe (orienté ou non) d'ordre n. La matrice d'adjacence de G est la matrice carrée A d'ordre n dont les coefficients non diagonaux a_{ij} sont le nombre d'arêtes liant le sommet i au sommet j. Les coefficients diagonaux a_{ii} sont le nombre de boucles pour le sommet i.*

Nous voulons trouver une autre méthode pour trouver le circuit hamiltonien sans avoir à faire l'arbre.
Par exemple, pour les codes en base 2 à 2 chiffres, nous avons la matrice d'adjacence suivante :



Nous n'avons pas trouvé de manière à modifier la matrice pour trouver le circuit hamiltonien. Nous avons donc abandonné cette idée.

Pour le poids des arêtes, c'est inutile vu que dans notre cas, elles ont toutes la même probabilité donc le même poids.

4 Avril 2012 :

Grace à nos exemples de calculs du nombre de tapes et du pourcentage de gain (cf. 14 Mars), nous avons pu les modéliser avec les formules suivantes :

Si on a un code à n chiffres et de base q, alors on obtient les résultats suivants pour le nombre de tapes :

Nombre de tapes sans répétition (Uniquement pour la base 10):

Avec notre méthode nous trouvons :

$$t_N = (n-1) + \prod_{i=0}^{n-1}(10-i)$$

La méthode « classique » donne ce résultat :

$$t_A = n \times \prod_{i=0}^{n-1}(10-i)$$

Nombre de tapes avec répétition :

Avec notre méthode nous trouvons :

$$t_N = (n-1) + \prod_{i=0}^{n-1} q$$

La méthode « classique » donne ce résultat :

$$t_A = n \times \prod_{i=0}^{n-1} q$$

Dans tous les cas, la relation entre les deux méthodes est :

$$t_A = (t_N - (n-1)) \times n$$

A partir des nombre de tapes, nous avons le pourcentage de gain qui est donné par la formule :

$$p = 100(1 - \frac{t_N}{t_A})$$

Suite à un cours d'informatique avec M. Van Caneghem, nous avons découvert un logiciel intitulé « Graphe à énergie minimale » qui a pour particularité de réarranger automatiquement les sommets d'un graphe connexe pour le rendre plus symétrique (ou esthétique dans le cas échéant).

Les avantages du logiciel :

- Possibilité de sauvegarder les travaux
- Création du graphe à partir d'un fichier texte (plus simple pour paramétrer sommets et arêtes)
- Option permettant une représentation plus esthétique.

L'avantage/inconvénient : Interface très voire trop simple.

Les inconvénients :

- Graphe non orienté obligatoirement
- Disparition des boucles
- Pas de coloration des arêtes
- Impossible de renommer les sommets

La semaine prochaine nous passons la présentation orale, donc dans la semaine nous allons mettre en place le Powerpoint.

11 Avril 2012 :

Pendant la semaine, nous avons terminé notre présentation PowerPoint. Dans un besoin de clarté, nous avons repris chacun des graphes et des arbres et les avons tous mis au format informatique à l'aide de l'ensemble des logiciels précédemment cités.

Nous avons effectué notre présentation devant la classe.
Nous avons noté plusieurs imprécisions et quelques éléments éclaircir. Nous avons déjà quelques idées d'éléments à rajouter pour la présentation finale.
Nous avons rediscuté de la maquette, mise de côté il y a plusieurs semaines (cf.14 Mars) et nous avons eu une réponse concernant le budget.

18 Avril 2012 :

Sur cette séance, écourtée par les présentations des groupes restants, nous avons confirmé les matériaux utilisés pour la maquette, et nous avons parlé des ouvertures possibles.

23 Avril 2012 au 5 Mai 2012:

Nous finalisons la maquette.

Nous avons décidé d'opter pour un cahier commun de recherche au format informatique et avons donc repris l'ensemble du contenu du cahier.

Une légère modification a été nécessaire, les graphes et arbres ne pouvant pas être redessinés à la main, nous avons pris leur équivalent créés avec les logiciels.

Bilan

Nous avons mis au point deux méthodes pour résoudre notre problème qui demandent initialement un ou plusieurs choix arbitraires mais obligatoires :

1) Graphe/arbre : Utilisée pour des petites bases.

A l'aide d'un arbre, on recherche un circuit hamiltonien dans un graphe et on en extrait au moins une suite.

2) Tableau : Utilisée en base 10

A l'aide d'un tableau, en respectant des règles de constructions, on trouve une unique suite.

Nos méthodes se révèlent très efficaces pour toutes les bases et notamment sur des codes très longs.

On a mis au point un moyen de trouver un code désiré dans les suites en bases 10 pour les codes à 3 et 4 chiffres (programme).

La bonne entente dans le groupe nous a permis d'avoir une coordination optimale et donc de bien avancer notre sujet.

Il est clair que malgré notre avancée, le sujet n'est pas clos. En effet, nous avons abandonné plusieurs pistes en cours de route qui mériteraient d'être plus approfondies, et il y en a probablement d'autres auxquelles nous n'avons pas pensé.

Une évolution de notre sujet serait d'ouvrir ces méthodes de résolution à des codes beaucoup plus grands comme des numéros de cartes bleues. On pourrait aussi envisager d'augmenter la base, mais il faudrait certainement une autre méthode de résolution.
On peut trouver d'autres applications telles que le jeu du Mastermind (code à 4 chiffres, les couleurs remplacent les nombres) ou une codification de cartes à jouer pour leur donner un ordre prédéfini.

Sitographie

Bien que nous sommes allés voir quelques livres concernant la théorie des graphes, nous n'avons pas utilisé les ressources qu'il pourrait y avoir dedans et nous avons donc basé nos recherches uniquement sur internet. Ceci est une liste non exhaustive des sites que nous avons parcourus.

Wikipedia :

- http://fr.wikipedia.org/wiki/Digicode
- http://fr.wikipedia.org/wiki/Algorithme_de_Knuth-Morris-Pratt
- http://fr.wikipedia.org/wiki/Algorithme_de_Boyer-Moore

Logiciels:

- Graph creator: http://illuminations.nctm.org/ActivityDetail.aspx?id=20
- Geogebra : http://www.geogebra.org/cms/fr
- Tree Diagramm Generator : http://kera.name/treediag/
- Graphe à énergie minimale : http://ecume.univmed.fr/

Luminy : Cours de Maths Discrètes 2 de Mr. Beffara : http://iml.univ-mrs.fr/~beffara/MD2/cours-2010.pdf

Autres sites :

http://www.iro.umontreal.ca/~hamelsyl/cheminsMinA09_4.pdf

http://www.g-scop.inpg.fr/~rapinec/Graphe/Dijkstra/default.html