Université de Perpignan Via Domitia

Rapport de stage

Maître de stage : Patrick VILAMAJO

Entiers s'écrivant comme somme et produit de n uns

Sofiane BEN HAMMOU
Quoc Duong PHUNG

Perpignan, mai 2020

Remerciement

Nous remercions monsieur Patrick Vilamajo pour la bienveillante attention qu'il a porté à notre travail et pour ses bons conseils qui nous ont souvent tiré d'affaire.
Merci également à notre collègue et ami Félix Ourgaud, qui a pris le temps de nous apporter son aide, malgré un planning déjà très chargé.

FIGURE 1 – **L'éducation d'Achille par le centaure Chiron** (Le centaure Chiron apprit à Achille à tirer à l'arc) [1]

Sommaire

1 Sujet

"Entiers s'écrivant comme somme et produit de n uns"

Pour $n \in \mathbb{N}^*$, on considère l'ensemble E_n des entiers que l'on peut obtenir en utilisant n fois le nombre 1. l'addition, la multiplication et le parenthésage.
Il vous est demandé d'étudier ces ensembles E_n, d'établir des conjectures sur leurs éléments voire même de dégager des propriétés remarquables.

2 Recherche des premiers ensembles de E_n

• $n = 1$, $E_1 = \{1\}$, $Card(E_1) = 1$
• $n = 2$, on a 2 possibilités :
$1 \times 1 = 1$
$1 + 1 = 2$
Donc $E_2 = \{1{,}2\}$, $Card(E_2) = 2$
• $n = 3$:
$1 \times 1 \times 1 = 1$
$1 \times 1 + 1 = 2$ ou $1 + 1 \times 1 = 2$ ou $(1 + 1) \times 1 = 2$ ou $1 \times (1 + 1) = 2$
$1 + 1 + 1 = 3$
Il y a plusieurs façons d'obtenir 2.
Il n'y a pas d'autres possibilités.
Donc $E_3 = \{1{,}2{,}3\}$, $Card(E_3) = 3$
• $n = 4$:
$1 \times 1 \times 1 \times 1 = 1$
$1 \times 1 \times 1 + 1 = 2$ ou $1 \times 1 + 1 \times 1 = 2$ ou $1 + 1 \times 1 \times 1 = 2$
$1 \times 1 + 1 + 1 = 3$ ou $1 + 1 \times 1 + 1 = 3$ ou $1 + 1 + 1 \times 1 = 3$
$1 + 1 + 1 + 1 = 4$
Il y a plusieurs façons d'obtenir 2 et 3.
Il n'y a pas d'autres possibilités.
Donc $E_4 = \{1{,}2{,}3{,}4\}$, $Card(E_4) = 4$
• $n = 5$:
$1 \times 1 \times 1 \times 1 \times 1 = 1$
$1 \times 1 \times 1 \times 1 + 1 = 2$ ou $1 \times 1 \times 1 + 1 \times 1 = 2$ ou $1 \times 1 + 1 \times 1 \times 1 = 2$ ou $1 + 1 \times 1 \times 1 \times 1 = 2$
$1 \times 1 \times 1 + 1 + 1 = 3$ etc.
$1 \times 1 + 1 + 1 + 1 = 4$ ou $(1 + 1) \times (1 + 1) \times 1 = 4$ etc.
$1 + 1 + 1 + 1 + 1 = 5$ ou $(1 + 1) \times (1 + 1) + 1 = 5$ etc.
$(1 + 1) \times (1 + 1 + 1) = 6$ ou $(1 + 1 + 1) \times (1 + 1) = 6$
Donc $E_5 = \{1{,}2{,}3{,}4{,}5{,}6\}$, $Card(E_5) = 6$
• $n = 6$:
$1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1$
$1 \times 1 \times 1 \times 1 \times 1 + 1 = 2$
$1 \times 1 \times 1 \times 1 + 1 + 1 = 3$
$1 \times 1 \times 1 + 1 + 1 + 1 = 4$
$1 \times 1 + 1 + 1 + 1 + 1 = 5$
$1 + 1 + 1 + 1 + 1 + 1 = 6$
$(1 + 1) \times (1 + 1 + 1) + 1 = 7$
$(1 + 1) \times (1 + 1) \times (1 + 1) = 8$
$(1 + 1 + 1) \times (1 + 1 + 1) = 9$
$E_6 = \{1{,}2{,}3{,}4{,}5{,}6{,}7{,}8{,}9\}$, $Card(E_6) = 9$

$\rightarrow$ on voit facilement que $\{1{,}2{,}\ldots{,}n\} \in E_n$

$\bullet\, n = 7$:
$(1+1)\times(1+1)\times(1+1)\times 1 = 8$
$(1+1)\times(1+1)\times(1+1)+1 = 9$
$(1+1)\times(1+1+1+1+1) = 10$ ou $(1+1)\times[(1+1)\times(1+1)+1] = 10$
$(1+1+1)\times(1+1+1+1) = 12$
$E_7 = \{1,2,3,4,5,6,7,8,9,10,12\}$, $Card(E_7) = 11$

$\bullet\, n = 8$:
$(1+1)\times(1+1)\times(1+1)\times 1+1 = 9$
$(1+1)\times(1+1)\times(1+1)+1+1 = 10$
$(1+1)\times(1+1+1+1+1)+1 = 11$
$(1+1+1)\times(1+1+1+1)\times 1 = 12$
$(1+1+1)\times(1+1+1+1)+1 = 13$
$[(1+1)\times(1+1+1)+1]\times(1+1) = 14$
$(1+1+1)\times(1+1+1+1+1) = 15$
$(1+1+1+1)\times(1+1+1+1) = 16$
$(1+1)\times(1+1+1)\times(1+1+1) = 18$
$E_8 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18\}$, $Card(E_8) = 17$

$E_9 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,24,27\}$, $Card(E_9) = 23$

$E_{10} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,26,$
$27,28,30,32,36\}$, $Card(E_{10}) = 30$

$E_{11} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,$
$27,28,29,30,31,32,33,34,35,36,37,38,39,40,42,45,48,54\}$, $Card(E_{11}) = 44$

$E_{12} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,$
$27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,48,49,50,51,52,$
$54,55,56,57,60,63,64,72,81\}$, $Card(E_{12}) = 60$

.
.
.
etc.

On voit d'après les premiers résultats que :
1. $\{1,2,3,\ldots,n\} \in E_n$
2. $E_m \subset E_n, \forall\ m < n$
3. $x \in E_{n-1} \Rightarrow x+1 \in E_n$
4. $x \in E_{n-m} \Rightarrow x+m \in E_n,\ \forall\ m : 0 \leqslant m < n$

On peut faire attention au sous-ensemble F_n de E_n constitué d'entiers consécutifs depuis 1 :
$F_2 = \{1,2\}$
$F_3 = \{1,2,3\}$
$F_4 = \{1,2,3,4\}$
$F_5 = \{1,2,3,4,5,6\}$
$F_6 = \{1,2,3,4,5,6,7,8,9\}$
$F_7 = \{1,2,3,4,5,6,7,8,9,10\}$
$F_8 = \{1,2,3,4,5,6,7,8,9,10,\ldots,15,16\}$
$F_9 = \{1,2,3,4,5,6,7,8,9,10,\ldots,20,21\}$
$F_{10} = \{1,2,3,4,5,6,7,8,9,10,\ldots,21,22\}$
$F_{11} = \{1,2,3,4,5,6,7,8,9,10,\ldots,39,40\}$
$F_{12} = \{1,2,3,4,5,6,7,8,9,10,\ldots,45,46\}$
.
.
.
etc.

L'examen d'éléments de E_n est difficile quand n monte, parce que plus n monte plus card(E_n) monte et des éléments peuvent être constitués par des parenthésages complexes.
Pour continuer le stage, nous avons eu l'idée de :
- Écrire un programme C pour dénombrer tous des éléments de E_n.
- Chercher le plus petit n tel que E_n contienne x.
- Chercher la formule de max(E_n).

3 Réflexion préliminaire à la présentation de l'algorithme

En écrivant le programme C, nous avons été bloqué car le parenthésage rendait le travail très compliqué. Grâce aux conseils de notre maître de stage, nous avons pu résoudre ce blocage en utilisant la notation polonaise inverse (NPI).

La notation polonaise inverse (NPI) (en anglais RPN pour Reverse Polish Notation), également connue sous le nom de notation post-fixée, permet d'écrire de façon non ambiguë les formules arithmétiques sans utiliser de parenthèses. Dérivée de la notation polonaise présentée en 1924 par le mathématicien polonais Jan Łukasiewicz, elle s'en différencie par l'ordre des termes, les opérandes y étant présentés avant les opérateurs et non l'inverse. [2]
Par exemple :
$3 \times (4+7) \to 3\ 4\ 7\ +\ \times$
$2 \times (11+9) \div 5 - 6 \to 2\ 11\ 9\ +\ \times\ 5\ \div\ 6\ -$
Par cette notation, on peut éliminer complètement le parenthésage, mais la priorité n'a pas changé.

En effet, notre problème est un cas spécial de notation polonaise inverse, où on a juste une constante "1" et 2 opérateurs + et ×, cela rend le cas plus simple.

Par exemple :
$[(1+1)(1+1+1)+1] \times \{(1+1) \times [(1+1) \times (1+1)]\}$
$\to 1\ 1\ +\ 1\ 1\ +\ 1\ +\ \times\ 1\ +\ 1\ 1\ +\ 1\ 1\ +\ 1\ 1\ +\ \times\ \times\ \times$
$\Rightarrow$ Pour n uns on a toujours $n-1$ opérateurs soit + soit ×
Par cette approche, notre problème devient un problème de recherche sur toutes les notations polonaises possibles pour n uns !
Par example pour $n = 3$:
$1\ 1\ 1\ \times\ \times = 1$
$1\ 1\ \times\ 1\ \times = 1$
$1\ 1\ 1\ +\ \times = 2$
$1\ 1\ 1\ \times\ + = 2$
$1\ 1\ +\ 1\ \times = 2$
$1\ 1\ \times\ 1\ + = 2$
$1\ 1\ 1\ +\ + = 3$
$1\ 1\ +\ 1\ + = 3$
Donc $E_3 = \{1,2,3\}$

Dans l'algorithme, nous avons utilisé une structure de "pile" pour résoudre une expression polonaise.
Le détail de la programmation est en annexe.

4 Conjecture 1 : le plus petit n tel que E_n contienne x

Soit x un entier. Le plus petit η tel que E_η contienne x, est inférieur ou égal à n , où n est définit ainsi :

Supposons $x \in E_n$, $x \neq \{0{,}1\}$
Soit x un entier non premier, on peux décomposer x en facteurs premiers :
$x = x_1^{q_1} \times x_2^{q_2} \times x_3^{q_3} \times ... \times x_m^{q_m}$
Soit x un entier premier, on peux décomposer $x - 1$ en facteurs premiers :
$x - 1 = x_1^{q_1} \times x_2^{q_2} \times x_3^{q_3} \times ... \times x_m^{q_m}$
Tel que :
$x_1,x_2,x_3,\ldots,x_m$ sont premiers
$x_1 \in E_a$ (a est le plus petit tel que $x_1 \in E_a$, parce que pour $\alpha > a, x_1 \in E_\alpha$ aussi)
$x_2 \in E_b$ (b est le plus petit tel que $x_2 \in E_b$)
$x_3 \in E_c$ (c est le plus petit tel que $x_3 \in E_c$)
.
.
.
$x_m \in E_k$ (k est le plus petit tel que $x_m \in E_k$)
$a,b,c,\ldots,k < n$

Donc :
Pour x non premier : $n = a \times q_1 + b \times q_2 + c \times q_3 + \ldots + k \times q_m$
Pour x premier : $n = a \times q_1 + b \times q_2 + c \times q_3 + \ldots + k \times q_m + 1$

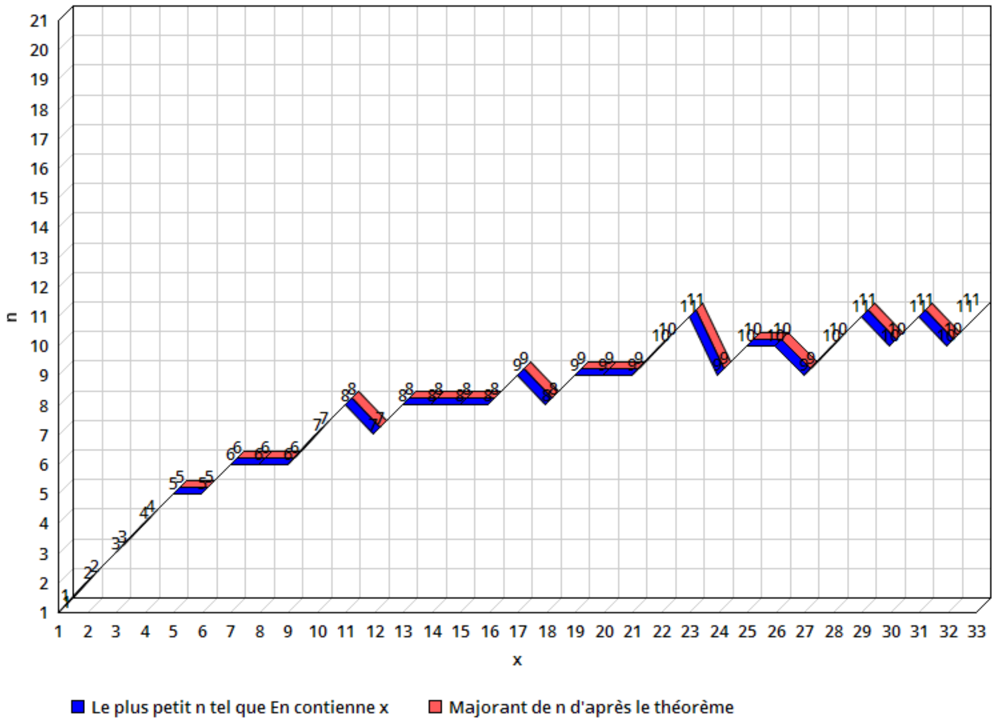
$\rightarrow$ on peux constituer un élément $x \in E_n$ par les éléments du sous-ensemble de E_n.
En plus, $x_1,x_2,x_3,\ldots,x_m$ qui appartiennent au plus petits $E_a,E_b,E_c,\ldots,E_k$ donc x appartient au plus petit E_n. C'est à dire n est le nombre minimum de 1 qu'il faut pour obtenir x.

Par exemple :

1) $x = 63$ non premier : $x = 63 = 3^2 \times 7 \Rightarrow q_1 = 2,\ q_2 = 1$
$3 \in E_3,\ 7 \in E_6 \Rightarrow a = 3,\ b = 6$
donc $n = a \times q_1 + b \times q_2 + c \times q_3 + \ldots + k \times q_m = 3 \times 2 + 6 = 12$
$\rightarrow$ 12 est le nombre minimum de 1 qu'il faut pour obtenir x.

2) $x = 43$ premier : $x - 1 = 43 - 1 = 2 \times 21$
$2 \in E_2,\ 21 \in E_9$
donc $n = a \times q_1 + b \times q_2 + c \times q_3 + \ldots + k \times q_m + 1 = 2 \times 1 + 9 \times 1 + 1 = 12$
$\rightarrow$ 12 est le nombre minimum de 1 qu'il faut pour obtenir x.

Commentaire :
Après vérification, nous avons trouvé 4 cas où $\eta < n$ c'est 46, 47, 55, 82.
Par exemple : $46 = 2 \times 23$, $2 \in E_2$ et $23 \in E_{11}$, donc $n = 2 \times 1 + 11 \times 1 = 13$, mais on a besoin juste de 12 uns pour constituer 46.



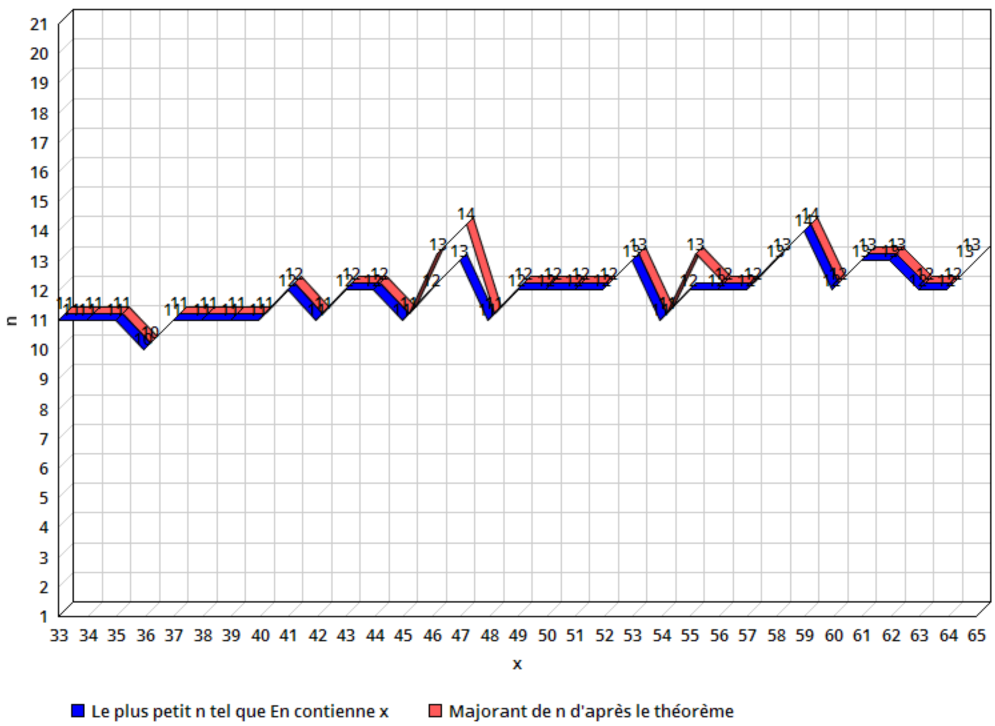


FIGURE 2 – Graphique représentant les plus petits E_n contenant x, en fonction de x.

5 Conjecture 2 : la formule de $max(E_n)$

$\forall\ n \geqslant 4$, $max(E_n) \geqslant M$, où M est défini ainsi :
$n-1 = 3q \rightarrow M = 3^{q-1} \times 2 \times 2$
$n-1 = 3q+1 \rightarrow M = 3^{q} \times 2$
$n-1 = 3q+2 \rightarrow M = 3^{q+1}$

Voici le calcul du maximum des premiers E_n :

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$max(E_n)$	1	2	3	4	6	9	12	18	27	36	54	81	108	162	243

En effet :
$n = 4,\ q = 1 \rightarrow max(E_4) = M = 3^{1-1} \times 2 \times 2 = 4$
$n = 5,\ q = 1 \rightarrow max(E_5) = M = 3^{1} \times 2 = 6$
$n = 6,\ q = 1 \rightarrow max(E_6) = M = 3^{1+1} = 9$
$n = 7,\ q = 2 \rightarrow max(E_7) = M = 3^{2-1} \times 2 \times 2 = 12$
$n = 8,\ q = 2 \rightarrow max(E_8) = M = 3^{2} \times 2 = 18$
etc.

Commentaire :
Nous avons vérifié que $max(E_n) = M$ jusqu'à $n = 30$. Mais nous ne savons pas si cette égalité subsiste par la suite.

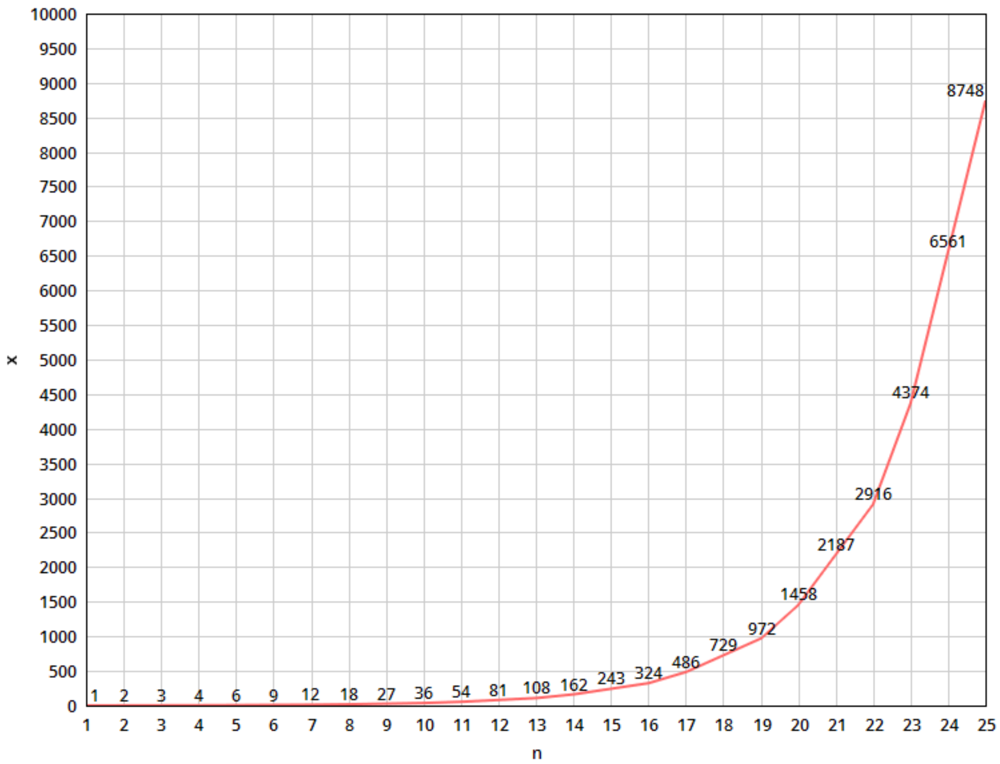


Figure 3 – Graphique représentant le plus grand nombre de E_n, en fonction de n.

6 Conclusion

- Notre sujet concerne les domaines mathématiques comme l'arithmétique, l'algèbre, les algorithmes, et les mathématiques discrètes. C'est pourquoi nous avons pu approcher la recherche de différentes façons , c'est d'ailleurs peut-être l'idée prédominante de ce stage.
- C'est cette impression que nous donne la notation polonaise. Si l'on change le point de vue, nous obtenons alors une idée complètement nouvelle. Nous avons donc plusieurs solutions pour un problème.
- Nous aurions pu aussi examiner $Card(E_n)$ et le sous-ensemble F_n, . Malheureusement, nous ne pourrons en dire plus car notre connaissance n'est pas suffisante pour accentuer cette recherche. Comme Ludwig Wittgenstein l'a dit : "Sur ce dont on ne peut parler, il faut garder le silence".[3]

7 Difficultés rencontrées

- En cherchant à résoudre ce problème, nous nous sommes heurtés à plusieurs difficultés. Tout d'abord, lors de la recherche des règles des éléments d'ensemble E_n, il a été très compliqué d'expliquer les éléments occasionnellement manquant. Puis pour ce sujet il a fallut également revoir des connaissances précédemment étudiés.
- Pour finir, le confinement causé par le virus Covid-19 a été un frein important pour notre travail en binôme, en effet, nous n'avons pas pu nous rencontrer pour traiter ce sujet en direct. Mais ceci n'est pas le plus important, la sécurité de tous étant prioritaire et essentielle, nous avons essayé de traiter ce problème à distance. (Ceci aura été une expérience enrichissante, qui mènera sans doute à une nouvelle tendance de travail).

8 Notes

[1] Regnault, Jean-Baptiste : *L'éducation d'Achille par le centaure Chiron*, 1782, Louvre, Paris.
[2] Wikipedia, Notation polonaise inverse.
[3] Wittgenstein, Ludwig (trad. Gilles Gaston Granger) : *Tractatus logico-philosophicus*, éd. Gallimard Tel, 1993 (ISBN 2-07-075864-8), p. 112, Proposition 7.

9 Annexe

Programme C++

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

class DISQUE //un element de la pile
{    private:
     int _valeur;
     DISQUE* _suivant;
     ~DISQUE(){}
     DISQUE(int n=0){ _valeur=n; _suivant= 0;}

     public:
     static void Empiler (DISQUE ** tete, int n);
     static void Depiler (DISQUE ** tete, int * n);
     static void Depiler (DISQUE ** tete); //depiler tout
     //
     int GetValeur() { return(_valeur); }
};

void DISQUE::Empiler (DISQUE ** tete, int n)
{    DISQUE * nouveau = new DISQUE(n);
     nouveau->_suivant=*tete;
     *tete=nouveau;
}

void DISQUE::Depiler (DISQUE ** tete, int * n)
{    if (!*tete) {*n = 0; return; } //erreur
     DISQUE * adetruire = *tete;
     *n = adetruire->_valeur;
     *tete=adetruire->_suivant;
     delete(adetruire);
}

void DISQUE::Depiler (DISQUE ** tete)
{    int n;
     while (*tete) Depiler(tete,&n);
}

//Cette fonction resoud une formule polonaise,
// en cas d'erreur elle renvoie 0
int Polonaise (char* formule, int longueur)
{    DISQUE * pile = 0; int i, m, n; char caractere;
```

```
    for (i=0;i<longueur;i++)
    {   caractere=formule[i];
        if (caractere=='1') DISQUE::Empiler(&pile,1);
        else
        {   DISQUE::Depiler(&pile,&m);
            DISQUE::Depiler(&pile,&n);
            if(!m||!n) return(0); //erreur
            if (caractere=='+') DISQUE::Empiler(&pile,m+n);
            if (caractere=='*') DISQUE::Empiler(&pile,m*n);
        }
    }
    return((*pile).GetValeur());
    DISQUE::Depiler(&pile);
}

//La classe liste pour les resultats
class LISTE
{   private:
    int * _liste;
    int _nbtotal;
    int _nbpresent;

    public:
    LISTE(int n=1000)
    {   _nbtotal=n;
        _nbpresent=0;
        _liste = new int[_nbtotal];
    }

    ~LISTE(){delete[] _liste;}

    bool DansListe (int n)
    {   for (int i=0;i<_nbpresent;i++)
            if (n==_liste[i]) return(true);
        return(false);
    }

    void Ajout (int n)
    {   if (_nbpresent >= _nbtotal)
        {   int * nouvelle;
            _nbtotal+=1000;
            nouvelle = new int[_nbtotal];
            for(int i=0;i<_nbpresent;i++)
            nouvelle[i]=_liste[i];
            delete[] _liste;
            _liste=nouvelle;
```

```
        }
        int i=0,j;
        while((_liste[i]<n)&&(i<_nbpresent)) i++;
        for (j=0;j<_nbpresent-i;j++)
            _liste[_nbpresent-j]=_liste[_nbpresent-j-1];
        _liste[i]=n;
        _nbpresent++;
    }

    void AfficherListe (FILE * resultats)
    {   for(int i=0;i<_nbpresent;i++)
            fprintf(resultats,"%d, ",_liste[i]);
        fprintf(resultats,"\n");
    }
};

long long boucles = 0;
//elle compte le nombre de boucles que fait l'algorithme

//Cette fonction place les operateurs de differentes manieres
// dans la formule et retient les nouveaux nombres formes ainsi
void Combinaisons(int i,int n,char * formule,char *
 operateurs,char * positions,LISTE * liste)
{   int j, min, resultat;boucles++;
    if (i==n-2)
    {   for (j=0;j<2*n-2;j++) formule[j]='1';
        formule[n*2-2]=operateurs[n-2];
        for (j=0;j<n-2;j++) formule[positions[j]+2]=operateurs[j];
        resultat=Polonaise(formule,2*n-1);
        if (resultat && !liste->DansListe(resultat))
 liste->Ajout(resultat);

    }
    else
    {   if (i==0) min=0; else min=positions[i-1]+1;
        for (positions[i]=min;positions[i]<2*n-4;positions[i]++) Combinaisons(i+1,n,formule,operateurs,positions,liste);
//Chaque passage par la fonction donne la position d'un operateur
    }
}

int main()
{   int n, i, j, compte, temps=0;
    printf("Entrez le nombre de 1 : ");
    scanf("%d", &n);
    char * formule = new char[2*n-1];
```

```
    char * operateurs = new char[n-1];
    char * positions = new char[n-2];
//la position des operateurs dans la formule
    LISTE * liste = new LISTE(1000);
    for(compte=0;compte<pow(2,n-1);compte++)
//parcours de toutes les combinaisons d'operateurs possibles
    {   i=(int(pow(2,n-1))-compte)*10/int(pow(2,n-1));
        if(i != temps) { temps=i; printf("%d\n",temps);}
        for(i=0;i<n-1;i++)
        {   j=compte/pow(2,i);
            if (j%2==0) operateurs[i]='+';
            else operateurs[i]='*';
        }
    Combinaisons(0,n,formule,operateurs,positions,liste);
    }
    delete[] formule;
    delete[] operateurs;
    delete[] positions;
    FILE *resultats = fopen("resultats.txt", "w");
fprintf(resultats,"Voici la liste des resultats possibles avec %d uns:\n",n)
    liste->AfficherListe(resultats);
    delete liste;
    fprintf(resultats,"Nombre de boucles : %lld\n",boucles);
    fclose(resultats);
    return(0);
}
```