Règles de Golomb

Evelyne Le Bezvoët

N°28095

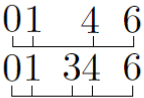
Définition

Règle de Golomb

Une règle de Golomb est une règle où chaque paire de marques mesure une distance différente des autres.

Ordre

Nombre de graduation d'une règle de Golomb



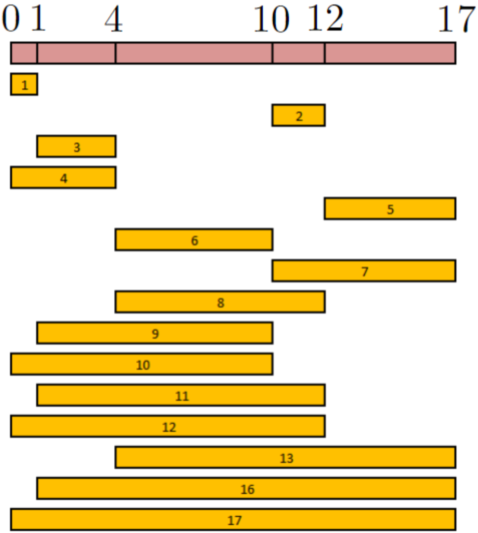


Figure 1 – https ://datagenetics.com/blog/february22013/17.png

Définition

Règle de Golomb optimale

Une règle de Golomb optimale est une règle de Golomb dont la taille pour un ordre donné est minimale.

Exemple : Règles d'ordre 5 non optimale et optimale

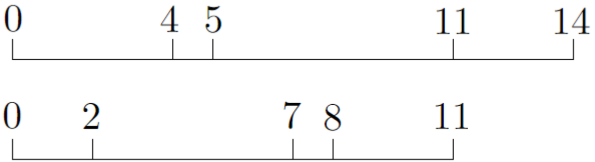


Figure 2 – Règles d'ordre 5

Définition bis

Règle de Golomb

Soit $m, n \in \mathbb{N}$, et soit $f : [m-1] \to [n]$ injective, avec $f(0) = 0$, $f(m-1) = n$.
f est une règle de Golomb de taille n et d'ordre m ssi

$$\forall i, j, k, l \in [m-1], f(i) - f(j) = f(k) - f(l) \Rightarrow i = k \text{ et } j = l$$

Santé et prévention

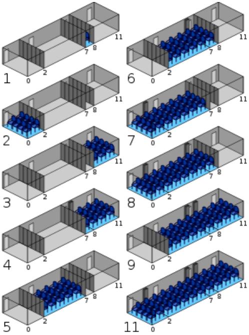
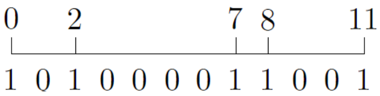


Figure 3 – https ://en.wikipedia.org/wiki/Golomb_ruler#/media/File :Golomb_ruler_conference_room.svg

Optimiser les coûts des nettoyages obligatoires après chaque conférence et gérer les contraintes d'aération.

Fonctions `liste` et `dist`

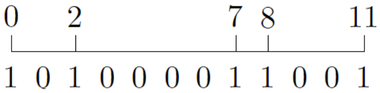


- `liste` : nombre $\rightarrow$ règle
- `dist` : nombre $\rightarrow$ liste des distances lorsque aucune d'entre elles ne se répète, renvoie `False` sinon

Exemple :

- Ecriture binaire de 2437 : 100110000101_2
- `liste(2437)` affiche `[0, 2, 7, 8, 11]`
- `dist(2437)` renvoie `[1, 2, 3, 4, 5, 6, 7, 8, 9, 11]`

Fonction golomb



- `golomb` : ordre o → règles optimales d'ordre o

Exemple :

- `golomb(5)` renvoie

```
|.|....||..| [0, 2, 7, 8, 11]
||..|....|.| [0, 1, 4, 9, 11]
|..||....|.| [0, 3, 4, 9, 11]
|.|....|..|| [0, 2, 7, 10, 11]
```

Variables : entier `x` ; liste d'entiers `r`

Définir fonction `golomb(o)`

Tant qu'une règle de golomb n'est pas trouvée

Si `liste(x)` est d'ordre o **et** `dist(x)` $\neq$ `False`

On rajoute `x` à `r`

On augmente `x` de 2

Tant que `liste(x)` est de même taille que la règle optimale

Si `liste(x)` est d'ordre o **et** `dist(x)` $\neq$ `False`

On rajoute `x` à `r`

On augmente `x` de 2

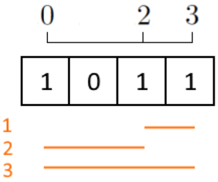
Renvoyer les règles trouvées

golomb(3)

x	bin(x)	liste(x)	dist(x)	Règle
1	0_2	[0]	[]	0
3	11_2	[0, 1]	[1]	0 1
5	101_2	[0, 2]	[2]	0 2
7	111_2	[0, 1, 2]	False	0 1 2
9	1001_2	[0, 3]	[3]	0 3
11	1011_2	[0, 1, 3]	[1, 2, 3]	0 1 3
13	1101_2	[0, 2, 3]	[1, 2, 3]	0 2 3
15	1111_2	[0, 1, 2, 3]	False	0 1 2 3

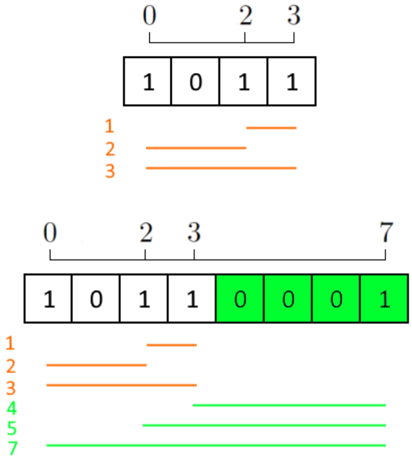
2ème algorithme : Shift Algorithm

Rajouter une graduation :

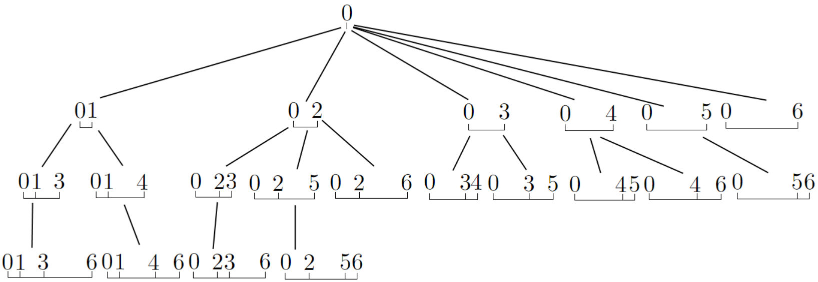


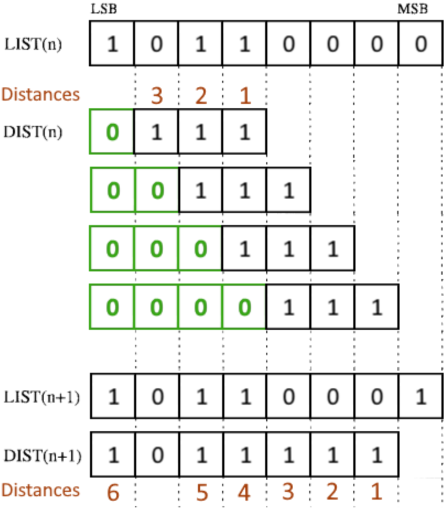
2ème algorithme : Shift Algorithm

Rajouter une graduation :



Parcours en profondeur




LSB
MSB
LIST(n)
1 0 1 1 0 0 0 0
Distances
3 2 1
DIST(n)
0 1 1 1
0 0 1 1 1
0 0 0 1 1 1
0 0 0 0 1 1 1
LIST(n+1)
1 0 1 1 0 0 0 1
DIST(n+1)
1 0 1 1 1 1 1
Distances
6 5 4 3 2 1

Comparaison des algorithmes I et II

- Calcul du tableau des distances
- Opérations sur Dist et Liste
- Algorithme I : Recherche sur toutes les règles possibles

Algorithme II : Recherche seulement sur les règles de Golomb d'ordre inférieur

Résultat : Algorithme I atteint les règles d'ordre 7
Algorithme II atteint les règles d'ordre 12

Améliorations de l'algorithme Shift

Elaguer l'arbre :

- en retirant les règles miroir
- en mettant des limites de tailles de règles plus précises

Règles presque optimales

Algorithme de Bose-Chowla :
Permet de trouver des bornes sup de recherche précises

Recherches actuelles

Règles de Golomb

- Règles de Golomb optimales (OGR-24) : Terminé le 13 octobre 2004 (après 1572 jours de calculs)
- Règles de Golomb optimales (OGR-25) : Terminé le 25 octobre 2008 (après 3006 jours de calculs)
- Règles de Golomb optimales (OGR-26) : Terminé le 24 février 2009 (après 24 jours de calculs)
- Règles de Golomb optimales (OGR-27) : Terminé le 19 février 2014 (après 1822 jours de calculs)

Historique des projets [modifier | modifier le code]

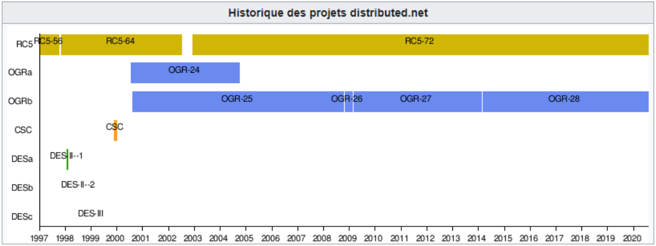
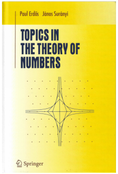


Figure 4 – https ://en.wikipedia.org/wiki/Distributed.net

Bibliographie

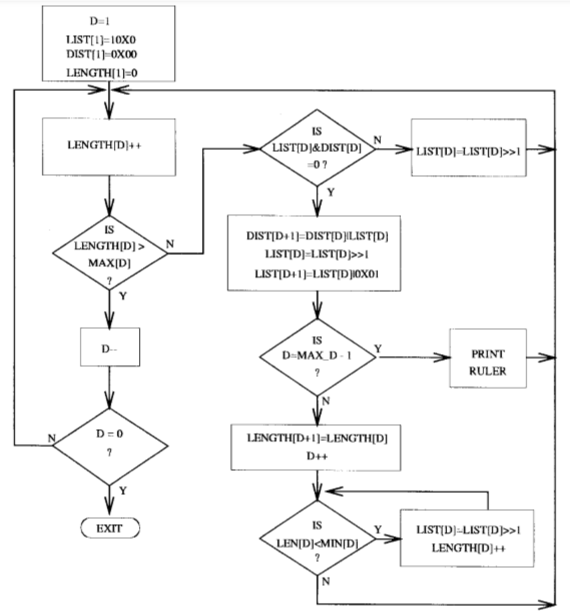
- Topics in the theory of Numbers, Paul Erdős, János Surányi ; Springer 1959



- Quelques problèmes de théorie des nombres par Paul Erdős, exercice 31 : https ://www.bsmath.hu/ p_erdos/1963-14.pdf

- Vidéo sur la démonstration de bose-chowla : https ://www.youtube.com/watch ?v=FxQru-3B0Es

- https ://datagenetics.com/blog/february22013/
- A review of the available contruction methods for Golomb Rulers Konstantin Drakakis
- A New Algorithm for Golomb Ruler Derivation and Proof of the 19 Mark Ruler Apostolos Dollas, Senior Member, IEEE, William T. Rankin, Member, IEEE, and David McCracken, Member, IEEE


D=1
LIST[1]=10X0
DIST[1]=0X00
LENGTH[1]=0
LENGTH[D]++
IS
LENGTH[D] >
MAX[D]
?
N
Y
D--
D = 0
?
N
Y
EXIT
IS
LIST[D]&DIST[D]
=0 ?
N
LIST[D]=LIST[D]>>1
Y
DIST[D+1]=DIST[D]|LIST[D]
LIST[D]=LIST[D]>>1
LIST[D+1]=LIST[D]|0X01
IS
D=MAX_D - 1
?
Y
PRINT
RULER
N
LENGTH[D+1]=LENGTH[D]
D++
IS
LEN[D]<MIN[D]
?
Y
LIST[D]=LIST[D]>>1
LENGTH[D]++
N

Règles presque optimales

Définition

Soit une famille de règles de Golomb de taille $n \in \mathbb{N}$. Soit $m(n)$ l'ordre associé. On appelle la famille asymptotiquement optimale ssi

$$\lim_{n \to \infty} \frac{\sqrt{n}}{m(n)} = 1$$

Majoration

Soit $s(n)$ l'ordre maximal d'une règle de taille n.

Théorème de P.Erdős et P.Turán

$$s(n) < \sqrt{n} + \sqrt[4]{n} + 1$$

Même inégalité retournée :

Théorème de P.Erdős et P.Turán

$$\left(\sqrt{s(n) - \frac{3}{4}} - \frac{1}{2}\right)^4 < n$$

Démonstration

Soit t un entier que l'on déterminera plus tard.
Considérons les $n+t$ intervalles contenant t éléments qui s'intersectent avec $[\![0, n]\!]$.

$$[\![-t+1, 0]\!], [\![-t+2, 1]\!], ..., [\![n, n+t-1]\!]$$

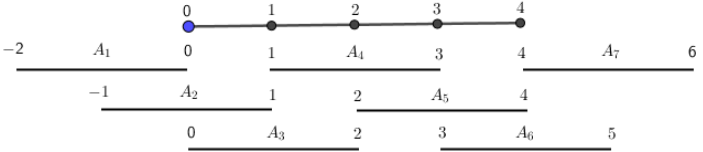


Figure 5 – Exemple pour $n = 4$, $t = 3$

Soit $A_1, ..., A_{n+t}$ le nombre d'éléments de l'ensemble de Sidon dans chacun de ces intervalles.

Démonstration

Soit $e_1, e_2, \dots, e_{s(n)}$ les éléments de l'ensemble de Sidon.
Soit D le nombre de fois qu'une paire (e_i, e_j), $i < j$ tombe dans la liste d'intervalles précédente.

$$D = \sum_{i=1}^{n+t} \binom{A_i}{2}$$

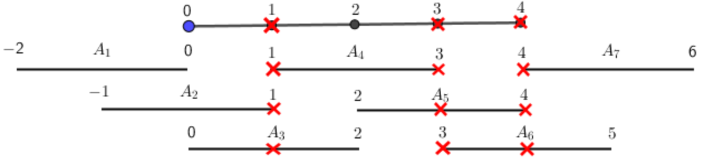


Figure 6 – Exemple pour $n = 4$, $t = 3$

Démonstration

Soit $d = e_j - e_i$.
Alors la paire (e_i, e_j) tombe dans $t - d$ intervalles.
Chaque d est différent. Ainsi,

$$D \leq \sum_{d=1}^{t-1} (t-d) = \frac{t(t-1)}{2}$$

On obtient alors

$$\sum_{i=1}^{n+t} \binom{A_i}{2} \leq \frac{t(t-1)}{2}$$

Démonstration

$$\sum_{i=1}^{n+t} \binom{A_i}{2} \leq \frac{t(t-1)}{2}$$

Démonstration

$$\sum_{i=1}^{n+t} \binom{A_i}{2} \leq \frac{t(t-1)}{2}$$

$$2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} = \sum_{i=1}^{n+t} A_i^2 - \sum_{i=1}^{n+t} A_i$$

Démonstration

$$\sum_{i=1}^{n+t} \binom{A_i}{2} \leq \frac{t(t-1)}{2}$$

$$2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} = \sum_{i=1}^{n+t} A_i^2 - \sum_{i=1}^{n+t} A_i$$

$$\sum_{i=1}^{n+t} A_i = ts$$

Démonstration

$$\sum_{i=1}^{n+t} \binom{A_i}{2} \leq \frac{t(t-1)}{2}$$

$$2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} = \sum_{i=1}^{n+t} A_i^2 - \sum_{i=1}^{n+t} A_i$$

$$\sum_{i=1}^{n+t} A_i = ts$$

Inégalité arithmético-quadratique

$$\sqrt{\frac{\sum_{i=1}^{n+t} A_i^2}{n+t}} \geq \frac{\sum_{i=1}^{n+t} A_i}{n+t}$$

Démonstration

$$\frac{t^2 s^2}{n+t} - ts \leq 2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} \leq t(t-1)$$

Démonstration

$$\frac{t^2 s^2}{n+t} - ts \leq 2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} \leq t(t-1)$$

$$s^2 - s\left(\frac{n}{t} + 1\right) - \left(\frac{n}{t} + 1\right)\left(t - 1\right) \leq 0$$

Démonstration

$$\frac{t^2 s^2}{n+t} - ts \leq 2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} \leq t(t-1)$$

$$s^2 - s\left(\frac{n}{t} + 1\right) - \left(\frac{n}{t} + 1\right)\left(t - 1\right) \leq 0$$

$$s \leq \frac{n}{2t} + \frac{1}{2} + \sqrt{n + t + \frac{n^2}{4t^2} - \frac{n}{2t} - \frac{3}{4}}$$

Démonstration

$$\frac{t^2 s^2}{n+t} - ts \leq 2 \cdot \sum_{i=1}^{n+t} \binom{A_i}{2} \leq t(t-1)$$

$$s^2 - s\left(\frac{n}{t} + 1\right) - \left(\frac{n}{t} + 1\right)\left(t - 1\right) \leq 0$$

$$s \leq \frac{n}{2t} + \frac{1}{2} + \sqrt{n + t + \frac{n^2}{4t^2} - \frac{n}{2t} - \frac{3}{4}}$$

Pour $t = \lfloor \sqrt[4]{n^3} \rfloor + 1$,

$$s < \sqrt{n} + \sqrt[4]{n} + 1$$

$\square$

2ème majoration et question

2ème majoration par J. Cilleruelo

$$s(n) < \sqrt{n} + \sqrt[4]{n} + \frac{1}{2}$$

2ème majoration et question

2ème majoration par J. Cilleruelo

$$s(n) < \sqrt{n} + \sqrt[4]{n} + \frac{1}{2}$$

Question de P.Erdős

Est-ce que

$$s(n) = \sqrt{n} + O(1)\,?$$

(P.Erdős offrait 500 $ pour la réponse)

Algorithme I :

```python
def dessinregle(x):
    regle = ""
    while x >= 1:
        if x % 2 == 1:
            regle += "|"
        else:
            regle += "."
        x = x//2
    return regle

def liste(x):
    l = []
    mem = 0
    while x >= 1:
        if x % 2 == 1:
            l.append(mem)
        mem += 1
        x = x//2
    return l

def dist(x):
    l = liste(x)
    d = []
    for i in range(len(l)):
        for j in range(i+1,len(l)):
            if l[j] - l[i] in d:
                return False
            d.append(l[j] - l[i])
    d.sort()
    return d


def golomb(ordre):
    maxi = 2**ordre
    found = False
    x = 1
    l = [0]
    regle = []
    while not found:
        l = liste(x)
        if len(l) == ordre:
            d = dist(x)
            if d != False:
                found = True
```

```python
                regle.append(x)
        x += 2
    n = l[-1]

    while l[-1] <= n :
        if len(l) == ordre:
            d = dist(x)
            if d != False:
                regle.append(x)
        x += 2
        l = liste(x)
    for i in range(len(regle)):
        print(dessinregle(regle[i]))
        print(liste(regle[i]))
        print()
```

Algorithme II :

```ocaml
(* Pour représenter les champs de bits : BIGINT *)
type bitfield = int list
let nbbits = 60 (* Nombre de bits utilisables *)
let vmax = 1 lsl nbbits
let mask = vmax - 1

let zero = [0]
let un = [1]
let rec inc = function (* Ajoute 1 *)
  | [] -> [1]
  | t::q when t+1 < vmax -> t+1::q
  | t::q -> 0 :: inc q
let rec lshift = function (* Multiplie par 2 *)
  | [] -> []
  | t::q when 2*t < vmax -> 2*t :: lshift q
  | t::q -> (2*t) land mask :: inc (lshift q)
let rec or_if_disj a b = (* retourne Some (a||b) si a&&b=0 et None sinon *)
  match a, b with
    | (l, []) | ([], l) -> Some l
    | (t1::q1, t2::q2)
       -> match or_if_disj q1 q2 with
            | None -> None
            | Some x -> if t1 land t2 = 0
                        then Some (t1 lor t2::x)
                        else None

let rec print = function
```

```
  | [] -> ()
  | [0] -> ()
  | [x] -> print_string (if x mod 2 = 1 then "|" else ".");
          print [x/2]
  | t::q -> let x = ref t in
             for i = 0 to nbbits - 1 do
               print_string (if !x mod 2 = 1 then "|" else ".");
               x := !x / 2
             done;
             print q;;

type state = {
  nbmarks: int; (* Nombre de marques *)
  lastmark: int; (* Index de la marque la plus grande *)
  list: bitfield; (* Champ de bit des marques 01...n *)
  dist: bitfield; (* Champ de bit des distances n...21 *)
}

let rec next s maxi =
  if s.lastmark >= maxi then [] else
    let cand = next { nbmarks = s.nbmarks;
                      list = lshift s.list;
                      dist = s.dist;
                      lastmark = s.lastmark+1 } maxi
    in match or_if_disj s.dist s.list with
         | None -> cand
         | Some l -> { nbmarks = s.nbmarks+1;
                       dist = l;
                       list = inc (lshift s.list);
                       lastmark = s.lastmark+1 }::cand

let select { nbmarks=nbmarks;
             lastmark=lastmark;
             dist=dist; list=list }
  = true

let golomb nbmarks maxi =
  let rec expl s =
    if s.nbmarks = nbmarks then
      (print s.list; print_newline ())
    else List.iter expl (List.filter select (next s maxi))
  in expl {list=un; dist=zero; nbmarks=1; lastmark=0};;

golomb 5 11;;
golomb 6 17;;
golomb 7 25;;
```

```
golomb 8 34;;
golomb 9 44;;
golomb 10 55;;
golomb 11 72;;
golomb 12 85;;
golomb 13 106;;
golomb 14 127;;
golomb 15 151;;
golomb 16 177;;
```