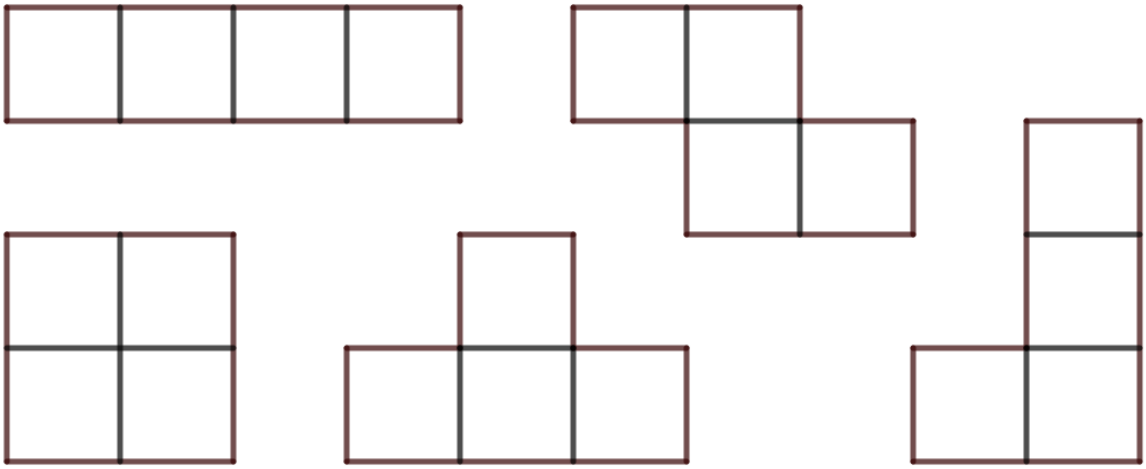
DES DÉFIS POUR NOS ÉLÈVES

LES CINQ TÉTRAMINOS

(d'après Martin Gardner)

On pourrait dans un premier temps demander aux élèves de trouver tous les tétraminos (formés par quatre carrés identiques joints par un côté). Il y en a bien cinq.

Les voici :



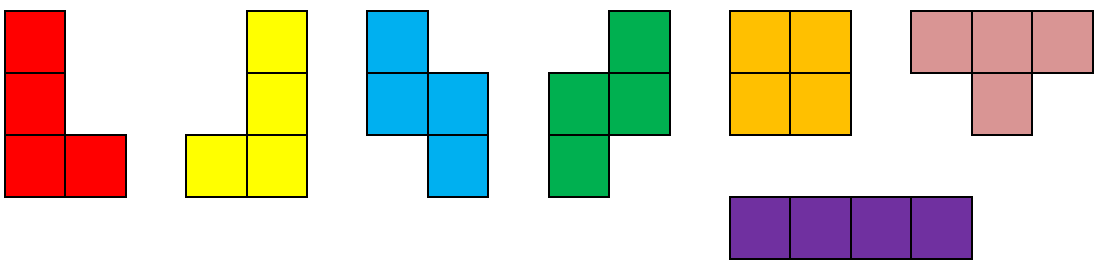
Est-il possible de les assembler pour recouvrir un rectangle 4 × 5 ?

Si oui, faire une figure et à démontrer si non.

NB : Les pièces sont retournables.

Prolongement 1 : On propose d'utiliser deux pièces de chaque sorte de tétramino pour un recouvrir un rectangle deux fois plus vaste, c'est-à-dire ayant une aire de 40 carreaux (plusieurs rectangles ont cette aire).

Prolongement 2 : Avec les pièces du jeu TETRIS (**pièces non retournables**), est-il possible de recouvrir un rectangle 7x4 ? Et avec deux exemplaires de chaque pièce, est-il possible de recouvrir un rectangle 7x8 ?



Retour au sommaire

DES DÉFIS POUR NOS ÉLÈVES

QUEL EST CE NOMBRE ?

(cycle 3)

Mon nombre de centaines est le double du nombre formé par mes deux derniers chiffres.
Mon chiffre des unités est la moitié de mon chiffre des dizaines.
La somme de tous mes chiffres est un nombre inférieur à 10.

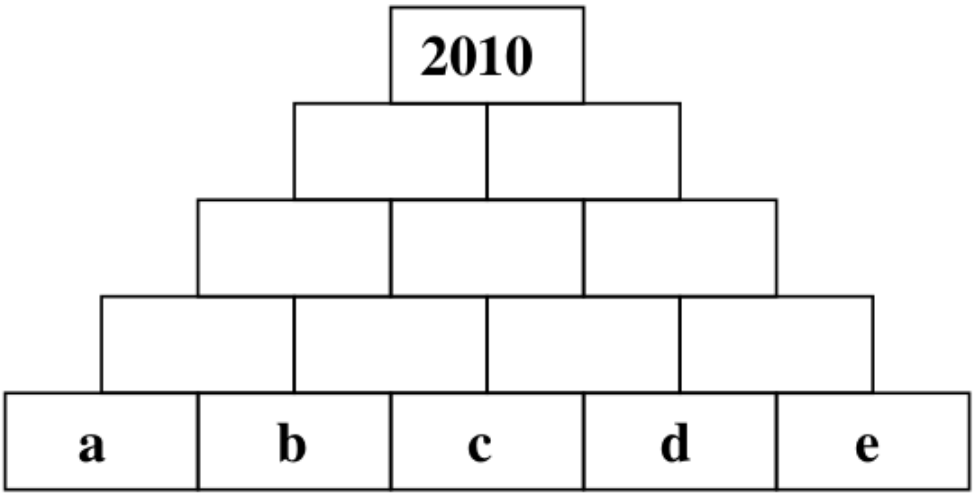
Qui suis-je ?

DÉFI ALGORITHMIQUE N° 146

Certaines énigmes du rallye mathématique de Lorraine auraient certainement été plus simples à résoudre à l'aide d'un petit programme informatique.

Nous vous proposons ici, comme défi, de résoudre l'exercice ci-dessous à l'aide d'un programme.

L'exercice suivant avait été proposé en 2010.



Dans ce mur, le nombre inscrit sur chaque brique est la somme des deux nombres inscrits sur les briques sur lesquelles elle repose.

Les nombres a, b, c, d et e inscrits sur les briques de la base sont des entiers non nuls tous différents et le commissaire Albert Girard a même réussi à déterminer que le nombre a est le plus grand nombre possible tel que la brique du sommet soit égale à 2010.

Aidez-le à retrouver les valeurs des nombres a, b, c, d et e.

On demande d'écrire une fonction mur(N) qui, pour un entier N au sommet du mur ci-dessus, renvoie les valeurs a, b, c, d et e, avec a le plus grand possible.

Retour au sommaire

SOLUTIONS DÉFIS N°145

1 - DATES PYTHAGORICIENNES

Le but est de déterminer les dates au format jj/mm/aa telles que $jj^2 + mm^2 = aa^2$, comme par exemple le 16 décembre 2020 : 16²+12²=20². Notons que c'est l'année qui joue le rôle de l'hypoténuse.

Petit rappel théorique

Les acteurs intervenant dans ce qui suit sont évidemment des entiers positifs.

- Un triplet de nombres entiers positifs $(u\,;\,v\,;\,w)$ est *pythagoricien* si et seulement si $u^2 + v^2 = w^2$.
- Un triplet $(u\,;\,v\,;\,w)$ est *pythagoricien primitif* si et seulement si $u^2 + v^2 = w^2$ et $\{u\,;\,v\,;\,w\}$ sont premiers entre eux dans leur ensemble.
- Dans un triplet pythagoricien, u et v ne peuvent être tous les deux impairs. En effet, dans ce cas $u^2 + v^2$ serait congru à 2 modulo 4 alors que w^2 ne peut être congru qu'à 0 ou 1 modulo 4 suivant la parité de w.
- Dans un triplet pythagoricien primitif, u et v ne peuvent être tous les deux pairs car alors w le serait aussi, ce qui nuirait gravement à l'irréductibilité de $(u\,;\,v\,;\,w)$. Pour un triplet primitif, u et v sont donc de parités différentes, et w impair.
- Un triplet pythagoricien est de la forme $(ku\,;kv\,;\,kw)$ où $(u\,;\,v\,;\,w)$ est un triplet pythagoricien primitif et $k \in \mathbb{N}^*$. L'entier k est simplement le PGCD du triplet d'origine. Il suffit donc de connaître les primitifs pour en déduire les autres par proportionnalité.
- Les triplets pythagoriciens primitifs $s\,(u\,;\,v\,;\,w)$ avec u impair et v pair sont de la forme :

$$\begin{cases} u = b^2 - a^2 \\ v = 2ab \\ w = b^2 + a^2 \end{cases} \text{ avec } 0 < a < b \text{ ; } a+b \text{ impair et } a \wedge b = 1.$$

a) Il est clair qu'un tel triplet est pythagoricien. Remarquons aussi que $u = b^2 - a^2$ est impair puisque $a + b$ et $a - b$ le sont, et que $v = 2ab$ est pair.

b) Si un nombre premier p divise les trois éléments d'un tel triplet, alors $p \neq 2$ et il divise $w + v$ et $w - v$;
Il divise donc $a - b$ et $a + b$, par conséquent $2a$ et $2b$.
Étant premier avec 2, p divise donc a et b… d'où $p = 1$.

c) Un tel triplet est donc bien un triplet primitif avec u impair et v pair.
Considérons maintenant un triplet primitif $(u\,;\,v\,;\,w)$ avec u impair et v pair.
Nous avons donc $v = 2t$ et par conséquent $4t^2 = w^2 - u^2 = (w-u)(w+u)$.
$w - u$ et $w + u$ étant tous les deux pairs, posons $w - u = 2r$ et $w + u = 2s$.
Il en découle que $u = s - r$; $w = s + r$ et $t^2 = rs$.
r et s sont premiers entre eux car un diviseur commun le serait aussi de u, w, t et donc de v.
Qui plus est $r < s$ et $r + s$ étant impair, r et s sont de parités différentes.
Si un nombre premier p divise r, alors il divise t et donc le facteur p apparaît avec une puissance paire dans $t^2 = rs$. Ne pouvant figurer dans la décomposition de s, il est donc élevé à une puissance paire dans celle de r. Cela prouve que r est un carré parfait et par symétrie s aussi.
Nous avons donc $r = a^2$, $s = b^2$ avec $0 < a < b$, a et b de parités différentes et $a \wedge b = 1$.
En reportant la chose, $u = b^2 - a^2$, $v = 2ab$ et $w = b^2 + a^2$. Ce qui achève la démonstration.

Retour au sommaire

- Les triplets pythagoriciens primitifs $(u\,;\,v\,;\,w)$ avec u pair et v impair sont évidemment obtenus par échange de u et v dans le cas précédent.

Revenons à nos moutons

On va donc chercher les triplets primitifs pour lesquels la première coordonnée n'excède pas 31 et la deuxième 12. De ceux-là on déduira par proportionnalité les triplets non primitifs qui conviennent.

Premier cas :
Les triplets $(b^2-a^2\,;2ab\,;\,b^2+a^2)$ avec $0<a<b,\ a\ +\ b$ impair, $a\wedge b=1,\ ab\leq 6,\ b^2-a^2\leq 31.$

a	b	b^2-a^2	$2ab$	primitif	déduits
1	2	3	4	(3 ; 4 ; 5)	(6 ; 8 ; 10) (9 ; 12 ; 15)
1	4	15	8	(15 ; 8 ; 17)	
2	3	5	12	(5 ; 12 ; 13)	

Les triplets $(2ab\,;b^2-a^2\,;\,b^2+a^2)$ avec $0<a<b$, $a\ +\ b$ impair et $a\wedge b=1,\ 2ab\leq 31,\ b^2-a^2\leq 12.$

a	b	$2ab$	b^2-a^2	primitif	déduits
1	2	4	3	(4 ; 3 ; 5)	(8 ; 6 ; 10) (12 ; 9 ; 15) (16 ; 12 ; 20)
2	3	12	5	(12 ; 5 ; 13)	(24 ; 10 ; 26)
3	4	24	7	(24 ; 7 ; 25)	

Les 12 dates répondant au problème posé sont donc :

04/03/05 ; 03/04/05 ; 08/06/10 ; 06/08/10 ; 12/05/13 ; 05/12/13 ;
12/09/15 ; 09/12/15 ; 15/08/17 ; 16/12/20 ; 24/07/25 ; 24/10/26.

2 – PUZZLE DE GRENOBLE

Nous vous renvoyons à l'article « LE PUZZLE DE GRENOBLE » dans notre rubrique MATHS et JEUX.

Retour au sommaire

SOLUTION ALGO-RALLYE 145

Le défi algorithmique du PV 145 reprenait la question subsidiaire du Rallye 2009 concernant le jeu du loup et de l'agneau dans un quadrillage. On demandait de produire des échantillons de plusieurs parties pour estimer la proportion des parties gagnées par le loup et celles gagnées par l'agneau. Il était alors possible de connaître les espérances de gains de chacun des deux animaux.

La fonction **échantillonnage(N)** renvoie les proportions de séries de 100 parties où le gain du loup est supérieur à celui de l'agneau et celles où c'est celui de l'agneau qui est supérieur.

Pour simuler une partie (dans la fonction **partie(n)**), on considère un tableau de 3 tableaux de 3 entiers : 0 pour une case vide, 1 pour une case occupée par le loup et 2 pour une case occupée par l'agneau.

Chacun lance un dé (variable **de**) et leur fonction de déplacement (procédures **déplacement_loup(res)** et **déplacement_agneau(res)** selon le résultat **res** du dé) respective modifie la position des deux acteurs selon les règles du jeu. Après chaque déplacement, on vérifie si une des situations de fin de partie (fonction **fin()**) est rencontrée.

La fonction **position(acteur)** renvoie les coordonnées de l'acteur dans le tableau. Si l'acteur a disparu du plateau, alors le loup a gagné. On effectue quand même la procédure de déplacement.

Le programme peut paraître relativement complexe. Toutefois, certains raccourcis de programmation n'ont pas été employés pour préserver une certaines lisibilité. Il faut enfin noter que les lignes d'un tableau de tableaux correspondent aux ordonnées dans le quadrillage avec l'origine du repère en haut à gauche du quadrillage.

Pseudo-code :

```
Fonction partie(n : entier ; agneau, loup : entiers)
    plateau ← [[0,0,1],[0,0,0],[2,0,0]] ;      0 : case vide, 1 : loup, 2 : agneau
    agneau ← 0 ;                gain de l'agneau
    loup ← 0 ;            gain du loup
    pour i allant de 1 à n, faire :
        tant que fin()=0, faire :    fin() renvoie 0 si la partie n'est pas terminée
            dé ← entier_aléatoire(1,6) ;
            déplacement_agneau(dé) ;
            dé ← entier_aléatoire(1,6) ;
            déplacement_loup(dé) ;
        finTantque ;
        si fin()=1, alors :    fin() renvoie 1 si le loup gagne et 2 si c'est l'agneau
            loup ← loup + 2 ;
        sinon :
            agneau ← agneau + 1 ;
        finSi ;
        plateau ← [[0,0,1],[0,0,0],[2,0,0]] ;      réinitialisation du plateau
    finPour ;
    renvoyer agneau,loup.Fonction fin(;acteur : entier)    vérifie la fin de la partie
    xl,yl ← position(1) ;        coordonnées du loup
    xa,ya ← position(2) ;            coordonnées de l'agneau
    acteur ← 0 ;                 personnage gagnant
    si xl = -1 ou xa = -1, alors :      position renvoie (- 1 ; - 1) si l'un des personnages a
        acteur ← 1 ;                        disparu : le loup a alors gagné
    sinon :
        si xa > xl ou ya < yl, alors :      l'agneau gagne s'il dépasse une position
            acteur ← 2 ;                        possible pour le loup
        finSi ;
    finSi ;
    renvoyer acteur
```

Retour au sommaire

```
Fonction position(acteur : entier ; x,y : entiers)        renvoie les coordonnées de l'acteur
        x ← -1 ;        position renvoie (- 1 ; - 1) si l'acteur n'est plus sur le plateau
        y ← -1 ;
        pour i allant de 1 à 3, faire :
                pour j allant de 1 à 3, faire :
                        si plateau[i][j] = acteur, alors :
                                x ← j ;
                                y ← i ;
                        finSi ;
                finPour ;
        finPour ;
        renvoyer x,y

Procédure déplacement_agneau(res : entier;)
        x,y ← position(2) ;
        plateau[y][x] ← 0 ;          on remplace la position de départ par une case vide
        si res est pair, alors :
                si x<2, alors :      déplacement vers la droite en restant dans le tableau
                        x ← x+1 ;
                finSi ;
        sinon :
                si y>0, alors :      déplacement vers le haut en restant dans le tableau
                        y ← y-1 ;
                finSi ;
        finSi ;
        plateau[y][x] ← 2 ;          on remplace la position d'arrivée par 2

Procédure déplacement_loup(res : entier;)
        x,y ← position(2) ;
        plateau[y][x] ← 0 ;          on remplace la position de départ par une case vide
        si res est pair, alors :
                si x>0, alors :      déplacement vers la gauche en restant dans le tableau
                        x ← x-1 ;
                finSi ;
        sinon :
                si y<2, alors :      déplacement vers le bas en restant dans le tableau
                        y ← y+1 ;
                finSi ;
        finSi ;
        plateau[y][x] ← 2 ;          on remplace la position d'arrivée par 1Fonction
échantillonnage(N : entier ; agneau, loup : flottants)
        loup ← 0 ;
        agneau ← 0 ;
        pour i allant de 1 à N, faire :
                résultat ←  partie(100) ;
                si résultat[0] > résultat[1], alors :
                        agneau ← agneau + 1/N ;
                sinon :
                        loup ← loup + 1/N ;
                finSi ;
        finPour ;
        renvoyer agneau, loup
```

Retour au sommaire

Python

```
from random import randint

# Déplacement des acteurs

def deplacement_loup(res):
    global plateau
    x,y=position(1)
    plateau[y][x]=0
    if res%2==0:
        if x>0:     # déplacement vers la gauche
            x=x-1
    else:
        if y<2:     #déplacement vers le bas
            y=y+1
    plateau[y][x]=1

def deplacement_agneau(res):
    global plateau
    x,y=position(2)
    plateau[y][x]=0
    if res%2==0:
        if x<2:     # déplacement vers la droite
            x=x+1
    else:
        if y>0:     #déplacement vers le haut
            y=y-1
    plateau[y][x]=2

# Détermination de la fin de la partie et du numéro du gagnant

def fin():
    xl,yl=position(1)
    xa,ya=position(2)
    acteur=0
    if xl==-1 or xa==-1:        # le loup gagne s'il atteint
        acteur=1                # la position de l'agneau
    else :
        if xa>xl or ya<yl:      # l'agneau gagne s'il dépasse
            acteur=2            # la position du loup
    return acteur

# Détermination de la position de l'acteur sur le plateau

def position(acteur):
    global plateau
    x,y=-1,-1
```

Retour au sommaire

```
    for i in range(3):
        for j in range(3):
            if plateau[i][j]==acteur:
                x,y=j,i
    return x,y        # si l'acteur n'est plus sur la grille, # c'est que son adversaire a pris sa place

# Répétition de n parties

def partie(n):
    global plateau
    plateau=[[0,0,1],
             [0,0,0],
             [2,0,0]] # 1 : loup, 2 : agneau
    loup,agneau=0,0 # nombre de parties gagnées par chaque acteur
    for i in range(n):
        while fin()==0:
            de=randint(1,6)
            deplacement_agneau(de) ## test de position !!
            de=randint(1,6)
            deplacement_loup(de)
        if fin()==1:
            loup=loup+2
        else:
            agneau=agneau+1
        plateau=[[0,0,1],
                 [0,0,0],
                 [2,0,0]]
    return agneau,loup

# Création des échantillons et comparaison

def echantillonnage(N):
    loup,agneau=0,0
    for i in range(N):
        resultat=partie(100)
        if resultat[0]>resultat[1]:
            agneau+=1/N
        else:
            loup+=1/N
    return agneau,loup
```

Retour au sommaire

DÉFI POUR NOS COLLÈGUES

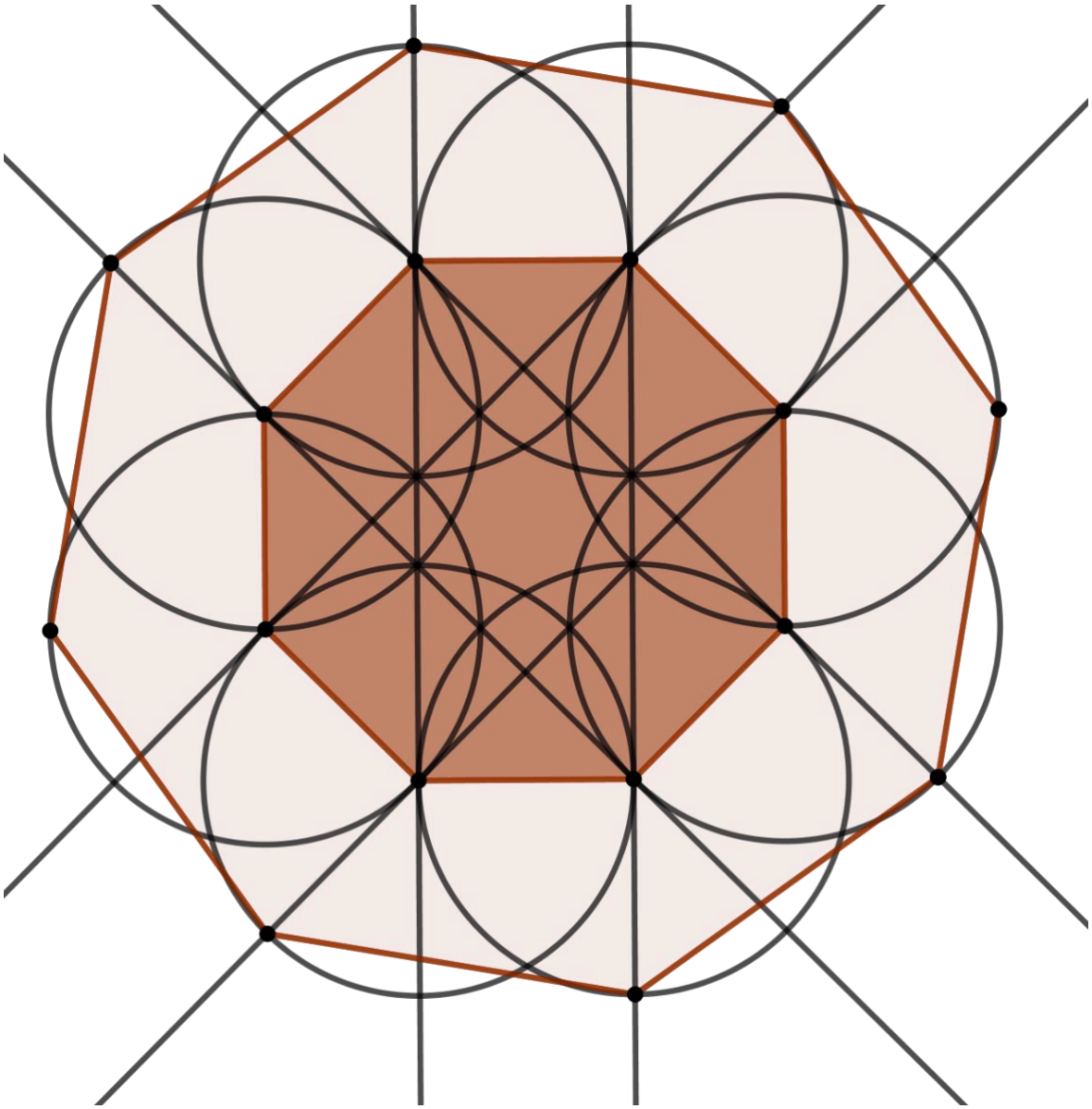
OCTOGONES

Proposé par Fathi DRISSI

En cherchant une trisection de l'octogone régulier pour un futur article de la rubrique « Maths et découpages », j'ai trouvé une construction dont la justification pourrait être donnée comme défi aux collègues dans le prochain numéro du PV.

Sur la figure ci-dessous, le petit octogone est un octogone régulier.

Démontrer que le grand octogone est aussi un octogone régulier et son aire vaut trois fois celle du petit octogone.



Retour au sommaire