Mathématiques Discrètes

Devoir surveillé n° 2— le 18 décembre 2018

Prenez le temps de lire ce sujet. Ce devoir comporte 4 exercices. Les exercices sont indépendants. Le barème est sur 22 points. Toute réponse doit être **justifiée**. Calculatrice et documents interdits.

Exercice 1 [3 points]

`''.join(['a','bc', 'd'])` et `''.join(['ab', 'cd'])` sont deux expressions Python produisant la même chaîne `'abcd'`.

Q 1.1 [3 points] Étant donnée une chaîne `s` de longueur n, exprimez en fonction de n le nombre de listes `l`, ne contenant que des chaînes de caractères non vides, telles que `''.join(l) == s`. *Éléments de réponse*

- première version. Je vais compter les listes `l` par longueur croissante.
  - la liste vide ne peut convenir que lorsque la chaine `s` est la chaîne vide.
  - sinon la liste n'est pas vide. dans ce cas la longueur de la liste est inférieure ou égale à la longueur de la chaîne car aucun des morceaux qui constituent cette liste ne peut être la chaine vide. on a donc $1 \leq len(l) \leq n$
    d'autre part si je note x_1, x_2, ... x_k la longueur des chaînes $l[0]$, $l[1]$ jusqu'à $l[k-1]$, on doit avoir $x_1 + x_2 + ...x_k = n$ et $x_1 \geq 1$, $x_2 \geq 1$, ..., $x_k \geq 1$ reciproquement chaque solution correspond à un découpage.
    compter le nombre de listes l de longueur k répondant au problème posé est le nombre de sol de $y_1 + y_2 + ...y_k = n - k$, cela revient à compter les mots de $k-1$ barres $n-k$ points. ces mots sont de longueur longueur $n-1$, on choisit $k-1$ barres. de tels mots, il y en a $\binom{n-1}{k-1}$;
    k peut varier de 1 à n donc $k-1$ varie de 0 à $n-1$. en posant $j = k-1$, on retrouve la formule du binôme donc 2^{n-1} découpage
- autre version

$$u_{n+1} = 2u_n$$

$$u_1 = 1$$

  À chaque découpage d'une chaine de taille n, on peut associer deux nouveaux découpages d'une chaine de taille $n+1$, le premier en ajoutant le dernier caractère à la fin de la derniere chaine, le second en ajoutant à la liste une chaine de longueur 1 formé de ce dernier caractère.
- autre version le premier caractère, je commence une chaîne, puis à chaque caractère suivant, je dois me demander si je continue la chaine courante ou si je commence une nouvelle chaine, Il y a n-1 choix binaires à faire.
- autre version À chaque découpage, je peux associer l'ensemble formés par tout les indices des caractères situés en fin des chaînes dans le découpage, privé de $n-1$. Je construis ainsi une application de l'ensemble des découpages, dans l'ensemble des parties de l'intervalle entier de 0 à $n-2$ bornes comprises. Elle est bijective, à chaque découpage, on associe une partie, et à chaque partie, on est capable de retrouver le découpage. il y a $n-1$ éléments dans l'intervalle, donc 2^{n-1} éléments dans l'ensemble des parties

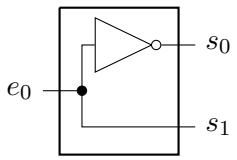
Exercice 2 [5 points]

Un décodeur $(n, 2^n)$ est un circuit combinatoire, ayant n entrées et 2^n sorties. Comme les entrées e_0, e_1, ... ,e_{n-1} prennent leurs valeurs dans l'ensemble $\{0, 1\}$, exactement 2^n valeurs peuvent être prises par les entrées. Chacune peut être représentée par un entier e codé sur n bits.

$$e = \sum_{i=0}^{n-1} e_i 2^i$$

Une seule des sorties s_0, s_1, ... , s_{2^n-1} du décodeur sera à 1, celle qui correspond au nombre e. Toutes les autres sorties sont à 0. Voici la table de vérité du décodeur $(1,2)$:

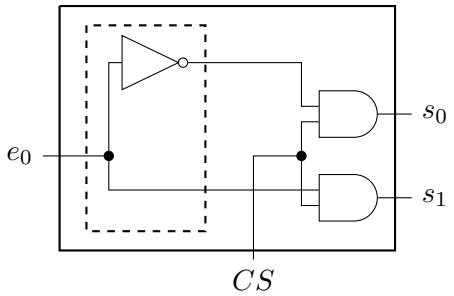
e_0	s_1	s_0
0	0	1
1	1	0



Pour des raisons pratiques, on ajoute une entrée supplémentaire au circuit nommée CS (chip select), cette entrée agit comme un interrupteur marche/arret du circuit. C'est-à-dire, lorsqu'elle est à 0, toutes les sorties du circuit sont mises à 0 et lorsqu'elle est à 1 le circuit fonctionne normalement. Afin de bien préciser voici la table de vérité du circuit :

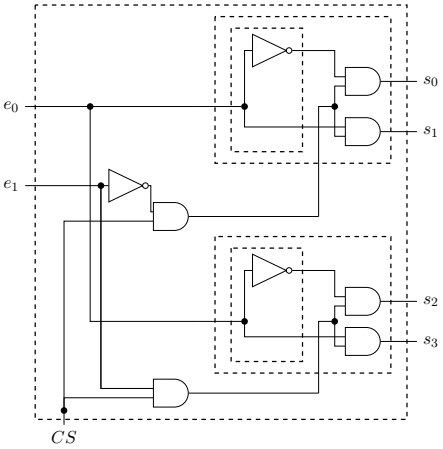
et une réalisation :

CS	e_0	s_1	s_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

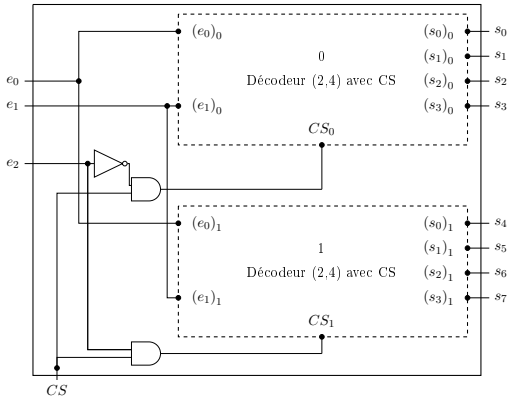


En utilisant deux décodeurs $(1,2)$ avec CS. On réalise le circuit suivant qui correspond à un décodeur $(2,4)$ avec CS :

CS	e_1	e_0	s_3	s_2	s_1	s_0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



En utilisant deux décodeurs $(2,4)$ avec CS. On peut réaliser un circuit suivant qui correspond à un décodeur $(3,8)$ avec CS.



De manière générale, en utilisant deux décodeurs $(n, 2^n)$ avec CS, on est capable de construire un décodeur $(n+1, 2^{n+1})$ avec CS.

On note u_n le nombre de portes logiques (et, non, ou) utilisée pour réaliser un décodeur $(n, 2^n)$ avec CS.

Q 2.1 [1 point] Donnez les valeurs de u_1, u_2, u_3.

Éléments de réponse

$$u_1 = 3$$
$$u_2 = 3 + 2u_1 = 9$$
$$u_3 = 2u_2 + 3 = 21$$

Q 2.2 [1 point] Donnez la relation de récurrence satisfaite par les termes de cette suite pour $n \geq 1$. Précisez la nature de la suite.

Éléments de réponse

$$\forall n \geq 1 u_{n+1} = 2u_n + 3$$

Il s'agit d'une suite arithmético-géométrique

Q 2.3 [1 point] Quel sens pourrait-on donner à un décodeur $(0, 1)$ avec CS ? Si cela avait un sens, combien de portes logiques cela nécessiterait ? *Éléments de réponse*

oui, pas d'autre entrée que CS et une seule sortie s_0 qui vaut 1 lorsque à la fois toutes les entrées autres que CS sont à 0 et CS vaut 1. Dans tous les autres cas s_0 vaut 0. On peut réaliser cela en plaçant un fil entre CS et la sortie s_0. Il ne faut donc aucune porte logique.

Q 2.4 [2 points] Donnez une expression de u_n directement en fonction de n, c'est-à-dire ne faisant plus intervenir de termes précédents de la suite. *Éléments de réponse*

Comme il s'agit d'une suite arithético-géométrique, on cherche une solution constante, qui satisfait l'équation de récurrence.

$$l = 2l + 3$$
$$l = -3$$

on pose ensuite $v_n = u_n - l$ v_n satisfait l'équation

$$v_{n+1} = u_{n+1} - l = 2u_n + 3 - l = 2u_n + 6 = 2(u_n + 3) = 2(u_n - l) = 2v_n$$

donc (v_n) est une suite géométrique de raison 2. $v_n = 2^n v_0$ et $v_0 = u_0 - l = 0 + 3$ $v_n = 3 \times 2^n$ et $u_n = v_n + l = 3 \times 2^n - 3$ On retrouve bien $u_0 = 0$, $u_1 = 3$, $u_2 = 9$, $u_3 = 21$

Exercice 3 [9 points]

On donne le code suivant :

```
def C(n,m):
    assert type(n)==int,n>=0
    assert type(m)==int,m>=0
    if n==0 or m==0:
       return 1
    else:
       return C(n-1,m)+C(n,m-1)+C(n-1,m-1)
```

Q 3.1 [1 point] Calculez $C(i, j)$ pour i et j inférieur ou égal à 3.

Éléments de réponse

```
C(0,0)= 1  C(0,1)= 1  C(0,2)= 1  C(0,3)=1
C(1,0)= 1  C(1,1)= 3  C(1,2)= 5  C(1,3)=7
C(2,0)= 1  C(2,1)= 5  C(2,2)=13  C(2,3)=25
C(3,0)= 1  C(3,1)= 7  C(3,2)=25  C(3,3)=63
```

Q 3.2 [1 point] Démontrez que pour tout couple d'entier $C(n,m) = C(m,n)$. *Éléments de réponse*

Je fais une preuve par récurrence.

$$\mathcal{H}_k \forall (n,m) \in (\mathbb{N} \times \mathbb{N}) n+m \leq k \Longrightarrow C(n,m) = C(m,n)$$

- Initialisation : $\mathcal{H}_0$ est vraie car C(0,0)=C(0,0)=1
- Hérédité : supposons $\mathcal{H}_k$ vraie, et prouvons $\mathcal{H}_{k+1}$ soit n et m deux entiers naturels tels que $n+m \leq k+1$
  - si n=0 alors c'est fini car $C(0,m) = 1$ et $C(m,0) = 1$
  - si m=0 alors c'est fini car $C(n,0) = 1$ et $C(0,n) = 1$
  - si n>0 et m>0 alors
    - d'une part $C(n,m) = C(n-1,m) + C(n-1,m-1) + C(n,m-1)$
    - d'autre part $C(m,n) = C(m-1,n) + C(m-1,n-1) + C(m,n-1)$

    Comme $n+m \leq k+1$ on a $n-1+m \leq k$ et $n-1+m-1 \leq k-1 \leq k$. Pour chacun des termes, on peut appliquer l'hypothèse de récurrence donc $C(m-1,n) = C(n,m-1)$ et $C(n-1,m-1) = C(m-1,n-1)$ et $C(n-1,m) = C(m,n-1)$. Par conséquent, $C(n,m) = C(m,n)$.

On considère les mots sur l'alphabet $\mathcal{A} = \{U, R, D\}$.

On définit pour tout mot m de $\mathcal{A}^*$ respectivement les nombres $|m|_U$,$|m|_R$, et $|m|_D$ comme les nombres d'occurrences respectifs des lettres U, R et D.

Une relation $\mathcal{R}$ sur les mots est définie par $m_1 \mathcal{R} m_2$ si par définition

$$|m_1|_U + |m_1|_D = |m_2|_U + |m_2|_D \text{ et } |m_1|_R + |m_1|_D = |m_2|_R + |m_2|_D.$$

Q 3.3 [1 point] Montrez que $\mathcal{R}$ est une relation d'équivalence. *Éléments de réponse*

- $\mathcal{R}$ est réflexive. Car pour tout mot m
  - la somme du nombre d'occurrences de U et du nombre d'occurences de D est égale à elle même
  - et la somme du nombre d'occurrences de R et du nombre d'occurences de D est égale à elle même.
- $\mathcal{R}$ est symétrique. Car pour tout couple de mots (m_1, m_2), si $m_1 \mathcal{R} m_2$ alors

  $$|m_1|_U + |m_1|_D = |m_2|_U + |m_2|_D \text{ et } |m_1|_R + |m_1|_D = |m_2|_R + |m_2|_D$$

  par conséquent

  $$|m_2|_U + |m_2|_D = |m_1|_U + |m_1|_D \text{ et } |m_2|_R + |m_2|_D = |m_1|_R + |m_1|_D$$

  * et enfin $m_2 \mathcal{R} m_1$
- $\mathcal{R}$ est transitive. Car pour tout triplet de mots (m_1, m_2, m_3), si $m_1 \mathcal{R} m_2$ et si $m_2 \mathcal{R} m_3$ alors

  $$|m_1|_U + |m_1|_D = |m_2|_U + |m_2|_D \text{ et } |m_1|_R + |m_1|_D = |m_2|_R + |m_2|_D$$

  et

  $$|m_2|_U + |m_2|_D = |m_3|_U + |m_3|_D \text{ et } |m_2|_R + |m_2|_D = |m_3|_R + |m_3|_D$$

  donc

  $$|m_1|_U + |m_1|_D = |m_3|_U + |m_3|_D \text{ et } |m_1|_R + |m_1|_D = |m_3|_R + |m_3|_D$$

  et enfin $m_1 \mathcal{R} m_3$

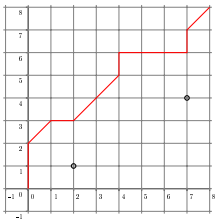
On interprète chaque mot de $\mathcal{A}^*$ comme un chemin tracé depuis l'origine $(0,0)$ en ne faisant que des pas vers le haut (U) des pas vers la droite (R), et des pas en diagonale (D).

$$U : (x,y) \mapsto (x,y+1)$$

$$R : (x,y) \mapsto (x+1,y)$$

$$D : (x,y) \mapsto (x+1,y+1).$$

Q 3.4 [1 point] Dessinez le chemin $UUDRDDURRRUD$. *Éléments de réponse*



$UUDRDDURRRUD$

Q 3.5 [1 point] Que dire de deux mots dont l'interprétation mène au même point ? Donnez la signification de la relation $\mathcal{R}$. *Éléments de réponse*

Deux mots dont l'interpretation mène au même point sont évidemment en relation, en effet l'ordonnée finale est égale à la somme du nombre de U et du nombre de D qui composent le mot. L'abscisse finale est égale à la somme du nombre de R et du nombre de D. Réciproquement, si deux mots sont en relation alors ils ont même interprétation. La relation $\mathcal{R}$ sert à caractèriser les mots de même interpretation.

Q 3.6 [1 point] Donnez la classe d'équivalence de UR, celle de UUR. *Éléments de réponse*

$$classe(UR) = \{UR, RU, D\}$$

$$classe(UUR) = \{UUR, URU, RUU, UD, DU\}$$

Q 3.7 [1 point] Dans la classe d'équivalence de $UURR$, combien y a t-il de mots sans D ? avec un seul D ? Combien y a-t-il de D au maximum ? *Éléments de réponse*

Une méthode pourrait être de les lister tous, et de les compter. Les mots sans D contiennent exactement deux U et deux R c'est facile à compter, $\frac{4!}{2!2!} = 6$ les mots avec un seul D comportent exactement un U, un R et un D donc il y en a $3! = 6$. Il y a au maximum 2 D. (Et il n'y a qu'un seul mot comme cela on retrouve 13=6+6+1)

On note $C(n,m)$ le nombre mots dans la classe d'équivalence de U^nR^m où U est répété n fois et R est répété m fois.

Q 3.8 [1 point] Quelle relation de récurrence lie $C(n,m)$ aux nombres $C(n-1,m)$, $C(n-1,m-1)$ et $C(n,m-1)$? Vous justifierez soigneusement. *Éléments de réponse*

Pour atteindre le point de coordonnées (n,m). de trois choses l'une
- ou bien le dernier pas est vers le haut
- ou bien le dernier pas est en diagonale
- ou bien le dernier pas est vers la droite

Il suffit de compter le nombre de manière d'atteindre ces trois points et d'utiliser le principe de la somme.

Dans la suite, on définit $Dc(n) = C(n, n)$.

Q 3.9 [1 point] Démontrez que

$$Dc(n) = \sum_{k=0}^{n} \binom{n+k}{k} \binom{n}{k}.$$

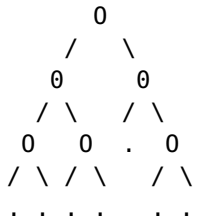
Éléments de réponse

j'appelle k le nombre de pas à droite accompli. il faut qu'il y ait autant de pas vers le haut. pour atteindre le point de coordonnées (n, n) il faut faire encore $(n-k)$ pas en diagonale. le mot décrivant le chemin est donc de longueur $(n-k+k+k)$ c'est à dire $(n+k)$. Combien y a-t-il de tels mots ? Il faut choisir la place des R puis celle des U puis on complète par des D. Puis on utilise le principe de la somme. le nombre k de pas à droite peut être $0, 1, 2, ..., n$

Exercice 4 [5 points]

On veut réaliser une fonction `arbroazar` en Python qui fabrique un arbre binaire étiqueté par `None`, de taille n, choisi pseudo-aléatoirement. On souhaite que la probabilité soit uniforme sur l'ensemble des arbres de taille n. C'est-à-dire que chaque arbre de taille n possède exactement la même probabilité d'être fabriqué.

Q 4.1 [1 point] Dessinez un arbre binaire de taille 6. Quelle est la probabilité qu'il soit fabriqué par un appel à la fonction `arbroazar` (supposée réalisée) ? *Éléments de réponse*



La proba est

$$\frac{1}{C_6} = \frac{7 \times 6! \times 6!}{12!}$$

T.S.V.P.

Une classe `Binary_Tree` a été réalisée en Python. Elle permet de construire soit un arbre vide, soit un arbre binaire dont on donne les deux sous-arbres gauche et droit. Voici une trace d'une session utilisant cette classe :

```
>>> a = Binary_Tree()
>>> a.is_empty()
True
>>> b = Binary_Tree(a,a)
>>> b.is_empty()
False
>>> print(b)
()
>>> c = Binary_Tree(a,b)
>>> print(c)
()()
```

On donne maintenant le code de la fonction f paramétrée par un entier naturel.

```
def f(n):
    """ :CU: n>=0 """
    if n==0:
        return Binary_Tree()
    else:
        k=randint(0,n-1)
        g=f(k)
        d=f(n-1-k)
        return Binary_Tree(g, d)
```

Q 4.2 [1 point] Que vaut $f(n)$? *Éléments de réponse*

La fonction renvoie un arbre binaire de taille n.

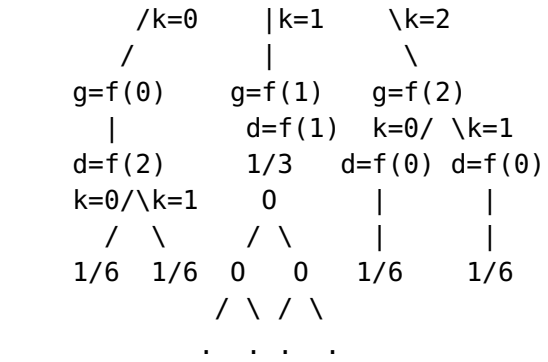
Q 4.3 [1 point] Dessiner tous les résultats possibles pour $f(3)$. Indiquer leur probabilité d'apparition, en supposant l'indépendance, et l'uniformité des tirages aléatoires fournit par `randint`.

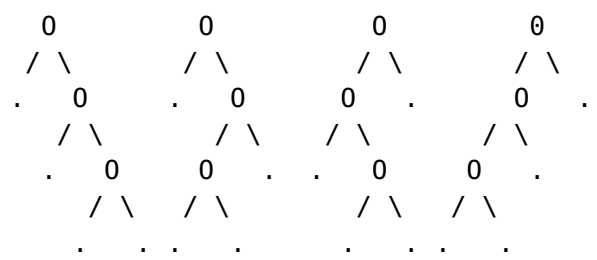
```
. désigne l'arbre vide
Il convient de remarquer que
f(0) = .
f(1) = O
      / \
     .   .

f(2) = ou bien  O     ou bien  O     avec 1/2
               / \             / \
              O   .           .   O
             / \                 / \
            .   .               .   .

pour f(3)

         premier tirage
         /      |      \
```





Dans la suite n désigne un entier, et k un entier tel que $k < n$

Q 4.4 [1 point] Combien y a-t-il d'arbres de taille n dont le sous arbre gauche est de taille égale à k. Donnez le code d'une fonction nommee `nb_arbre` paramétrée par n et k qui calcule ce nombre.

Éléments de réponse

$$C_k \times C_{n-1-k}$$

```
def nb_arbre(n,k):
    return catalan(k)*catalan(n-1-k)
```

Voici le code d'une fonction, la fonction `catalan` attend un entier n en paramètre et renvoie le nombre d'arbres binaires de taille n dont les noeuds contiennent `None` .

```
def determine_indice(n,numero):
    k=0
    cumul =0
    while cumul<=numero:
        cumul+=catalan(k)*catalan(n-1-k)
        k+=1
    return k-1
```

Les contraintes d'utilisation de cette fonction `determine_indice` sont

- `n` est un entier naturel et
- `numero` est un entier naturel inférieur strictement à `catalan(n)`.

Elle permet d'associer à chaque numéro d'arbre binaire de taille n, la taille de son sous-arbre gauche.

Q 4.5 [1 point] En utilisant la fonction `determine_indice`. Proposez une implantation, de la fonction `arbroazar` *Éléments de réponse*

```
def arbroazar(n):
    if n==0:
        return Binary_Tree()
    else:
        num=randint(0,catalan(n)-1)
        k=determine_indice(n,num)
        g=arbroazar(k)
        d=arbroazar(n-1-k)
        return Binary_Tree(g, d)
```