- HAI503I: Algorithmique 4 -

Cours 1 : Rappels de complexité, probabilités

L3 informatique
Université de Montpellier

1. Rappels de complexité
2. Rappels (?) de probabilités discrètes
3. Bits et entiers aléatoires ou pseudo-aléatoires
4. Tirage de réels, simulation de lois
5. Bornes des queues de distribution

1. Rappels de complexité

2. Rappels (?) de probabilités discrètes

3. Bits et entiers aléatoires ou pseudo-aléatoires

4. Tirage de réels, simulation de lois

5. Bornes des queues de distribution

Exemple de base

```
<inst. 1>;
pour i = 1 à n faire
    <inst. 2>;
pour i = 1 à n faire
    pour j = 1 à n faire
        <inst. 3>;
retourner var
```

- <inst. N> : opérations élémentaires

Exemple de base

```
1 <inst. 1>;
2 pour i = 1 à n faire
3 |  <inst. 2>;
4 pour i = 1 à n faire
5 |  pour j = 1 à n faire
6 |  |  <inst. 3>;
7 retourner var
```

- <inst. N> : opérations élémentaires
- L1 et L7 : $O(1)$ (**Hypothèse WORD-RAM**)
- L2 exécute n fois L3 : $O(n)$
- L5 exécute n fois L6 : $O(n)$
- L4 exécute n fois L5 : $O(n^2)$

Total

$2 \times O(1) + O(n) + O(n^2) = O(n^2)$

Exemple de base

```
1 <inst. 1>;
2 pour i = 1 à n faire
3 |   <inst. 2>;
4 pour i = 1 à n faire
5 |   pour j = 1 à n faire
6 |   |   <inst. 3>;
7 retourner var
```

- `<inst. N>` : opérations élémentaires
- L1 et L7 : $O(1)$ (**Hypothèse WORD-RAM**)
- L2 exécute n fois L3 : $O(n)$
- L5 exécute n fois L6 : $O(n)$
- L4 exécute n fois L5 : $O(n^2)$

Total

$2 \times O(1) + O(n) + O(n^2) = O(n^2)$

En clair :

'Mon algo. a une complexité $O(n^2)$ (où $n =$ taille de l'entrée)'
$\rightsquigarrow$ si n est assez grand, le nb. d'opérations est $\leq$ constante$\times n^2$

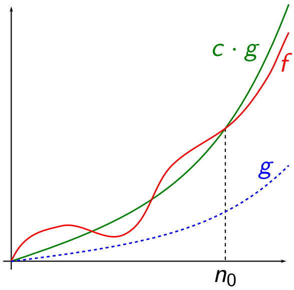
Notations de Landau

« Grand O »

Soit $f, g : \mathbb{N} \to \mathbb{R}_+$. Alors $\boldsymbol{f = O(g)}$ si

$$\exists c > 0, n_0 \geq 0, \forall n \geq n_0, f(n) \leq c \cdot g(n).$$

« f est un grand O de g s'il existe une constante c et un entier n_0 tels que pour toute valeur n plus grande que n_0, $f(n)$ est inférieur ou égal à $c \cdot g(n)$ »



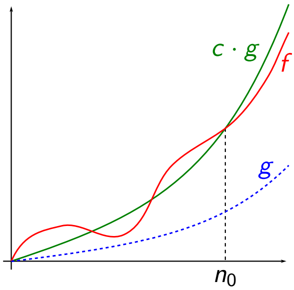
Notations de Landau

« Grand O »

Soit $f, g : \mathbb{N} \to \mathbb{R}_+$. Alors $\boldsymbol{f = O(g)}$ si

$$\exists c > 0, n_0 \geq 0, \forall n \geq n_0, f(n) \leq c \cdot g(n).$$

« f est un grand O de g s'il existe une constante c et un entier n_0 tels que pour toute valeur n plus grande que n_0, $f(n)$ est inférieur ou égal à $c \cdot g(n)$ »



À retenir

$f = O(g)$ si pour n suffisamment grand, f est plus petite que g, à une constante multiplicative près.

Utilisation en complexité

- Avantages pour la théorie :
  - Négliger les cas de bases
  - Pas besoin de compter chaque opération en détail
  - Flexibilité sur les opérations élémentaires

Utilisation en complexité

- Avantages pour la théorie :
  - Négliger les cas de bases
  - Pas besoin de compter chaque opération en détail
  - Flexibilité sur les opérations élémentaires
- Avantages pour la pratique :
  - Indépendant des détails de programmation (nb. de variables intermédiaires, ...)
  - Indépendant de l'environnement d'exécution : système d'exploitation, vitesse de la machine, compilateur, ...

Utilisation en complexité

- Avantages pour la théorie :
  - Négliger les cas de bases
  - Pas besoin de compter chaque opération en détail
  - Flexibilité sur les opérations élémentaires

- Avantages pour la pratique :
  - Indépendant des détails de programmation (nb. de variables intermédiaires, ...)
  - Indépendant de l'environnement d'exécution : système d'exploitation, vitesse de la machine, compilateur, ...

Un temps de calcul dépend du moment et de l'endroit.
Un résultat de complexité reste vrai **pour toujours !**

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand** O
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'
- **Comparatifs des fonctions de bases**
  - Les plus grands exposants de polynones *l'emportent*
  - '*Les exponentiels battent les polynomes qui battent les log*'

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'
- **Comparatifs des fonctions de bases**
  - Les plus grands exposants de polynones *l'emportent*
  - '*Les exponentiels battent les polynomes qui battent les log*'
- **Exemple :**

  $n.O(n) + 5n^3 + 3n + 9\log n + 1.4^n$
  $= O(n^2) + O(n^3) + O(n) + O(\log n) + O(1.4^n)$
  $= O(n^3) + O(\log n) + O(1.4^n)$
  $= O(1.4^n)$

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'
- **Comparatifs des fonctions de bases**
  - Les plus grands exposants de polynones *l'emportent*
  - '*Les exponentiels battent les polynomes qui battent les log*'
- **Calculs de limites**

Limite et O

Si $\lim_{n\to+\infty}\left(\frac{f(n)}{g(n)}\right) = c$, pour $c \in \mathbb{R}_+$, alors on a $f = O(g)$ et
si $\lim_{n\to+\infty}\left(\frac{f(n)}{g(n)}\right) = +\infty$ on a $f \neq O(g)$

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'
- **Comparatifs des fonctions de bases**
  - Les plus grands exposants de polynones *l'emportent*
  - '*Les exponentiels battent les polynomes qui battent les log*'
- **Calculs de limites**

Limites et inverses

Si f et g sont des fonctions strictements positives alors on a :

- Si $\exists c > 0$ telle que $\frac{f(n)}{g(n)} \underset{n \to +\infty}{\longrightarrow} c$, **alors** $f = O(g)$ **et** $g = O(f)$

- Si $\frac{f(n)}{g(n)} \underset{n \to +\infty}{\longrightarrow} 0$, **alors** $f = O(g)$ **et** $g \neq O(f)$**.**

- Si $\frac{f(n)}{g(n)} \underset{n \to +\infty}{\longrightarrow} +\infty$, **alors** $f \neq O(g)$ **et** $g = O(f)$**.**

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
  - Les O se multiplient entre eux
  - Dans une somme, '*seul le plus grand O survit*'
- **Comparatifs des fonctions de bases**
  - Les plus grands exposants de polynones *l'emportent*
  - '*Les exponentiels battent les polynomes qui battent les log*'
- **Calculs de limites**

Limites et inverses

Si f et g sont des fonctions strictements positives alors on a :

- Si $\exists c > 0$ telle que $\frac{f(n)}{g(n)} \underset{n\to+\infty}{\longrightarrow} c$, **alors** $f = O(g)$ **et** $g = O(f)$

- Si $\frac{f(n)}{g(n)} \underset{n\to+\infty}{\longrightarrow} 0$, **alors** $f = O(g)$ **et** $g \neq O(f)$**.**

- Si $\frac{f(n)}{g(n)} \underset{n\to+\infty}{\longrightarrow} +\infty$, **alors** $f \neq O(g)$ **et** $g = O(f)$**.**

- **Exemple :**

$f(n) = 2n$ et $g(n) = 10\sqrt{n}\log(n)$

On a $\frac{f(n)}{g(n)} = \frac{2n}{10\sqrt{n}\log(n)} = \frac{\sqrt{n}}{5\log(n)}$

Et $\frac{f(n)}{g(n)} \underset{n\to+\infty}{\longrightarrow} +\infty$

Donc $f \neq O(g)$ et $g = O(f)$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. `<inst. 1>`
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. `<inst. 2>`

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$

$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$
$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$
$t(n) \leq 7c + 8.t(n-3) \leq \dots$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$
$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$
$t(n) \leq 7c + 8.t(n-3) \leq \ldots$
$t(n) \leq (2^i - 1)c + 2^i.t(n-i) \leq \ldots$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$
$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$
$t(n) \leq 7c + 8.t(n-3) \leq \ldots$
$t(n) \leq (2^i - 1)c + 2^i.t(n-i) \leq \ldots$
$t(n) \leq (2^n - 1)c + 2^n.t(0)$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

ALGOREC1(n) :

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$

$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$

$t(n) \leq 7c + 8.t(n-3) \leq \dots$

$t(n) \leq (2^i - 1)c + 2^i.t(n-i) \leq \dots$

$t(n) \leq (2^n - 1)c + 2^n.t(0)$

Finalement

$t(n) = O(2^n)$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - **Exemple :**

$\textsc{AlgoRec1}(n)$:

1. <inst. 1>
2. *AlgoRec1*$(n-1)$
3. *AlgoRec1*$(n-1)$
4. <inst. 2>

On note **t(n)** la complexité et **c** un majorant du temps des op. élémentaires

$t(n) \leq c + 2.t(n-1) \leq c + 2(c + 2.t(n-2))$
$t(n) \leq 3c + 4.t(n-2) \leq 3c + 4(c + 2.t(n-3))$
$t(n) \leq 7c + 8.t(n-3) \leq \ldots$
$t(n) \leq (2^i - 1)c + 2^i.t(n-i) \leq \ldots$
$t(n) \leq (2^n - 1)c + 2^n.t(0)$

Finalement

$t(n) = O(2^n)$

Pour prouver la ligne de calcul rouge, il faudrait une récurrence propre !

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**

Master Theorem

S'il existe trois entiers $a \geq 0$, $b > 1$, $d \geq 0$ et $n_0 > 0$ tels que pour tout $n \geq n_0$,

$$T(n) \leq aT(\lceil n/b \rceil) + O(n^d)$$

Alors

$$T(n) = \begin{cases} O(n^d) & \text{si } b^d > a \\ O(n^d \log n) & \text{si } b^d = a \\ O(n^{\frac{\log a}{\log b}}) & \text{si } b^d < a \end{cases}$$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**

Master Theorem

S'il existe trois entiers $a \geq 0$, $b > 1$, $d \geq 0$ et $n_0 > 0$ tels que pour tout $n \geq n_0$,

$$\boldsymbol{T(n) \leq aT(\lceil n/b \rceil) + O(n^d)}$$

Alors

$$\boldsymbol{T(n) = \begin{cases} O(n^d) & \text{si } b^d > a \\ O(n^d \log n) & \text{si } b^d = a \\ O(n^{\frac{\log a}{\log b}}) & \text{si } b^d < a \end{cases}}$$

- **Exemple :** Tri-Fusion :
  - $t(n) \leq 2t(\lceil n/2 \rceil) + O(n)$: $a = 2$, $b = 2$, $d = 1$
  - Cas $b^d = a$: on obtient $t(n) = O(n^d \log n) = O(n \log n)$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...

AlgoExTq1(n) :

1. $i \leftarrow 0$
2. **Tant que** $i < n$
3. **Si** i est impair, **alors** $i \leftarrow i - 1$
4. **Si** i est pair, **alors** $i \leftarrow i + 3$
5. **Retourner** n

AlgoExTq2(n) :

1. $i \leftarrow n$
2. **Tant que** $i \geq 1$
3. **Si** i est impair, **alors** $i \leftarrow 3i + 1$
4. **Si** i est pair, **alors** $i \leftarrow i/2$
5. **Retourner** n

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...

AlgoExTq1(n) :

1. $i \leftarrow 0$
2. **Tant que** $i < n$
3. **Si** i est impair, **alors** $i \leftarrow i - 1$
4. **Si** i est pair, **alors** $i \leftarrow i + 3$
5. **Retourner** n

Complexité en $O(n)$

AlgoExTq2(n) :

1. $i \leftarrow n$
2. **Tant que** $i \geq 1$
3. **Si** i est impair, **alors** $i \leftarrow 3i + 1$
4. **Si** i est pair, **alors** $i \leftarrow i/2$
5. **Retourner** n

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...

AlgoExTq1(n) :

1. $i \leftarrow 0$
2. **Tant que** $i < n$
3. **Si** i est impair, **alors** $i \leftarrow i - 1$
4. **Si** i est pair, **alors** $i \leftarrow i + 3$
5. **Retourner** n

Complexité en $O(n)$

AlgoExTq2(n) :

1. $i \leftarrow n$
2. **Tant que** $i \geq 1$
3. **Si** i est impair, **alors** $i \leftarrow 3i + 1$
4. **Si** i est pair, **alors** $i \leftarrow i/2$
5. **Retourner** n

Complexité ?, terminaison ?

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...
  - Un exemple classique :

$\textsc{AlgoExTq3}(n)$:

1. $i \leftarrow n$
2. **Tant que** $i > 1$
3. $\quad i \leftarrow i/2$
4. **Retourner** n

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...
  - Un exemple classique :

ALGOEXTQ3(n) :

1. $i \leftarrow n$
2. **Tant que** $i > 1$
3. $\quad i \leftarrow i/2$
4. **Retourner** n

- Notons i_k la valeur de i au $k$ème tour de boucle.
- $i_0 = n$ et $i_k = i_{k-1}/2$
- Donc on a : $i_k = i_{k-1}/2 = i_{k-2}/4 = \cdots = i_0/2^k = n/2^k$
- $i_k \leq 1$ dès que $n/2^k \leq 1$, cad $n \leq 2^k$ et $\log n \leq k$

Rappels de calculs avec les « grand O »

- **Cas des appels récursifs**
  - Dérouler le calcul de complexité
  - Appel récursif sur une fraction de l'entrée : **Master Theorem**
- **Cas des boucles** *'Tant que'*
  - Il faut se débrouiller parfois *'à la main'*...
  - Un exemple classique :

AlgoExTq3(n) :

1. $i \leftarrow n$
2. **Tant que** $i > 1$
3. $\quad i \leftarrow i/2$
4. **Retourner** n

Complexité en $O(\log n)$

- Notons i_k la valeur de i au $k$ème tour de boucle.
- $i_0 = n$ et $i_k = i_{k-1}/2$
- Donc on a : $i_k = i_{k-1}/2 = i_{k-2}/4 = \cdots = i_0/2^k = n/2^k$
- $i_k \leq 1$ dès que $n/2^k \leq 1$, cad $n \leq 2^k$ et $\log n \leq k$

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand** O
- **Comparatifs des fonctions de bases**
- **Calculs de limites**
- **Cas des appels récursifs**
- **Cas des boucles** '*Tant que*'

Rappels de calculs avec les « grand O »

- **Sommes et produits de grand O**
- **Comparatifs des fonctions de bases**
- **Calculs de limites**
- **Cas des appels récursifs**
- **Cas des boucles** '*Tant que*'

- **Où trouver des rappels, s'entraîner ?**
  - Cours de L2, Complexité et Master Theorem disponible sous le Moodle du cours
  - En profiter pour revoir les règles de calcul des **logarithmes et exponentiels**, ainsi que les **sommes arithmétiques et géométriques**
  - Fiche de TD1

Probabilité

Le langage des probabilités permet de **modéliser** une expérience aléatoire, probabiliste.

Probabilité

Le langage des probabilités permet de **modéliser** une expérience aléatoire, probabiliste.

- Un tirage à pile ou face
- Un lancer de dé
- Le nombre de personnes dans une file d'attente
- Le tirage d'un nombre au hasard
- Une instance d'un algorithme prise au hasard
- ...

Probabilité

Le langage des probabilités permet de **modéliser** une expérience aléatoire, probabiliste.

- Un tirage à pile ou face
- Un lancer de dé
- Le nombre de personnes dans une file d'attente
- Le tirage d'un nombre au hasard
- Une instance d'un algorithme prise au hasard
- ...

Dans ce cours, on va s'en servir pour :

- Etudier la génération d'objets aléatoires sur machine
- Etudier des algorithmes probabilistes
- Analyser le comportement d'algorithmes déterministes 'en moyenne'

Probabilité

Espace probabilisé *discret*

- **Univers** : ensemble des résultats possibles de l'expérience probabiliste, souvent noté Ω
  - **Évènement primitif** : un élément de l'univers / un résultat possible
  - **Évènement** : sous-ensemble de l'univers / ensemble de résultats possibles

Probabilité

Espace probabilisé *discret*

- **Univers** : ensemble des résultats possibles de l'expérience probabiliste, souvent noté Ω
  - **Évènement primitif** : un élément de l'univers / un résultat possible
  - **Évènement** : sous-ensemble de l'univers / ensemble de résultats possibles
- **Probabilités** : une valeur **positive** pour chaque élément primitif x, notée $\Pr[x]$
  - **Probabilité d'un évènement** : somme des probabilités de ses éléments
  - Par convention : **somme totale** $= 1$

Probabilité

Espace probabilisé *discret*

- **Univers** : ensemble des résultats possibles de l'expérience probabiliste, souvent noté Ω
  - **Évènement primitif** : un élément de l'univers / un résultat possible
  - **Évènement** : sous-ensemble de l'univers / ensemble de résultats possibles
- **Probabilités** : une valeur **positive** pour chaque élément primitif x, notée $\Pr[x]$
  - **Probabilité d'un évènement** : somme des probabilités de ses éléments
  - Par convention : **somme totale** $= 1$

Exemple 1 : dé équilibré à 6 faces

- Univers : $\Omega = \{$⚀, ⚁, ⚂, ⚃, ⚄, ⚅$\}$
- Probabilité associée à chaque évènement : $\frac{1}{6}$
- $\Pr[$dé au moins 5$] = \Pr[\{$⚄, ⚅$\}] = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers :

$$\begin{aligned}\Omega = \{&(\text{⚀},\text{⚀}),(\text{⚀},\text{⚁}),(\text{⚀},\text{⚂}),(\text{⚀},\text{⚃}),(\text{⚀},\text{⚄}),(\text{⚀},\text{⚅}),\\
&(\text{⚁},\text{⚀}),(\text{⚁},\text{⚁}),(\text{⚁},\text{⚂}),(\text{⚁},\text{⚃}),(\text{⚁},\text{⚄}),(\text{⚁},\text{⚅}),\\
&(\text{⚂},\text{⚀}),(\text{⚂},\text{⚁}),(\text{⚂},\text{⚂}),(\text{⚂},\text{⚃}),(\text{⚂},\text{⚄}),(\text{⚂},\text{⚅}),\\
&(\text{⚃},\text{⚀}),(\text{⚃},\text{⚁}),(\text{⚃},\text{⚂}),(\text{⚃},\text{⚃}),(\text{⚃},\text{⚄}),(\text{⚃},\text{⚅}),\\
&(\text{⚄},\text{⚀}),(\text{⚄},\text{⚁}),(\text{⚄},\text{⚂}),(\text{⚄},\text{⚃}),(\text{⚄},\text{⚄}),(\text{⚄},\text{⚅}),\\
&(\text{⚅},\text{⚀}),(\text{⚅},\text{⚁}),(\text{⚅},\text{⚂}),(\text{⚅},\text{⚃}),(\text{⚅},\text{⚄}),(\text{⚅},\text{⚅})\}\end{aligned}$$

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers :

$$\begin{aligned}\Omega = \{&(\epsdice{1},\epsdice{1}),(\epsdice{1},\epsdice{2}),(\epsdice{1},\epsdice{3}),(\epsdice{1},\epsdice{4}),(\epsdice{1},\epsdice{5}),(\epsdice{1},\epsdice{6}),\\
&(\epsdice{2},\epsdice{1}),(\epsdice{2},\epsdice{2}),(\epsdice{2},\epsdice{3}),(\epsdice{2},\epsdice{4}),(\epsdice{2},\epsdice{5}),(\epsdice{2},\epsdice{6}),\\
&(\epsdice{3},\epsdice{1}),(\epsdice{3},\epsdice{2}),(\epsdice{3},\epsdice{3}),(\epsdice{3},\epsdice{4}),(\epsdice{3},\epsdice{5}),(\epsdice{3},\epsdice{6}),\\
&(\epsdice{4},\epsdice{1}),(\epsdice{4},\epsdice{2}),(\epsdice{4},\epsdice{3}),(\epsdice{4},\epsdice{4}),(\epsdice{4},\epsdice{5}),(\epsdice{4},\epsdice{6}),\\
&(\epsdice{5},\epsdice{1}),(\epsdice{5},\epsdice{2}),(\epsdice{5},\epsdice{3}),(\epsdice{5},\epsdice{4}),(\epsdice{5},\epsdice{5}),(\epsdice{5},\epsdice{6}),\\
&(\epsdice{6},\epsdice{1}),(\epsdice{6},\epsdice{2}),(\epsdice{6},\epsdice{3}),(\epsdice{6},\epsdice{4}),(\epsdice{6},\epsdice{5}),(\epsdice{6},\epsdice{6})\}\end{aligned}$$

C'est fastidieux... Souvent on ne donne pas Ω explicitement...

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers : $\Omega = \{(⚀, ⚀), (⚀, ⚁), (⚀, ⚂), \dots\}$
- Probabilité associée à chaque évènement : $\frac{1}{36}$

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers : $\Omega = \{($⚀, ⚀$), ($⚀, ⚁$), ($⚀, ⚂$), \ldots\}$
- Probabilité associée à chaque évènement : $\frac{1}{36}$
- Pr[un des deux dés est $\geq 3] = \frac{32}{36}$
- Pr[la somme des deux dés est paire] $= \cdots$

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers : $\Omega = \{(\text{⚀}, \text{⚀}), (\text{⚀}, \text{⚁}), (\text{⚀}, \text{⚂}), \ldots\}$
- Probabilité associée à chaque évènement : $\frac{1}{36}$
- $\Pr[\text{un des deux dés est } \geq 3] = \frac{32}{36}$
- $\Pr[\text{la somme des deux dés est paire}] = \cdots$

Quand tous les évènements primitifs ont même probabilité, on parle de probabilité **uniforme**.

Probabilité

Exemple 2 : jet de **deux** dés équilibrés à 6 faces

- Univers : $\Omega = \{$(⚀, ⚀), (⚀, ⚁), (⚀, ⚂), $\dots\}$
- Probabilité associée à chaque évènement : $\frac{1}{36}$
- $\mathrm{Pr}[\text{un des deux dés est } \geq 3] = \frac{32}{36}$
- $\mathrm{Pr}[\text{la somme des deux dés est paire}] = \cdots$

Quand tous les évènements primitifs ont même probabilité, on parle de probabilité **uniforme**.

Dans ce cas là, en notant p la probabilité d'un évènement primitif, on a :
$\mathrm{Pr}[\Omega] = p.|\Omega| = 1$ donc $p = \frac{1}{|\Omega|}$
Et la probabilité d'un évènement A vaut :

$$\mathrm{Pr}[A] = p.|A| = \frac{|A|}{|\Omega|} = \text{"}\frac{\text{nb cas favorables}}{\text{nb cas possibles}}\text{"}$$

Variable aléatoire

Définitions

- Une **variable aléatoire** est une fonction $X : \Omega \to V$ avec $V \subseteq \mathbb{R}$
- Pour $v \in V$:
  - « $X = v$ » est l'évènement $\{\omega \in \Omega : X(\omega) = v\}$ et $\Pr[X = v] = \sum_{\omega : X(\omega) = v} \Pr[\omega]$
  - « $X \leq v$ » est l'évènement $\{\omega \in \Omega : X(\omega) \leq v\}$ et $\Pr[X \leq v] = \sum_{\omega : X(\omega) \leq v} \Pr[\omega]$
  - etc...

Variable aléatoire

Définitions

- Une **variable aléatoire** est une fonction $X : \Omega \to V$ avec $V \subseteq \mathbb{R}$
- Pour $v \in V$:
  - « $X = v$ » est l'évènement $\{\omega \in \Omega : X(\omega) = v\}$ et $\Pr[X = v] = \sum_{\omega : X(\omega) = v} \Pr[\omega]$
  - « $X \leq v$ » est l'évènement $\{\omega \in \Omega : X(\omega) \leq v\}$ et $\Pr[X \leq v] = \sum_{\omega : X(\omega) \leq v} \Pr[\omega]$
  - etc...

Une variable aléatoire n'est ni une variable, ni aléatoire ☺

Exemple : dé équilibré à 6 faces

- Nombre de point obtenus :
  - X : {⚀, ⚁, ⚂, ⚃, ⚄, ⚅} $\to \{1, 2, 3, 4, 5, 6\}$
  - $\Pr[X = t] = \frac{1}{6}$ pour tout $t \in \{1, 2, 3, 4, 5, 6\}$ et $\Pr[X \leq 4] = \frac{2}{3}$
- Parité du dé :
  - Y : {⚀, ⚁, ⚂, ⚃, ⚄, ⚅} $\to \{0, 1\}$
  - $\Pr[Y = 0] = \Pr[Y = 1] = \frac{1}{2}$

Espérance

Définition

Soit $X : \Omega \to V$ une variable aléatoire. **L'espérance de X** est :

$$\mathbb{E}[X] = \sum_{v \in V} v \times \Pr[X = v]$$

Remarques

- Intuitivement : résultat obtenu *en moyenne*

Espérance

Définition

Soit $X : \Omega \to V$ une variable aléatoire. **L'espérance de X** est :

$$\mathbb{E}[X] = \sum_{v \in V} v \times \Pr[X = v]$$

Remarques

- Intuitivement : résultat obtenu *en moyenne*

Exemple : dé équilibré à 6 faces

- $\mathbb{E}[X] = \sum_{v=1}^{6} v \times \Pr[X = v] = \sum_{v=1}^{6} \frac{v}{6} = \frac{1}{6}.\sum_{v=1}^{6} v = \frac{7}{2}$
- $\mathbb{E}[Y] = 0 \times \Pr[Y = 0] + 1 \times \Pr[Y = 1] = 0 \times \frac{1}{2} + 1 \times \frac{1}{2} = \frac{1}{2}$

Probabilité conditionnelle

Définition

- La **probabilité de E sachant F** est $\Pr[E \mid F] = \frac{\Pr[E \cap F]}{\Pr[F]}$
- L'**espérance de X sachant F** est
  $\mathbb{E}[X \mid F] = \sum_{v \in V} t \times \Pr[X = v \mid F]$

Remarque

- C'est la probabilité que E arrive sachant que F s'est produit, et l'espérance de X sachant que F s'est produit.
- Cela revient à 'réduire' Ω aux évènements de F.

Probabilité conditionnelle

Définition

- La **probabilité de E sachant F** est $\Pr[E \mid F] = \frac{\Pr[E \cap F]}{\Pr[F]}$
- L'**espérance de X sachant F** est $\mathbb{E}[X \mid F] = \sum_{v \in V} t \times \Pr[X = v \mid F]$

Remarque

- C'est la probabilité que E arrive sachant que F s'est produit, et l'espérance de X sachant que F s'est produit.
- Cela revient à 'réduire' Ω aux évènements de F.

Exemple : dé équilibré à 6 faces

- $\Pr[X \geq 4 \mid Y = 0] = \Pr[\{X \geq 4\} \cap \{Y = 0\}]/\Pr[Y = 0] = \frac{1}{3}/\frac{1}{2} = \frac{2}{3}$
- $\mathbb{E}[X \mid Y = 1] = \sum_{v \in \{1,\dots,6\}} v \times \Pr[X = v \mid Y = 1] =$
  $1 \times \frac{1}{3} + 2 \times 0 + 3 \times \frac{1}{3} + 4 \times 0 + 5 \times \frac{1}{3} + 6 \times 0 = 3$

Indépendance

Définition

- Deux évènements sont **indépendants** si $\Pr[E \wedge F] = \Pr[E].\Pr[F]$
- Deux variables aléatoires sont indépendantes si $\Pr[X = u \wedge Y = v] = \Pr[X = u].\Pr[Y = v]$ pour tous u, v

Remarque

En proba, l'usage est d'utiliser des notations logiques ($\vee$, $\wedge$, $\neg$) plus que des notations ensemblistes ($\cup$, $\cap$, ...)

Indépendance

Définition

- Deux évènements sont **indépendants** si $\Pr[E \wedge F] = \Pr[E].\Pr[F]$
- Deux variables aléatoires sont indépendantes si $\Pr[X = u \wedge Y = v] = \Pr[X = u].\Pr[Y = v]$ pour tous u, v

Remarque

En proba, l'usage est d'utiliser des notations logiques ($\vee$, $\wedge$, $\neg$) plus que des notations ensemblistes ($\cup$, $\cap$, ...)

Exemples

Indépendance

Définition

- Deux évènements sont **indépendants** si $\Pr[E \wedge F] = \Pr[E].\Pr[F]$
- Deux variables aléatoires sont indépendantes si
  $\Pr[X = u \wedge Y = v] = \Pr[X = u].\Pr[Y = v]$ pour tous u, v

Remarque

En proba, l'usage est d'utiliser des notations logiques ($\vee$, $\wedge$, $\neg$) plus que des notations ensemblistes ($\cup$, $\cap$, ...)

Exemples

- On lance deux dés, disons un bleu et un rouge.
- $E = \{$la valeur du dé rouge est paire $\}$
  $F = \{$la valeur du dé bleu est 3 ou 4 $\}$
  $G = \{$la valeur du dé rouge est 1 ou 2 ou 3$\}$
- E et F sont indépendants, E et G ne le sont pas...

Indépendance

Définition

- Deux évènements sont **indépendants** si $\Pr[E \wedge F] = \Pr[E].\Pr[F]$
- Deux variables aléatoires sont indépendantes si
  $\Pr[X = u \wedge Y = v] = \Pr[X = u].\Pr[Y = v]$ pour tous u, v

Remarque

En proba, l'usage est d'utiliser des notations logiques ($\vee$, $\wedge$, $\neg$) plus que des notations ensemblistes ($\cup$, $\cap$, ...)

Exemples

- On lance deux dés, disons un bleu et un rouge.
- X = la valeur du dé rouge
  Y = la valeur du dé bleu
  Z = la somme des valeurs des dés rouge et bleu
- X et Y sont indépendantes, X et Z ne le sont pas...

Propriétés

Propriétés

- Soit E et F deux évènements :
  - $\Pr[\neg E] = 1 - \Pr[E]$
  - $\Pr[E \vee F] \leq \Pr[E] + \Pr[F]$ **Inégalité de Boole** ou **'Union bound'**
- Soit $X, Y : \Omega \to V$ deux variables aléatoires :
  - $\sum_{v \in V} \Pr[X = v] = 1$
  - $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ **Linéarité de l'espérance**
- Si $\Omega = \bigsqcup_i F_i$, partition de Ω en $(F_i)_i$
  - $\Pr[E] = \sum_i \Pr[E \mid F_i].\Pr[F_i]$ formule des **probabilités totales**
  - $\mathbb{E}[X] = \sum_i \mathbb{E}[X \mid F_i].\Pr[F_i]$ formule de **l'espérance totale**

Exemple d'algorithme probabiliste : calcul de π

Un exemple de la méthode de Monte-Carlo :

CalculPi(n) :

1. $c \leftarrow 0$
2. **Répéter** n fois :
3. $\quad x \leftarrow$ réel aléatoire entre 0 et 1
4. $\quad y \leftarrow$ réel aléatoire entre 0 et 1
5. $\quad$ **Si** $x^2 + y^2 \leq 1$ **alors** $c \leftarrow c + 1$
6. **Renvoyer** $4c/n$

Exemple d'algorithme probabiliste : calcul de π

Un exemple de la méthode de Monte-Carlo :

$\textsc{CalculPi}(n)$:

1. $c \leftarrow 0$
2. **Répéter** n fois :
3. $\quad x \leftarrow$ réel aléatoire entre 0 et 1
4. $\quad y \leftarrow$ réel aléatoire entre 0 et 1
5. $\quad$ **Si** $x^2 + y^2 \leq 1$ **alors** $c \leftarrow c + 1$
6. **Renvoyer** $4c/n$

Comment tirer des réels aléatoires ?

- Comment faire algorithmiquement (théorie) ?
- Comment faire sur un ordinateur (pratique) ? (Cf exemple...)

Brique de base : Bits aléatoires

Théoriquement, dans un monde idéal :

- Accès à des *bits aléatoires* :
  - Une fonction $\textsc{RandomBit}()$ qui renvoie 0 ou 1 avec probabilité $\frac{1}{2}$
  - Appels consécutifs à $\textsc{RandomBit}()$ indépendants
- Construction d'*objets* aléatoires à partir des bits
  - entiers, rationnels, lettres d'un alphabet, ...
  - arbres, graphes, permutations, ...
  - reéls, ...
- Simulation de lois
  - bits biaisés : par exemple un bit 0 avec proba $\frac{1}{3}$ et 1 avec proba $\frac{2}{3}$
  - tirer des évènements selon des probabilités non uniformes
  - tirer un point aléatoire dans un cercle, sur une sphère, un donuts...

Brique de base : Bits aléatoires

Théoriquement, dans un monde idéal :

- Accès à des *bits aléatoires* :
  - Une fonction $\textsc{RandomBit}()$ qui renvoie 0 ou 1 avec probabilité $\frac{1}{2}$
  - Appels consécutifs à $\textsc{RandomBit}()$ indépendants
- Construction d'*objets* aléatoires à partir des bits
  - entiers, rationnels, lettres d'un alphabet, ...
  - arbres, graphes, permutations, ...
  - reéls, ...
- Simulation de lois
  - bits biaisés : par exemple un bit 0 avec proba $\frac{1}{3}$ et 1 avec proba $\frac{2}{3}$
  - tirer des évènements selon des probabilités non uniformes
  - tirer un point aléatoire dans un cercle, sur une sphère, un donuts...

Difficulté :

- Comment écrire une fonction $\textsc{RandomBit}()$?
- *Vrai* aléa est-il possible ? miracle quantique ? autres solutions ?
- Comment faire tout ce qu'on veut si on a accès à un $\textsc{RandomBit}()$ acceptable ?

Le pseudo-aléa

Solution déployée actuellement

- Générateurs *pseudo*-aléatoires
  - algorithme qui produit des bits qui *semblent* aléatoires
  - aspects théoriques et pratiques satisfaisants
- Générateurs de bits, mais aussi directement **d'entiers**, flottants, etc.
- Gérés dans des bibliothèques logicielles $\rightarrow$ `random` dans Python

Le pseudo-aléa

Solution déployée actuellement

- Générateurs *pseudo*-aléatoires
  - algorithme qui produit des bits qui *semblent* aléatoires
  - aspects théoriques et pratiques satisfaisants
- Générateurs de bits, mais aussi directement **d'entiers**, flottants, etc.
- Gérés dans des bibliothèques logicielles → `random` dans Python

Remarques

- Restent des algorithmes **déterministes** → suite fixée
- Entrée de l'algorithme : **graîne**
  - changer la graîne doit modifier *complètement* la suite
  - choix de graîne : quelque chose d'*imprévisible* ou au contraire de fixé

Exemple : générateurs congruentiels linéaires

Suite (X_n) définie par $X_{n+1} = (aX_n + c) \bmod m$

- X_0 doit être fixé : *graîne* du générateur
- a, c et m définissent le générateur
- parfois : seuls certains bits de X_n sont utilisés

Quelques choix classiques

- m premier, $c = 0$, a *primitif modulo* m (Lehmer)
- $m = 2^k$, $c = 0$, $a = 3$ ou $5 \bmod 8$
- m et c premiers entre eux, $a - 1$ divisible par les facteurs premiers de m

Exemple : générateurs congruentiels linéaires

Suite (X_n) définie par $X_{n+1} = (aX_n + c) \bmod m$

- X_0 doit être fixé : *graîne* du générateur
- a, c et m définissent le générateur
- parfois : seuls certains bits de X_n sont utilisés

Quelques choix classiques

- m premier, $c = 0$, a *primitif modulo* m (Lehmer)
- $m = 2^k$, $c = 0$, $a = 3$ ou $5 \bmod 8$
- m et c premiers entre eux, $a - 1$ divisible par les facteurs premiers de m

Exemples

- `rand()` de `stdlib.h` : $m = 2^{31}$, $a = 1103515245$, $c = 12345$
- `minstd_rand()` de `C++11` : $m = 2^{31} - 1$, $a = 48271$, $c = 0$
- `java.util.Random()` : $m = 2^{48}$, $a = 25214903917$, $c = 11$, bits 16 à 47
- `random()` de `Python` : algo de Mersenne-Twister, vecteurs de dimension 623, de période $m = 2^{19937} - 1$...

Conclusion sur l'aléa et le pseudo-aléa

Problématique du pseudo-aléa

- Construire des *bons* générateurs est difficile
- Limitations intrinsèques (période, etc.)

Bon, on fait quoi alors ?

- En théorie : on suppose l'accès à des bits parfaitement aléatoires
- En pratique : on utilise les générateurs des bibliothèques, en général suffisants
- Dans les deux cas : on dispose d'une source d'aléa, on simule des lois

Problématique

On sait

- tirer des bits aléatoires, ou
- tirer des entiers aléatoires

On veut

- tirer un nombre réel aléatoire
- tirer des bits *biaisés* : $\Pr[b = 0] \neq \Pr[b = 1]$
- choisir un élément dans un ensemble, *non uniformément* : par ex. un dé qui tombe une fois sur deux sur ⚅
- tirer un graphe aléatoire
- ...

Problématique

On sait

- tirer des bits aléatoires, ou
- tirer des entiers aléatoires

On veut

- tirer un nombre réel aléatoire
- tirer des bits *biaisés* : $\Pr[b = 0] \neq \Pr[b = 1]$
- choisir un élément dans un ensemble, *non uniformément* : par ex. un dé qui tombe une fois sur deux sur ⚅
- tirer un graphe aléatoire
- ...

But

- Construire des **algorithmes** pour ces tirages

Tirer un réel aléatoire, tirer un bit biaisé

Tirer un réel aléatoire entre 0 et 1

- À précision n : tirer n bits aléatoires b_1, ..., b_n et renvoyer $\overline{0, b_1 \cdots b_n}^2$
- En pratique : `random()` dans différents langages renvoie $x \in [0, 1]$

Tirer un réel aléatoire, tirer un bit biaisé

Tirer un réel aléatoire entre 0 et 1

- À précision n : tirer n bits aléatoires $b_1, \ldots, b_n$ et renvoyer $\overline{0, b_1 \cdots b_n}^2$
- En pratique : `random()` dans différents langages renvoie $x \in [0, 1]$

Tirer un bit biaisé

BIT BIAISÉ(p)

1. $x \leftarrow$ réel aléatoire entre 0 et 1
2. Si $x \leq p$: renvoyer 1
3. Sinon : renvoyer 0

Lois de probabilité

Définition

La **loi (ou distribution) d'une variable aléatoire** $X : \Omega \to V$ est la donnée de $\Pr[X = v]$ pour tout $v \in V$.

Exemple

La loi de la somme S de deux dés équilibrés est :

$\Pr[S = 2] = \frac{1}{36}$ $\quad \Pr[S = 3] = \frac{2}{36}$ $\quad \Pr[S = 4] = \frac{3}{36}$

$\Pr[S = 5] = \frac{4}{36}$ $\quad \dots$

Lois de probabilité

Définition

La **loi (ou distribution) d'une variable aléatoire** $X : \Omega \to V$ est la donnée de $\Pr[X = v]$ pour tout $v \in V$.

Exemple

La loi de la somme S de deux dés équilibrés est :
$\Pr[S = 2] = \frac{1}{36}$ $\Pr[S = 3] = \frac{2}{36}$ $\Pr[S = 4] = \frac{3}{36}$
$\Pr[S = 5] = \frac{4}{36}$ $\ldots$

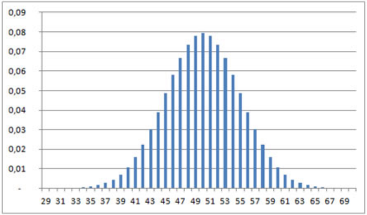
Quelques lois usuelles ($X : \Omega \to V$)

- **Uniforme** : $V = \{1, \ldots, n\}$ et
  $\Pr[X = v] = 1/|V| = 1/n$ pour tout $v \in V$
  Tirage d'un entier entre 1 et $|V|$ $\mathbb{E}[X] = \frac{n+1}{2}$
- **Bernoulli**(p) : $V = \{0, 1\}$ et
  $\Pr[X = 1] = p$ et $\Pr[X = 0] = 1 - p$
  Tirage d'un bit aléatoire biaisé $\mathbb{E}[X] = p$

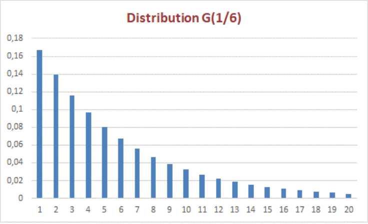
Lois de probabilité

Quelques lois usuelles ($X : \Omega \to V$)

- **Binomiale**(p, n) : $V = \{0, \ldots, n\}$ et
  $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ pour $k = 0, \ldots, n$
  Somme de n variables de Bernoulli de paramètre p $\mathbb{E}[X] = np$
- **Géométrique**(p) : $V = \mathbb{N}$ et $\Pr[X = n] = p(1-p)^{n-1}$ pour $n \in \mathbb{N}$
  Premier apparition du '1' dans une suite de variables de Bernoulli de paramètre p $\mathbb{E}[X] = 1/p$



Loi Binomiale(0.5,100)

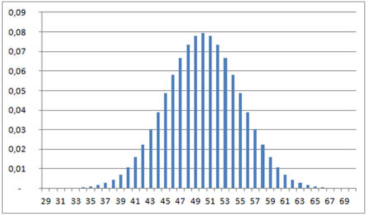


Loi Géométrique$(\frac{1}{6})$

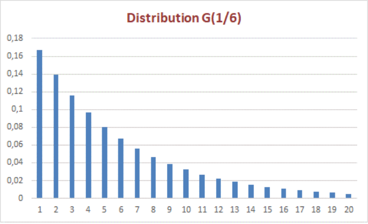
Lois de probabilité

Quelques lois usuelles ($X : \Omega \to V$)

- **Binomiale**(p, n) : $V = \{0, \ldots, n\}$ et
  $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ pour $k = 0, \ldots, n$
  Somme de n variables de Bernoulli de paramètre p $\mathbb{E}[X] = np$
- **Géométrique**(p) : $V = \mathbb{N}$ et $\Pr[X = n] = p(1-p)^{n-1}$ pour $n \in \mathbb{N}$
  Premier apparition du '1' dans une suite de variables de Bernoulli de paramètre p $\mathbb{E}[X] = 1/p$



Loi Binomiale(0.5,100)



Loi Géométrique($\frac{1}{6}$)

En Python, `random.binomial(n, p)`, `random.geometric(p)`....

Exemple de loi plus complexe

Retour à Monte Carlo

Comment tirer (x, y) aléatoirement dans le disque D de centre $(0, 0)$ et de rayon 1 ?

$$D = \{(x, y) \ : \ -1 \leq x, y \leq 1 \text{ et } x^2 + y^2 \leq 1\}$$

- Tirer x et y dans $[-1, 1] \to 2 \cdot \textsc{random}() - 1$
- Assurer que $x^2 + y^2 \leq 1 \to$ méthode du *rejet* (ou de *Monte Carlo*)

Monte Carlo

1. $(x, y) \leftarrow (1, 1)$
2. Tant que $x^2 + y^2 > 1$:
3. $\quad x \leftarrow 2 \cdot \textsc{random}() - 1$
4. $\quad y \leftarrow 2 \cdot \textsc{random}() - 1$
5. Renvoyer (x, y)

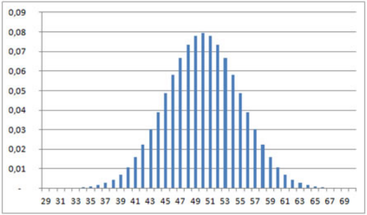
Propriétés

- L'algorithme renvoie bien $(x, y) \in D$
- On peut montrer qu'ils sont uniformément répartis
- Complexité : $\Pr[(x, y) \in D] = \frac{\pi}{4}$ $\Rightarrow$ espérance de $\frac{4}{\pi}$ essais : $O(1)$

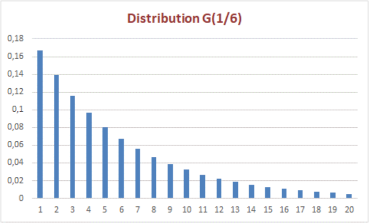
Queues de distribution

Queue d'une distribution

Sur la représentation graphique de la loi, c'est la partie qui s'éloigne de la moyenne.



Loi Binomiale(0.5,100)

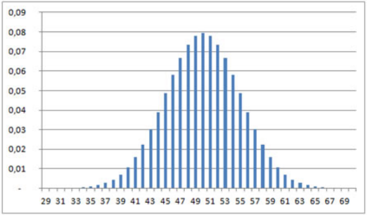


Loi Géométrique$(\frac{1}{6})$

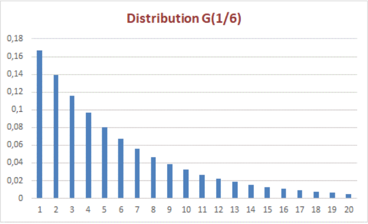
Queues de distribution

Queue d'une distribution

Sur la représentation graphique de la loi, c'est la partie qui s'éloigne de la moyenne.



Loi Binomiale(0.5,100)



Loi Géométrique$(\frac{1}{6})$

La Théorie des Probabilités

Un des buts : fournir des bornes sur les queues de distribution

Queues de distribution

Inégalité de Markov

Soit $X : \Omega \to V$ une variable aléatoire **positive** ($V \subseteq \mathbb{R}^+$).
Pour tout $t > 0$, on a : $\Pr[X \geq t] \leq \frac{1}{t}.\mathbb{E}[X]$

Preuve On a :

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{v \in V} v.\Pr[X = v] \\
&= \sum_{v < t} v.\Pr[X = v] + \sum_{v \geq t} v.\Pr[X = v] \\
&\geq \sum_{v \geq t} v.\Pr[X = v] \qquad \text{car } X \text{ est positive} \\
&\geq t.\sum_{v \geq t} \Pr[X = v] = t.\Pr[X \geq t]
\end{aligned}$$

Bilan final

Utiliser de l'aléa en algorithmique

- On suppose une source parfaite
- On utilise des lois simples — souvent
- Si loi complexe $\rightarrow$ algorithme — rarement
- Implantations :
  - bibliothèque `random` de Python (pseudo-aléa)
  - On ignore la différence avec le *vrai* aléa

Preuves en présence d'aléa

- Aléa parfait $\rightarrow$ probabilités
- Propriétés de base (probabilités conditionnelles, espérance, ...)

Revoir les probabilités

- Exercices au TD1
- Poly « Probabilités discrètes » sur Moodle