Probabilités et simulation

application à l'analyse d'algorithmes et à la randomization

Équipe pédagogique

Jean-Marc.Vincent@imag.fr (Cours)

Arnaud.Legrand@imag.fr (Travaux Dirigés)

Équipe-projet MESCAL
Laboratoire d'Informatique de Grenoble, Inria Rhône-Alpes
Université Joseph Fourier

Analyse d'algorithmes

`Tri-Rapide` (E,n)

Données : E ensemble de n éléments distincts comparables

Résultat : Le même ensemble trié

if $|E| = 1$

  `Return` (E)

else

  x=`choix` (E)

  $E_< = \{y \in E,\ y < x\}$ $E_> = E - E_< - \{x\}$

  `Return` (`Tri-Rapide` $(E_<)$,x,`Tri-Rapide` $(E_>)$)

Complexité :

- au pire : $\mathcal{O}(n^2)$,
- au mieux : $\mathcal{O}(n \log n)$,
- en moyenne : dépend des données en entrée

$\Rightarrow$ **analyse en moyenne** $\mathcal{O}(n \log n)$ **si les données en entrée sont réparties uniformement**

Analyse d'algorithmes

`Tri-Rapide` (E,n)

Données : E ensemble de n éléments distincts comparables

Résultat : Le même ensemble trié

if $|E| = 1$

  `Return` (E)

else

  x=`choix` (E)

  $E_< = \{y \in E,\ y < x\}$ $E_> = E - E_< - \{x\}$

  `Return` (`Tri-Rapide` $(E_<)$,x,`Tri-Rapide` $(E_>)$)

Complexité :

- au pire : $\mathcal{O}(n^2)$,
- au mieux : $\mathcal{O}(n \log n)$,
- en moyenne : dépend des données en entrée

$\Rightarrow$ **analyse en moyenne $\mathcal{O}(n \log n)$ si les données en entrée sont réparties uniformement**

Analyse d'algorithmes

`Tri-Rapide` (E,n)

Données : E ensemble de n éléments distincts comparables

Résultat : Le même ensemble trié

if $|E| = 1$

  `Return` (E)

else

  x=`choix` (E)

  $E_< = \{y \in E,\ y < x\}$ $E_> = E - E_< - \{x\}$

  `Return` (`Tri-Rapide` $(E_<)$,x,`Tri-Rapide` $(E_>)$)

Complexité :

- au pire : $\mathcal{O}(n^2)$,
- au mieux : $\mathcal{O}(n \log n)$,
- en moyenne : dépend des données en entrée

$\Rightarrow$ **analyse en moyenne $\mathcal{O}(n \log n)$ si les données en entrée sont réparties uniformement**

```
Tri-Rapide (E,n)
    Données : E ensemble de n éléments distincts comparables
    Résultat : Le même ensemble trié

    if |E| = 1
      Return (E)
    else
      x=choix-uniforme(E)
      E_<={y ∈ E, y < x} E_> = E − E_< − {x}
      Return (Tri-Rapide (E_<),x,Tri-Rapide (E_>))
```

Complexité :

- au pire : $\mathcal{O}(n^2)$,
- au mieux : $\mathcal{O}(n \log n)$,
- en moyenne :

$\Rightarrow$ **analyse en moyenne indépendante des données en entrée complexité en** $\mathcal{O}(n \log n)$

Contrôle de l'aléatoire : contexte des réseaux

- **canal de communication bruité**
  - taux de bruit $\Rightarrow$ contrôle d'erreur
  - qualification du protocole
  ex : bit de parité, parité horizontale/verticale, CRC...

$\Rightarrow$ maîtriser l'environnement

Contrôle de l'aléatoire : contexte des circuits

- **composants électroniques**
  - durée de vie $\Rightarrow$ mode de fonctionnement dégradé
  - duplication des fonctionnalités
  - dimensionnement

$\Rightarrow$ maîtriser l'environnement

Algorithmes randomisés : contexte des réseaux

- **canal de communication partagé pas de synchronisation**

```
Send (M)
    Données : Un message M à émettre
    Résultat : Le message M délivré

    repeat
        if écoute-porteuse()
            émission-entête()
            if collision-détectée()
                transmission-interrompue()
        else
            attendre un certain temps
    until transmission complète;
```

Exemple de règle : à chaque échec on double l'intervalle *I* et on choisit uniformement sur *I*

⇒ **utiliser *RANDOM* pour mélanger les comportements.**

Algorithmes randomisés : contexte de la cryptographie

cryptage clé publique/privée basé sur la décomposition de grands entiers
avoir de grands nombres premiers
difficiles à générer
générer p avec une quasi-certitude qu'il est premier

Test-Prime(n,k)
Données : Entier n à tester, k nombre de témoins
Résultat : n est composé (sûr) ou fortement probablement premier

Calculer s et d $n-1=2^s d$
for k *fois*
  a= choix uniforme dans $[1, n-1]$
  if $a^d \mod n \neq 1$ *et* $a^{2^r d} \mod n \neq -1$ *pour tous les* r *dans l'intervalle* $[0; s-1]$
    `Return` (composé)
`Return` (probablement premier)

algorithme de Miller-Rabin

$\Rightarrow$ **réduire la complexité d'un algorithme (acceptable).**

Objectif du cours

Objectif

1. Acquérir et maîtriser le langage des probabilités dans le contexte informatique (modélisation)
2. Savoir générer des données distribuées selon une loi donnée (écrire les algorithmes).
3. Savoir construire des plans d'expérience simples et savoir analyser les résultats avec rigueur.

Méthodes

1. Cours : aspects fondamentaux, grands exemples
2. TD : exercices avec résumé du cours, écriture d'algorithmes et de preuves, mise en œuvre
3. Expérimentation dans le cadre d'un mini-projet (éventuellement 2)

Evaluation

1. Examen écrit
2. Contrôle continu : Mini-projets + Quicks

Plan du cours

Partie I : Structures discrètes

1. Analyse de données expérimentales
   TD : Le générateur Random
2. Variables aléatoires de loi discrète
   TD : Transformation de générateurs
3. Génération selon une loi discrète donnée
   TD : Génération de loi discrète : distribution classique / arbitraire / non borné
4. Génération de structures combinatoires, fonction génératrice
   TD : génération uniforme d'arbres
5. Analyse en moyenne d'algorithmes
   TD : analyse de coût moyen, analyse du quicksort
6. Algorithmes randomisés
   TD : calcul de la coupe min

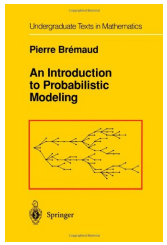
Plan du cours (suite)

Partie II : Espaces continus

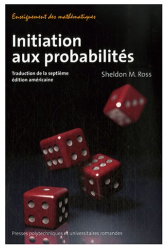
1. Génération selon une loi continue
   TD : génération de vecteurs Gaussiens
2. Convergences : loi des grands nombres, fonction caractéristique
   TD : génération de lois continues arbitraires
3. Analyse de données (statistiques descriptives)
   TD : analyse d'échantillon
4. Estimation statistique et intervalles de confiance
   TD : calcul de taille d'échantillon
5. Décision et tests
   TD : test d'adéquation
6. Chaînes de Markov sur un espace discret
   TD : modèles d'automates probabilistes

Ouvrages de référence

- **Ouvrages de base en probabilité**:

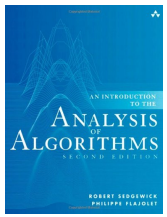


Introduction aux Probabilités. Pierre Brémaud, Springer, 2007.



Initiation aux probabilités Sheldon M. Ross, PPUR 2013.

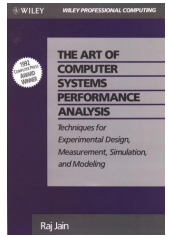
- **Orienté analyse d'algorithmes** :



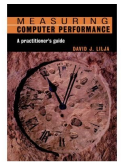
Analysis of algorithms Sedgewick & Flajolet, Addison Wesley 2012

Ouvrages de référence

- **Orienté évaluation de performances** :

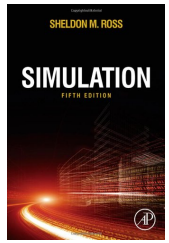


The art of computer system performance analysis, Raj Jain, Wiley 1991



Measuring Computer Performance, Lilja Cambridge Univ Press 2005

- **Simulation à événements discrets** :



Simulation Sheldon M. Ross, Wiley 2012