Les nombres aléatoires

- Il est souvent utile d'avoir une fonction qui produit des nombres apparemment aléatoires

- fonction $rand()$ prédéfinie ou écrite "à la main"

- chaque fois appelée, elle donne un nombre différent (normalement en $[0..N-1]$)

- Pas vraiment aléatoire mais statistiquement

Génération de nombres

- entiers pseudo-aléatoires $x_1, ..., x_i, ...$

- calcul simple comme $x_{i+1} = x_i * m + c \pmod N$ pour m bien choisi et N une puissance de 2

- diviser par N pour un réel aléatoire en [0..1)

- ensuite mutiplier par R pour un réel en $[0..R)$

- et arrondir pour un entier en $[0..R-1]$

- (prendre x_i mod R très mauvais si R une puissance de 2!!)

Génération de structures

- Soit générer un entier dans le bon intervalle et décoder

- Soit générer la structure itérativement avec un entier généré chaque fois que l'algorithme doit prendre une décision

- deuxième méthode plus facile à programmer

- et évite les problèmes d'entiers trop grands

Utilisation 0: interface avec l'extérieur

- programme de jeu: ne pas toujours jouer pareil
- joli écran
- réseau: attendre temps aléatoire avant de réessayer après collision

Utilisation 1: tester un algorithme

- Générer un jeu de tests sur les cas aléatoires

- Ne remplace pas les tests systématiques qui utilisent chaque branche du programme

- "Tester ne prouve jamais qu'un programme est sans faute"

- Sauf pour ceux avec un nombre fini (et petit) de données possibles

Utilisation 2: calcul numérique de statistiques

- Par exemple: superficie moyenne du polygone défini (enveloppe convexe) par 10 points choisis aléatoirement dans le carré $[0,1][0,1]$

- Générer, disons 1000, cas aléatoires et calculer la superficie de chacun

- dangereux: avec très faible probabilité peut être loin du vrai moyen.

- très dangereux: si la distribution est très bizarre (écart type très grand), presque certain d'être loin du vrai moyen.

Utilisation 3: Simulation de processus (naturels ...)

Exemples:

- propagation d'une maladie
- effet sur la circulation de modifications d'une route
- résilience d'un réseau à des pannes
- comportement d'une bourse où des commandes de ventes ou d'achats sont générées par des logiciels

où la complexité des interactions entre les composants d'un système exclut tout calcul direct.

Utilisation 4: algorithmes probabilistes

- algorithmes pour un calcul exact qui utilisent des choix aléatoires

- probabilité d'échec mais, souvent, ils reconnaissent qu'un échec est arrivé

- répéter plusieurs fois (avec choix différents) réduit exponentiellement la probabilité d'échec

Exemple: un algorithme de factorisation

Donné un grand entier n chercher un facteur p dans le cas où n est trop grand pour essayer tous les facteurs possibles jusqu'á racine de n

- un groupe défini selon deux paramètres et où les calculs utilisent l'arithmétique (mod p)

- on calcule une grande puissance d'un élément aléatoire; il y a une bonne probabilité que le résultat soit l'identité

- si on fait l'arithmétique (mod n), le fait d'avoir trouvé l'identité se manifeste par un calcul impossible de x^{-1} où x a un facteur commun avec n

- donc on a trouvé un facteur!

- répéter plusieurs fois avec les paramètres du groupe choisis aléatoirement!

Le choix de probabilité d'échec

- un algorithme classique avec un choix déterministe des aléas marcherait dans presque tous les cas

- mais peut-être nous ne savons pas comment programmer un bon choix

- l'algorithme probabiliste a la même probabilité d'échec à chaque essai que le déterministe

- mais maintenant c'est une très petite probabilité pour chaque n au lieu de certitude d'échec pour un très petit nombre de cas.