Analyse d'Algorithmes et Génération Aléatoire

TD 2 : Génération d'objets combinatoires

1 Retour sur la génération d'entiers

Exercice** 1 **: RAND48

Question 1.1 Implémenter le générateur RAND48 présenté dans le TD 1. On rappelle que l'entier généré est un entier sur 48 bits.

On veut se débrouiller pour ne gaspiller aucun bit des entiers générés.

Question 1.2 Implémenter une fonction initialisant le générateur avec une graine fournie en argument (choisir à chaque fois une graine $> 2^{40}$).

Question 1.3 Implémenter une fonction `bit_suivant` sans argument et renvoyant le bit aléatoire suivant. Dans ce but, la fonction fait appel à RAND48 afin d'obtenir un entier, puis renvoie successivement les 48 bits de l'entier, avant de faire à nouveau appel au générateur.

Question 1.4 Etant donné un entier n. Écrire une fonction `rand_int` générant uniformément un entier entre 0 et n inclus, en utilisant le générateur de bits aléatoire et basée sur la méthode de rejet, si nécessaire.

Question 1.5 Proposer une étude expérimentale pour quantifier le nombre de bits rejetés lorsque l'on appelle la fonction `rand_int` pour quelques valeurs n bien choisies.

Dans la suite, on fera appel ces générateurs uniformes d'entiers et de bits lorsqu'on aura besoin d'aléa.

2 Génération de permutations

Lorsque des études expérimentales sont demandées, appuyer votre réponse sur des courbes ou graphiques.

Exercice** 2 **: Comparaisons de deux algorithmes

Question 2.1 Encoder l'algorithme `gen_permututation_1` présenté dans le cours. On fera particulièrement attention à utiliser les structures de données qui y sont mentionnées.

Question 2.2 Proposer une étude pour quantifier le nombre de bits nécessaires (consommés + rejetés) lorsque l'on appelle la fonction `gen_permututation_1` pour quelques valeurs n bien choisies.

Question 2.3 Encoder l'algorithme `gen_permututation_2` présenté dans le cours. On fera particulièrement attention à utiliser les structures de données qui y sont mentionnées.

Question 2.4 Proposer une étude pour quantifier le nombre de bits nécessaires (consommés + rejetés) lorsque l'on appelle la fonction `gen_permututation_2` pour quelques valeurs n bien choisies.

Question 2.5 Proposer une étude comparative du temps d'exécution des deux algorithmes.

Exercice* 3 *: Début d'étude statistique

Question 3.1 Étant donné n, calculer la distribution de probabilité de l'emplacement de l'entier n dans l'ensemble des permutations de longueur n.

Question 3.2 Générer quelques milliers de permutations de longueur $n = 1000$ en utilisant `gen_permututation_2` et calculer la distribution de l'emplacement de l'élément 1000 dans ces permutations.

Question 3.3 Étant donné n, calculer la probabilité que les trois entiers $1, 2, 3$ apparaissent de façon consécutives dans une permutation de taille n.

Question 3.4 En faisant varier n, générer quelques milliers de permutations de longueur n en utilisant `gen_permututation_2` et vérifier expérimentalement la probabilité d'avoir les entiers $1, 2, 3$ consécutifs dans les permutations.

Question 3.5 Étant donné n, rappeler la distribution des permutations de longueurs n par rapport au nombre de cycles apparaissant dans leur décomposition en cycle. Représenter cette distribution de probabilité lorsque $n = 1000$.

Question 3.6 Écrire une fonction `cycles` qui étant donnée une permutation la décompose en cycle.

Question 3.7 Générer quelques milliers de permutations de longueur $n = 1000$, décomposer les en produit de cycle et confronter vos résultat à la distribution théorique.

3 Générateur d'arbres de Catalan

Cette section a pour but la génération d'arbres binaires via un algorithme de unranking, et via l'algorithme de Rémy avec un souci d'efficacité au niveau de la gestion des bits aléatoires nécessaires. Dans un deuxième temps, nous construirons des arbres aléatoires et analyserons des caractéristiques importantes de cette structure.

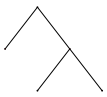
3.1 Générateur d'arbres binaires

Cette section permet de construire uniformément au hasard des arbres binaires.

Exercice* 4 *: Structure d'arbre binaire

Question 4.1 Définir une structure de données permettant d'encoder des arbres binaires. On rappelle qu'un arbre binaire est soit (1) une feuille, soit (2) un nœud interne ayant deux fils qui sont chacun des arbres binaires.

Question 4.2 Définir une fonction `arbre2str` prenant en entrée un arbre binaire, et une chaine de caractères `str` et générant un fichier texte nommé `str.txt` contenant l'encodage sous forme d'expression bien parenthésée de l'arbre binaire. Les indentations, espaces et retour à la ligne au sein du fichier ne sont pas importants pour la suite. Ainsi l'arbre binaire suivant est encodé via l'expression : (() (() ())).



De façon plus formelle. Si φ est l'application qui transforme un arbre en chaine de parenthèses, alors en notant ε une feuille, on a $\varphi(\varepsilon) = ()$ et sinon, si l'arbre, noté A a une racine, en enfant gauche G et un enfant droit D, alors $\varphi(A) = (\varphi(G)\varphi(D))$.

Exercice* 5 *: Génération récursive d'arbres binaires

Question 5.1 Rappeler l'interprétation combinatoire des arbres binaires.

Question 5.2 Si l'on souhaite contruire un arbre de taille n (n nœuds internes), quelle est la probabilité que l'enfant gauche soit de taille k (avec $k \in \{0, \dots, n-1\}$) ?

Question 5.3 Étant donné un entier n fourni par l'utilisateur, écrire une fonction renvoyant un tableau de taille $n+1$ de telle sorte que dans la case i est stockée le i-ème nombre de Catalan, C_i.

Question 5.4 Quelle est la plus grande valeur de n pour laquelle la fonction précédente termine en moins de 1 seconde ?

Question 5.5 Définir une fonction de génération récursive d'arbre binaire.

Exercice* 6 *: Unranking d'arbres binaires

Question 6.1 Expliciter un ordre total sur les arbres binaires de taille n. Pour cela commencer par représenter les premiers arbres de tailles 0 à 4 et indiquer leur rang.

Question 6.2 Définir une fonction de unranking des arbres binaires, prenant les paramètres : n pour la taille et r pour le rang de l'arbre.

Exercice* 7 *: Générateur de Rémy

Question 7.1 Encoder l'algorithme de Rémy. Afin de ne pas avoir à parcourir à chaque étape l'arbre afin de retrouver le nœud choisi aléatoirement, maintenir un tableau associant au nœud i un pointeur vers son parent (attention à la racine de l'arbre).
On définit la taille d'un arbre comme étant son nombre de nœuds internes.

Question 7.2 Générer des arbres de taille 10, 30, 50, 100, 1000 et 10000.

3.2 Caractéristiques d'arbres binaires typiques

Cette section permet d'étudier des mesures d'arbres binaires typiques.

Exercice* 8 *: Mesures

Question 8.1 On souhaite mesurer le nombre exact de bits aléatoires qui ont été nécessaires lors de la génération d'un arbre.

Question 8.2 On souhaite calculer la profondeur d'un arbre, sachant que la profondeur de l'arbre réduit à une feuille est 0 et que si l'arbre n'est pas réduit à une feuille, sa profondeur est le maximum des profondeurs de ses fils à qui on ajoute 1.

Question 8.3 On souhaite calculer la largeur d'un arbre, sachant que la largeur correspond au nombre maximum, sur l'ensemble des niveaux, de nœuds situés sur le même niveau.

Exercice* 9 *: Expérimentations

Question 9.1 Représenter sur différents graphiques les trois mesures effectuées sur différents arbres. Choisir judicieusement les tailles des arbres.

Question 9.2 Enrichir les graphiques des moyennes des mesures.

Question 9.3 Prendre en main l'outil graphviz (il existe des versions en ligne `http://www.webgraphviz.com/`, par exemple) basé sur des fichiers .dot et afficher les arbres générés.

4 Arbres Binaires de Recherche

Cette section a pour but l'étude de la profondeur moyenne des feuilles dans les Arbres Binaires de Recherche (ABR). Un ABR est une structure arborescente telle que chaque nœud interne contient une clé. Pour un nœud interne contenant la clé r, son sous-arbre gauche ne contient que des clés plus petites que r et son sous-arbre droit ne possède que des clés plus grandes que r.

4.1 Généralités

Un ABR est une structure dynamique : on y insère au fur et à mesure des clés. La taille de l'arbre est son nombre de nœuds internes, i.e. le nombre de clés qu'il contient.

— L'arbre de départ est l'arbre ne contenant qu'une feuille, donc de taille 0.

— À l'étape i, on souhaite insérer la clé r_i dans l'arbre déjà construit et contenant $i-1$ clés. En comparant r_i aux nœuds internes rencontrés depuis la racine, on descend soit dans le fils gauche, soit dans le fils droit. Dès qu'on tombe sur une feuille, on remplace cette feuille par un nœud interne étiqueté par r_i et dont les fils gauche et droit sont chacun réduit à une feuille.

Exercice* 10 *: Ensemble de clés

On peut montrer que toute suite de n clés distinctes pour lesquelles il existe un ordre total (toutes sont comparables) peut s'identifier à une suite sur les entiers de 1 à n. Donc, on n'utilisera que les suites des entiers de 1 à n en tant que clés.

Question 10.1 Combien existe-t-il de suites de longueur n contenant les n entiers de 1 à n ?

Exercice* 11 *: Ordre d'insertion

Question 11.1 Construire l'ABR obtenu après insertion des clés $1, 2, 3, 4, 5$.

Question 11.2 Construire l'ABR obtenu après insertion des clés $4, 2, 3, 1, 5$.

Question 11.3 Construire l'ABR obtenu après insertion des clés $4, 2, 3, 5, 1$.

Question 11.4 Quelle conclusion importante tirer des trois questions précédentes ?

Exercice* 12 *: Arbres de Catalan

On rappelle qu'un arbre de Catalan est un arbre binaire (non étiqueté). Le nombre d'arbres de Catalan de taille n (nombre de nœuds internes) est $\frac{1}{n+1}\binom{2n}{n}$.

Question 12.1 Prouver que tout arbre de Catalan correspond à la structure arborescente d'au moins un ABR.

Question 12.2 Prouver que deux ABR distincts ont nécessairement une structure arborescente différente.

Question 12.3 Que conclure des deux dernières questions ?

4.2 Catalan aléatoire

Cette section permet de définir des modèles d'ABR aléatoire.

Exercice* 13 *: Arbres de Catalan uniformes – ABR uniformes

Question 13.1 En choisissant la distribution uniforme de probabilité sur les arbres de Catalan (i.e. sur les ABR), comment générer un ABR uniformément au hasard ?

Question 13.2 Quelle est la profondeur moyenne d'une feuille dans les arbres de Catalans uniformes (i.e. dans les ABR munis de la distribution uniforme) ?

On rappelle que les ABR sont utilisés en raison d'une propriété importante sur la profondeur moyenne des feuilles.

Question 13.3 Cette profondeur est-elle celle donnée dans la question précédente ?

Exercice* 14 *: ABR non uniforme

L'idée est la suivante, on souhaiterait générer l'ABR encodant une suite aléatoire choisie uniformément parmi toutes les suites de n entiers.

Question 14.1 Prouver que la distribution induite sur les ABR par la distribution uniforme sur les suites de n entiers n'est pas la distribution uniforme.

Question 14.2 Pour chacune des suites de 3 entiers, donner l'ABR qui lui correspond.

Question 14.3 Si l'on choisit uniformément un des 5 ABR (de taille 3), quelle est la profondeur moyenne d'une feuille ?

Question 14.4 Si l'on choisit uniformément une suite de 3 entiers, quelle est la profondeur moyenne d'une feuille ?

Question 14.5 Que représente la profondeur moyenne d'une feuille dans le contexte des ABR ?

4.3 Expérimentations

On souhaite générer des ABR, suivant la distribution induite par la distribution uniforme sur les suites de n entiers. Toutefois, on ne veut pas générer une suite uniformément puis construire son ABR. On veut directement générer l'ABR relatif à une suite (qui serait choisie uniformément).

Exercice* 15 *: Génération d'ABR

Question 15.1 Définir une structure de données permettant d'encoder des ABR.

Pour générer un ABR de taille n, on procède en deux étapes. On génère sa structure arborescente (suivant une distribution adéquate !), puis on l'étiquette.
Afin de générer la structure de taille n, on répète n fois l'étape suivante (on part de l'arbre de taille 0). À l'étape i (pour i de 1 à n), une feuille est choisie uniformément parmi les i feuilles. Puis on la remplace par un nœud interne sans étiquette dont chacun des deux fils est une feuille.

Question 15.2 Coder un générateur de structure d'ABR, en prenant en argument la taille n.

Question 15.3 Donner une fonction prenant une structure d'ABR et l'étiquetant.

Question 15.4 Quelle est la complexité de l'algorithme global ?

Exercice* 16 *: Profondeur moyenne des feuilles

Question 16.1 Représenter des résultats expérimentaux concernant la profondeur moyenne des feuilles dans les ABR. Choisir les tailles des ABR et le nombre d'ABR générés afin que les résultats semblent fiables.

Question 16.2 La profondeur asymptotique des feuilles d'un ABR est en $\Theta(\log n)$. Ceci se confirme-t-il sur les expériences ?

Question 16.3 Déterminer expérimentalement la constante multiplicative cachée dans l'expression $\Theta(\log n)$.

Exercice* 17 *: Justifications du générateur

Question 17.1 Prouver que le générateur donné est correct, c'est-à-dire qu'il génère les ABR suivant la distribution induite par la distribution uniforme sur les permutations de 1 à n.