Génération de structures combinatoires

Génération de terrains aléatoires

L'objectif est de donner des algorithmes qui génèrent un terrain de n cases ayant k cases occupées par des obstacles.

Génération aléatoire uniforme de terrains comportant exactement k obstacles

Méthode 1 : Sachant qu'il y a exactement C_n^k terrains, on génère uniformément un entier dans l'intervalle $[1, C_n^k]$ et on génère la configuration correspondant au numéro généré,

Méthode 2 : On génère les k premiers éléments d'une permutation aléatoire de $\{1, \cdots, n\}$.

Méthode 2bis : On génère successivement et uniformément les obstacles sur l'ensemble des cases libres.

Méthode 3 : Tant que le nombre d'obstacles générés n'est pas égal à k, on choisit uniformément une position sur $[1, ..., n]$ et si la position est déjà occupée, on recommence.

Méthode 4 : On place successivement un obstacle sur la case i avec la probabilité $\frac{k-k_i}{n-i+1}$, où k_i est le nombre d'obstacles déjà placés sur les cases $\{1, \cdots, i-1\}$.

Génération aléatoire de terrains comportant une densité moyenne $d = \frac{k}{n}$ d'obstacles

Méthode 5 : De manière indépendante on génère un obstacle sur une case avec la probabilité d.

Méthode 5bis : Même méthode que que la précédente, mais avec rejet si la densité obtenue est hors d'un intervalle $[d - \epsilon, d + \epsilon]$.

Question 1.1 : Analyse et algorithmes

Quel est le nombre de configurations de terrains possible ?

Ecrire l'algorithme associé à chaque méthode et l'exécuter pour générer 5 terrains de 5 cases et 3 obstacles). On utilisera la fiche des tirages aléatoires donnée précédemment).

Question 1.2 : Preuve et complexité

Prouver que chaque méthode génère un terrain et que la génération est uniforme (sauf pour 5 et 5 bis).

Pour chaque méthode donner la distribution du coût de génération.

Générateur d'arbres aléatoires

On dispose d'un générateur `alea(int n)` qui génère un entier uniformement distribué sur $\{0, \cdots, n\}$.

D'autre part on dispose d'un type `arbre` binaire et de deux constructeurs d'arbres binaires :

- `arbre_vide()` crée un arbre vide ;
- `assemble_arbre(arbre x; arbre y)` prend en argument les arbres `x` et `y` et renvoie un arbre binaire dont le sous-arbre gauche (resp. droit) est `x` (resp. `y`).

Soit la fonction suivante :

```
arbre creer_arbre(int n);
arbre a1,a2;
si (n==0) alors
  return arbre_vide();
sinon
  q=alea(n-1);
  a1=creer_arbre(q);
  a2=creer_arbre(n-q-1);
  return assemble_arbre(a1,a2);
fin si
```

Question 2.1 : Analyse

Montrer qu'un appel à `creer_arbre(n)` génère un arbre ayant n sommets. Montrer que tout arbre A de n sommets peut être généré par `creer_arbre(n)` (la probabilité de générer A est strictement positive).

Question 2.2 : Preuve

Le générateur est-il de loi uniforme ?

Question 2.3 : Chemin à gauche

Soit X_n le nombre de sommets sur le chemin le plus à gauche dans l'arbre généré ayant n sommets. Calculer la loi de X_0, X_1, X_2 et X_3. vérifier que $\mathbb{E}X_3 = 1 + \frac{1}{3}(\mathbb{E}X_0 + \mathbb{E}X_1 + \mathbb{E}X_2)$.

Montrer que

$$\mathbb{E}X_n = 1 + \frac{1}{n}\sum_{i=0}^{n-1} \mathbb{E}X_i.$$

En déduire une expression de $\mathbb{E}X_n$ en fonction de $\mathbb{E}X_{n-1}$. Calculer $\mathbb{E}X_n$ et donner son équivalent pour n grand.

Question 2.4 : Conclusion

A quel algorithme cette construction vous fait-elle penser ?

Générateur uniforme aléatoires

On dispose du générateur `RANDOM` classique qui fournit un nombre réel dans $[0, 1[$.

Question 2.5 : Nombre d'arbres

Donner les valeurs de C_n, pour $n = 0,\ 1,\ 2,\ 3,\ 4$. Donner une relation de récurrence sur le nombre C_n d'arbres de taille n.

Question 2.6 : Algorithme

Proposer un algorithme de génération d'arbres distribués uniformement. Prouver votre algorithme.

Question 2.7 : Calcul de C_n

Donner une formule explicite pour C_n. On pourra utiliser la série génératrice associée à la suite C_n