L3-INFO Algorithmique et Modélisation

Quick – 26 février 2021 – durée 1 h

Sont interdits : les documents, les ordinateurs, les téléphones (incluant smartphone, tablettes,... tout ce qui contient un dispositif électronique).
Seuls les dictionnaires papier pour les personnes de langue étrangère sont autorisés.
Il sera tenu compte de la qualité de la rédaction et de la clarté de la présentation (2 pts).
Le barème indicatif : Exercice 1 : 5 pts ($\sim 20mn$) ; Exercice 2 : 13 pts ($\sim 45mn$).

1 Tableau circulairement trié

Un tableau T de taille n d'éléments comparables est dit *circulairement trié* par ordre croissant si il existe un indice i_{min} avec $1 \leqslant i_{min} \leqslant n$ tel que

$$T[i_{min}] \leqslant T[i_{min}+1] \leqslant T[i_{min}+2] \leqslant \cdots \leqslant T[n] \leqslant T[1] \leqslant T[2] \leqslant T[i_{min}-1].$$

Par exemple, pour le tableau suivant de taille $n = 10$

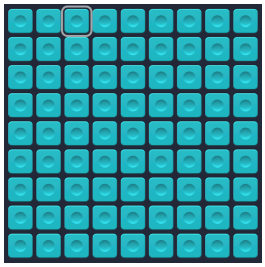
1	2	3	4	5	6	7	8	9	10
23	42	99	128	1024	2	3	10	13	19

l'indice i_{min} correspondant est 6.

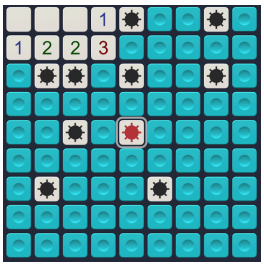
1. Écrire un algorithme naïf en $\mathcal{O}(n)$ opérations de comparaisons d'éléments, qui renvoie l'indice i_{min} d'un tableau de n éléments **distincts** trié circulairement.
2. Écrire un algorithme qui résout le même problème mais avec une complexité en $\mathcal{O}(\log n)$. On justifiera précisément cet algorithme (preuve et analyse de la complexité).
3. Démontrer que cet algorithme est optimal en nombre de comparaisons d'éléments.

2 Démineur, des bombes dans une grille

Dans le jeu du démineur, l'ordinateur propose au joueur une grille de $N = n \times n$ cases, k cases, cachées initialement au joueur contiennent des bombes. Celui-ci doit découvrir les bombes en tentant sa chance sur différentes cases...



(a) Grille initiale



(b) Partie perdue

(c) Positions des mines

FIGURE 1: Grille $N = 9 \times 9$ avec $k = 10$ mines

Pour initier le jeu l'ordinateur doit générer une grille $N = n \times n$ avec exactement k bombes cachées. Pour que le joueur humain ne soit pas avantagé (que la grille soit imprédictible) on souhaite construire une grille aléatoire et que toutes les grilles possibles aient la même probabilité d'apparition.

On dispose d'une fonction `random()` qui génère un nombre réel dans l'intervalle $[0, 1[$, on suppose que les appels successifs à `random()` sont modélisés par une suite de variables aléatoires indépendantes de même loi uniforme sur $[0, 1[$.

1. En une phrase que fait la fonction `piece` définie ci-dessous :

```
Fonction piece (p)
  Données : p un nombre réel 0 ⩽ p ⩽ 1
  Résultat : un entier (booléen) ...

  if random () < p
   └ Renvoie 1
  else
   └ Renvoie 0
```

2. Démontrer que la fonction `piece` fait bien ce que l'on attend d'elle.

On remarque que la génération d'une grille à $N = n \times n$ cases revient à numéroter toutes les cases de 1 à N et à construire un tableau de N bits, chaque bit indiquant s'il y a une bombe sur la case ou pas, et ayant exactement k bits à 1. On propose l'algorithme :

```
Fonction genereA (N, k)
  Données : N, k entiers, 0 ⩽ k ⩽ N
  Résultat : un tableau de N bits indicé de 1 à N

  p = k/N
  G = tableau de N bits
  for n = 1 to N
   └ G[i] = piece (p)
  Renvoie G
```

3. Que pensez-vous de ce générateur de grille `genereA` ? Justifier votre réponse.

À partir de la fonction `random()` on construit la fonction `alea(n)` suivante :

```
Fonction alea (N)
  Données : N un entier
  Résultat : un entier k ...

  k = floor ( N * random () ) +1 /* floor (x) est la partie entière inférieure
   du nombre réel x, floor (3,14) renvoie l'entier 3  */
  Renvoie k
```

4. En une phrase, que fait la fonction `alea` ?
5. Démontrer que la fonction `alea`, fait bien ce que vous venez de spécifier.

On utilise l'algorithme suivant pour générer une grille

```
Fonction genereB (N, k)
  Données : N, k entiers, 0 ⩽ k ⩽ N
  Résultat : un tableau de N bits ...

  bomb = 0
  G = tableau de N bits initialisé à 0
  while bomb < k
  |  i = alea (N)
  |  if G[i] ≠ 1
  |  |  G[i] = 1
  |  |  bomb = bomb + 1
  Renvoie G
```

6. Expliquer en une phrase le principe de cet algorithme (ne pas démontrer sa correction).
7. Donner une expression de son coût moyen (on pourra s'inspirer du problème du "Coupon collector" pour obtenir au moins un majorant).
8. Que se passe-t'il lorsque k est petit ? lorsque k est proche de N ? Proposer une optimisation de cet algorithme.
9. Dans le jeu du démineur, combien de grilles différentes sont-elles possibles en fonction de k et N ?
10. On souhaite enregistrer au fur et à mesure les différentes grilles générées et ne proposer que de nouvelles grilles. Proposer une structure de donnée permettant de stocker les grilles et de vérifier rapidement qu'une configuration a déjà été utilisée. Justifier votre réponse.