Domaine Sciences et Technologies
Master 1 Informatique

Aix*Marseille université

Aspects probabilistes pour l'informatique : TD 4
Code UE: SINB19AL

Année 2020-2021

Algorithmes randomisés

Exercice 1 On tire les cartes une apres l'autre parmi les 52 cartes et chaque fois on essaie de deviner leur valeur. Utilisez des variables indicatrices pour calculer le nombre esperée des cartes correctement devines si

(a) on ne se souvient pas des cartes déjà sorties ;
(b) on se souviens des cartes déjà sorties.

Exercice 2 (vestiaire à chapeaux) Utilisez des variables indicatrices pour résoudre le problème suivant, connu sous le nom de problème du *vestiaire à chapeaux*. Chaque client parmi n au total donne son chapeau à un employé d'un restaurant. Cet employé redonne les chapeaux aux clients dans un ordre aléatoire. Quel est le nombre attendu de clients qui récupéreront leurs chapeaux ?

Exercice 3 (coupon collecteur) Un collectionneur cherche à avoir toutes les vignettes d'une série de n vignettes distribuées aléatoirement dans des boîtes de céréales. Combien faut-il acheter de boîtes de céréales en moyenne pour avoir la collection complète ? *Indication : On pourra chercher à estimer l'espérance de la variable aléatoire T_i décrivant le nombre d'achats nécessaires pour obtenir la i-ème nouvelle vignette, en utilisant le résultat du cours concernant le nombre moyen de tirages d'une expérience de Bernoulli avant le premier succès, c'est-à-dire l'espérance d'une variable suivant une loi géométrique. On utilisera également les nombres harmoniques $H_n = \sum_{i=1}^{n} 1/i = \ln n + \gamma + o(1)$ avec $\gamma \approx 0,577$ la constante d'Euler-Mascheroni.*

Exercice 4 (test de deux polynômes) Considérons un algorithme pour tester si deux polynômes P et Q de meme degré d sont identiques. Par exemple, on cherche à savoir si $P = (X+1)(X-2)(X+3)(X-4)(X+5)(X-6)$ est egal à $Q = X^6 - 7X^3 + 25$. L'algorithme choisit un entier aleatoire r dans l'intervalle $\{1, \ldots, 100d\}$ et calcule $P(r)$ et $Q(r)$. Si $P(r) \neq Q(r)$, alors l'algorithme décide que les deux polynômes ne sont pas identiques. Si $P(r) = Q(r)$, alors l'algorithme décide que les deux polynômes sont identiques.

(a) Quelle est la probabilité que l'algorithme retourne une mauvaise réponse (c'est-à-dire qu'il décide que P et Q sont identiques, alors même qu'ils ne le sont en fait pas) ? (*Indication : Utiliser le fait qu'un polynôme de degré d possède au plus d racines.*) Comment faire pour diminuer la probabilité d'échec ?
(b) Supposons qu'on répète le tirage k fois (sans prendre soin de ne pas tester plusieurs fois la même valeur test r) et on retourne « P et Q sont identiques » si et seulement si $P(r_i) = Q(r_i)$ pour $i = 1, \ldots, k$. Quelle est la probabilité de retourner une mauvaise réponse ?
(c) Que devient cette probabilité si on prend soin de ne pas tester plusieurs fois la même valeur test r ?

Exercice 5 (tri rapide) Montrer que le nombre espéré de comparaisons du l'algorithme de tri rapide sur un tableau de n éléments est de $O(n \log n)$. *Indication : Soit $b_1 < b_2 < \cdots < b_n$ les éléments du tableau, dans l'ordre croissant. On considère une distribution aléatoire uniforme sur l'ensemble des permutations possibles de ces n éléments. Soit X_{ij} la variable aleatoire définie par $X_{ij} = 1$ si les éléments b_i et b_j sont comparés par le tri rapide dans le tableau correspondant, et $X_{ij} = 0$ sinon. Calculer $\mathbf{E}(X_{ij})$. Noter que si $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$, alors $\mathbf{E}(X)$ est le résultat souhaité, puis conclure.*

Exercice 6 (permutation aléatoire) Plusieurs algorithmes randomisés utilise le tirage aléatoire d'une permutation d'un tableau A de taille n. Le but d'un tel tirage est (1) d'obtenir chaque permutation de facon uniforme (avec la probabilité $\frac{1}{n!}$) et (2) que cette permutation est obtenue rapidement. Considerons la procédure Randomisation-Directe, connue aussi sous le nom de

Knuth shuffle. Dans l'itération i, l'élément $A[i]$ est choisi au hasard parmi les éléments $A[i]$ à $A[n]$.

```
Randomisation-Directe(A)
n := longueur[A] ;
pour i = 1 à n
      faire échanger A[i] ↔ A[Random(i, n)]
```

1. Quelle est la complexité de l'algorithme RANDOMISATION-DIRECTE ?
2. Montrer que chaque permutation a la probabilité $\frac{1}{n!}$ d'être retourné par l'algorithme.
   Indication : Etablir l'invariant de boucle suivant : juste avant la i-ème iteration de la boucle **pour**, *pour chaque $(i-1)$-permutation possible, le sous-tableau $A[1..i-1]$ contient cette $(i-1)$-permutation avec la probabilité $(n-i+1)!/n!$.*