TD 3 : Analyses amortie et probabiliste

Exercice 1. *File-sur-piles*

Avec un tableau dynamique, on obtient directement une structure de données de pile dont les deux opérations EMPILER et DÉPILER ont une complexité amortie constante et dont l'accès au $i$ème élément de la pile se fait en temps constant, quelquesoit i. Le but de l'exercice est d'implémenter une structure de données similaire pour avoir ces mêmes avantages pour une file.

1. Pourquoi les tableaux dynamiques ne suffisent pas *a priori* ?

Pour implanter la file, on utilise deux piles E (entrée) et S (sortie). Pour ENFILER un élément, on l'empile sur E. Pour défiler, on dépile depuis S. Si S est vide et qu'on doit DÉFILER, on commence par transférer tous les éléments de E dans S.

2. Écrire les deux algorithmes ENFILER et DÉFILER et analyser leurs complexités dans le pire des cas. *On suppose avoir accès aux opérations de base sur les piles.*

3. On va démontrer que le coût amorti par opérations est constant, avec les différentes méthodes.

1. Utiliser la *méthode de l'agrégat* : montrer que si on part d'une file vide et qu'on effectue n opérations ENFILER/DÉFILER, il y aura au plus $3n$ opérations EMPILER/DÉPILER effectuées. Pour cela, on pourra majorer le nombre d'opérations effectués pour les éléments qui sont dépilés et pour ceux qui ne sont jamais dépiler.
2. Utiliser la *méthode comptable* : au moment d'ENFILER un élément x, on paie une somme de 3, et on paie 1 pour DÉFILER. Montrer que le compte reste bien positif, et que chaque opération a un coût amorti constant. Pour cela, on peut montrer que le compte vaut à tout moment $2.e$ où e est le nombre d'éléments dans la pile E.
3. Utiliser la *méthode du potentiel*, en définissant le potentiel de la file (E,S) comme étant deux fois la taille de E.

Exercice 2. *Compteur avec incrément et décrément*

On cherche à maintenir un compteur qui peut alterner les incréments et les décréments. *La mesure de coût reste la même que dans le cours : chaque inversion de la valeur d'un bit coûte* 1.

1. Montrer qu'avec un compteur standard, il est possible que le coût amorti ne soit pas constant.

Pour retrouver un coût *amorti* constant par opération, on représente la valeur v du compteur par un couple de deux entiers (P,N) tel que $v = P - N$. Parmi les représentations de v, on appelle *représentation exclusive* un couple (P,N) qui vérifie $P \wedge N = 0$, où $\wedge$ représente un ET logique, bit-à-bit. *Par exemple,* $18 \wedge 9 = 0$ *car* $18 = 10010_2$ *et* $9 = 01001_2$*, donc* $(18,9)$ *est une représentation exclusive de la valeur* 9*. Par contre,* $20 \wedge 13 = 4$ *donc* $(20,13)$ *n'est pas une représentation exclusive de* 7*.*

2. Donner une représentation exclusive de 7, puis de 8, a chaque fois avec $N \neq 0$.

3. Soit (P,N) un couple d'entiers quelconque et $v = P - N$.

1. Supposons que les $k^{\text{èmes}}$ bits $P_{[k]}$ et $N_{[k]}$ de P et N sont tous deux à 1. Montrer qu'en les passant à 0, on obtient un nouveau couple (P',N') tel que $P' - N' = v$.
2. En déduire que tout entier v possède une représentation exclusive (P,N).
3. Montrer sur un exemple que la représentation exclusive d'un nombre n'est pas forcément unique.

4. L'idée du compteur (P,N) est d'incrémenter P pour incrémenter le compteur, et incrémenter N pour le décrémenter, tout en conservant l'invariant $P \wedge N = 0$.

1. Soit (P,N) un représentation exclusive de v. Expliquer comment obtenir, à partir du couple $(P+1,N)$, une représentation exclusive (P',N') de $v+1$.
2. Combien d'inversions de bits sont nécessaires pour passer de $(P+1,N)$ à (P',N'). ?
3. Écrire les deux algorithmes INCRÉMENT et DÉCRÉMENT qui prennent en entrée une représentation exclusive (P,N) d'un entier v, et renvoient une représentation exclusive (P',N') de $v+1$ et $v-1$, respectivement.

5. Quel est le coût individuel maximal d'une opération d'incrément ou de décrément ?

6. On cherche maintenant à faire une analyse *amortie* de ce compteur, avec la méthode du potentiel. Soit $v_0 = 0$ la valeur initiale du compteur, représentée par le couple $(P_0,N_0) = (0,0)$. On effectue t opérations INCRÉMENT ou DÉCRÉMENT successives. On note v_i la $i^{\text{ème}}$ valeur prise par le compteur, et (P_i,N_i) le couple correspondant. On note c_i le nombre de bits inversés lors de la $i^{\text{ème}}$ opération et on définit le potentiel Φ_i comme étant le nombre de bits à 1 dans P_i plus le nombre de bits à 1 dans N_i.

1. Rappeler l'expression du *coût amorti* a_i de la $i^{\text{ème}}$ opération, en fonction de c_i, Φ_i et Φ_{i-1}.
2. Montrer que $a_i \leq 2$.
3. Conclure.

Exercice 3. QUICKSELECT *particulier*

1. On s'intéresse à l'espérance E_n du nombre de comparaisons effectuées par QUICKSELECT$(T, 1)$ où T est un tableau de n éléments.

1. Montrer que $E_n = (n-1) + \frac{1}{n}\sum_{m=1}^{n-1} E_m$.
2. En déduire que $E_n \leq 2n - 2$.

2. De même, étudier l'espérance du nombre de comparaisons effectuées par QUICKSELECT$(T, 2)$.

3. Donner des algorithmes déterministes (simples !) pour ces deux cas, et comparer les complexités obtenues.

Exercice 4. MAXSAT *probabiliste*

Le problème MAXSAT est une variante du problème SAT. Dans cette variante, on se donne un ensemble de clauses $C_1, \ldots, C_m$, chacune contenant des littéraux sur les variables $x_1, \ldots, x_n$. Le problème MAXSAT consiste alors à trouver une affectation des variables $x_1, \ldots, x_n$ à `vrai/faux` qui maximise le nombre de clauses satisfaitent.

1. Proposer un algorithme déterministe exhaustif pour répondre au problème. Quelle est sa complexité ?

2. On souhaite maintenant écrire un algorithme probabiliste MAXSATPROBA pour le problème MAXSAT. Cet algorithme consiste à tirer une valeur `vrai` ou `faux` au hasard pour chaque variable et à retourner l'ensemble des clauses satisfaites par cette assignation.

1. On note k_i le nombre de littéraux de la clause C_i et X_i la variable aléatoire valant 1 si C_i est satisfaite et 0 sinon. Montrer que $\mathrm{E}[X_i] = 1 - \frac{1}{2^{k_i}}$.
2. En déduire que le nombre de clauses satisfaite par MAXSATPROBA en moyenne est supérieur ou égal à $m/2$.
3. On suppose maintenant en plus que chaque clause $C_1, \ldots, C_m$ contient au moins p littéraux pour une valeur $p \geq 2$. Montrer alors que l'algorithme MAXSATPROBA satisfait au moins $m.(1 - \frac{1}{2^p})$ clauses en moyenne.

3. On souhaite maintenant une version MAXSATDERAND de MAXSATPROBA qui n'utilise plus de tirage aléatoire mais assure toujours le même ratio de clauses satisfaites. Lorsque cela est possible, on dit qu'on obtient une version *dérandomisée* de l'algorithme probabiliste considéré.

1. Expliquer comment, algorithmiquement, choisir la valeur de x_1 pour être sûr de satisfaire au moins la moitié des clauses contenant x_1 comme variable.
2. Écrire alors l'algorithme MAXSATDERAND demandé, calculer sa complexité et prouver qu'il satisfait bien au moins la moitié des clauses.