TD 3 : Analyses amortie et probabiliste

Exercice 1. *File dynamique*

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 tableaux dynamiques E (entrée) et S (sortie). Pour ENFILER un élément, on l'ajoute en fin de E. Pour défiler, on enlève depuis la fin de S. Si S est vide et qu'on doit DÉFILER, on commence par transférer tous les éléments de E dans S, dans l'ordre inverse.

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 tableaux dynamiques :* AJOUT *et* SUPPRESSION.

3. Écrire un algorithme qui renvoie le $i$ème élément de la file.

4. 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 AJOUT/SUPRESSION effectuées sur les tableaux dynamiques E et S. 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 4, et on ne paie rien pour DÉFILER. Informellement, argumenter qu'un élément qui sera mis dans la file fera l'objet d'au plus 4 opérations élementaires. Conclure que le solde du compte est toujours positif et que le coût amorti est donc constant.
3. Utiliser la *méthode du potentiel*, en définissant le potentiel de la file (E,S) comme étant deux fois la taille de E pour montrer que le coût amorti est de 3 opérations sur E ou S par élément inséré dans la file.

Exercice 2. *Compteur avec incrément et décrément*

Le but de l'exercice est de construire un compteur qui peut alterner les incréments et les décréments avec un coût amorti constant et une occupation mémoire raisonnable. *La mesure de coût ici est la même que dans le cours : chaque inversion de la valeur d'un bit coûte* 1.

1. Montrer qu'avec un compteur standard, le coût amorti dépend de l'ordre dans lequel on effectue les incréments et décréments.

Pour retrouver un coût *amorti* constant par opération, on représente la valeur v du compteur, *que l'on suppose tout le temps positive*, par un couple de deux entiers positifs (P,N) tel que $v = P - N$ avec un compteur (standard) pour chacun des entiers N et P. Pour incrémenter v, on incrémente P et pour décrémenter v, on incrémente alors N.

2. On considère un tel compteur double pour v.

1. Argumenter rapidement que le coût amorti de `Incrément` et `Décrément` est alors constant lors d'une suite d'opérations sur v.
2. Justifier que l'espace mémoire occupé par P et N dépend du nombre d'opérations effectuées. Vous pourrez regarder ce qu'il se passe si on effectue successivement m fois les deux opérations `Incrément` puis `Décrément` en partant de $v = 0$?

Pour éviter ce problème, on va considérer des représentations de v particulières. On appelle *représentation exclusive* de v un couple (P,N) qui vérifie $v = P - N$ et $P \wedge N = 0$, 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*.*

3. Pour s'entraîner sur les représentations exclusives :

1. Donner une représentation exclusive de 7, puis de 8, a chaque fois avec $N \neq 0$.
2. Montrer sur un exemple que la représentation exclusive d'un nombre n'est pas forcément unique. Donner même pour tout entier $v > 0$ une infinité de représentation exclusive.

Un *compteur exclusif* (P,N) partant de $v = 0$ gère deux compteurs P et N tels que $v = P - N$, pour incrémenter v, on incrémente P pour décrémenter v, on incrémente N, et tout en conservant l'invariant $P \wedge N = 0$.

4. On considère un compteur (P,N) pour v.

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 partant de la un représentation exclusive (P,N) de v, expliquer comment obtenir une représentation exclusive (P',N') de $v+1$ à partir de $(P+1,N)$.
3. Combien d'inversions de bits sont nécessaires pour passer de $(P+1,N)$ à (P',N'). ?
4. 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. Le but de la question est de montrer que P et N sont bornés en fonction de la plus grande valeur prise par v. Concrètement, on considère une séquence d'appels à INCRÉMENT et DÉCRÉMENT en partant de $v=0$. On note $v_{\max}$ la plus grande valeur atteinte par v au cours de cette séquence d'appels et par $i_{\max}$ le plus grand indice d'un 1 dans l'écriture binaire de $v_{\max}$.

1. Monter qu'après un appel à INCRÉMENT, P ne peut pas atteindre la valeur $2^{i_{\max}+2}-1$, qui s'écrit en binaire par une suite de 1 de longueur $i_{\max}+2$.
2. En déduire que $P<4v_{\max}-1$ puis que $N<4v_{\max}-1$ tout au long de la séquence d'appels.
3. Remarquer que le problème évoqué question **3.** ne peut pas se produire avec un compteur exclusif.

6. Pour finir, on cherche 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 $a_i\leq 2$ pour tout $i=1,\dots,t$.
3. Conclure.

Exercice 3. QUICKSELECT *particulier*

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

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 A_n, l'espérance du nombre de comparaisons effectuées par QUICKSELECT$(T,2)$ et montrer que $A_n\leq 2n-3$.

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 $C_1,\dots,C_m$, chacune contenant des littéraux sur les variables $x_1,\dots,x_n$. Le problème MAXSAT consiste alors à trouver une affectation des variables $x_1,\dots,x_n$ 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 $C_1,\dots,C_m$ 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.