Esiee-Paris - unité d'algorithmique - Feuille d'exercices numéro 8

Décembre 2023 – R. Natowicz, I. Alame, A. Çela, X. Hilaire, T. Wu, W. Xu

Exercice 14. Intersection de deux ensembles Les tableaux $A[0:m]$ et $B[0:n]$ sont tous deux <u>strictement</u> croissants. On demande d'écrire une fonction `int[] inter(int[] A, int[] B)` qui retourne un tableau strictement croissant contenant l'intersection des valeurs de A et B.

La première méthode à laquelle on pense consiste à chercher chacune des valeurs de A dans le tableau B par une recherche séquentielle. La recherche d'une valeur de A est en $\Theta(n)$, la recherche de toutes les valeurs de A est en $\Theta(m \times n)$.

La seconde méthode à laquelle on peut penser consiste à appliquer une recherche dichotomique plutôt que séquentielle, de chaque valeur de A dans B. La recherche d'une valeur de A est en $\Theta(\log_2 n)$, la recherche de toutes les valeurs est en $\Theta(m \log_2 n)$.

On demande de faire ce calcul en $\Theta(m+n)$.

Bien voir l'importance du gain de temps de calcul. Avec $m = n = 10^3$, ce programme sera environ 500 fois plus rapide que la première version car $\frac{10^3 \times 10^3}{2 \times 10^3} = 500$, et environ 5 fois plus rapide que la seconde car $\frac{1000 \times \log_2(1000)}{2 \times 1000} = 4.983 \approx 5$.

On donne l'invariant de l'algorithme :

$$I(k,p,q): \qquad A \cap B = C[0:k] \cup (A[p:m] \cap B[q:n])$$

L'intersection $A \cap B$ que nous voulons calculer se décompose en deux parties :

1. ce qui a déjà été calculé. Ce résultat partiel est le k-préfixe de C ;
2. ce qui reste à calculer. C'est l'intersection des p et q suffixes de A et B.

La propriété $I(k,p,q)$ nous dit : " l'intersection de A et B est l'union du k préfixe de C et de l'intersection des p et q suffixes de A et B. "

Condition d'arrêt : il faut avoir $A \cap B = C[0:k]$. Donc $A[p:m] \cap B[q:n] = \emptyset$, autrement dit $p = m$ ou $q = n$.

Initialisation : rien n'a encore été calculé. Donc $C[0:k] = \emptyset$, autrement dit $k = 0$. Nous avons donc $A \cap B = A[p:m] \cap B[q:n]$. Autrement dit $p = 0$ et $q = 0$. L'initialisation est donc $k = 0$ et $p = 0$ et $q = 0$.

Progression : à vous de jouer. Trois cas peuvent se présenter : $a_p < b_q,\ a_p > b_q,\ a_p = b_q$.

Votre fonction `int[] inter(int[] A, int[] B)` calculera l'intersection $A \cap B$ dans le tableau $C[0:m]$ de taille $m = \min\{m,n\}$ car $|A \cap B| \le \min(m,n)$. À la fin du calcul vous aurez $A \cap B = C[0:k]$. Votre fonction retournera le k-préfixe du tableau C par l'instruction `return Arrays.copyOfRange(C,0,k)`.

Exercice 15. Premier plus long sous-tableau constant. On demande de calculer les indices d et f du premier plus long sous-tableau constant (pplstc) du tableau $T[0:n]$ et de retourner ces deux indices. Exemple : le pplstc de $T[0:n] = [1,2,1,2,2,2,3,3,3,2]$ est $T[d:f] = T[3:6] = [2,2,2]$.

On donne l'invariant de la fonction à écrire :

$I(d,f,j,k)$: " $T[d:f]$ est le pplstc du k-préfixe de T et $T[j:k]$ est son plus long suffixe constant. "

Écrire la fonction `int[] pplstc(int[] T)` qui fait ce calcul et retourne les indices d et f. La fonction sera commentée par l'invariant (initialisation, condition d'arrêt, progression.)

Exercice 16. Premier sous-tableau de somme maximum. Le tableau d'entiers $T[0:n]$ contient des valeurs entières quelconques : positives, négatives, nulles. Écrire une fonction `int[] pstsm(int[] T)` qui calcule les indices d et f de début et de fin du premier sous-tableau non vide de somme maximum ainsi que la somme des valeurs de ce sous-tableau. Ces trois valeurs seront retournées dans un tableau de trois cases : `return new int[] {d,f,s}`. On demande un calcul en $\Theta(n)$.

On donne l'invariant :

$I(d,f,s,j,k,s')$: le sous-tableau $T[d:f]$, de somme s, est le pstsm ~~pplstc~~ du k-préfixe de T, et le j-suffixe du k-préfixe, de somme s', est le suffixe de somme maximum.

On recommande de faire un petit dessin.

Écrire la fonction `int[] pstsm(int[] T)` commentée par l'invariant : init$^{\text{on}}$, cond$^{\text{on}}$ d'arrêt, progression.