Chapitre 7

Compromis espace-temps

Compromis espace-temps

- Il arrive souvent qu'il soit possible de diminuer le temps d'exécution d'un algorithme en utilisant de l'espace mémoire pour stocker de l'information additionnelle qui accélère le traitement des instances

  - Ex1: pour calculer la valeur d'une fonction f(x) pour une instance arbitraire x, nous pouvons stocker dans un tableau f[0..n-1] la valeur de f pour plusieurs points x_i pour $i \in \{0..n-1\}$.
    - La valeur de f(x) pour un x arbitraire est ensuite obtenue rapidement par interpolation des valeurs stockées dans le tableau

  - Ex2: le tri par dénombrement que nous étudierons dans ce chapitre

  - Ex3: les algorithmes de programmation dynamique que nous étudierons au prochain chapitre

Le tri par dénombrement

- Lorsque chaque élément A[i] d'un tableau A[0..n-1] est un entier satisfaisant:

$$l \leq A[i] \leq u \quad \forall\, i \in \{0..n-1\}$$

- L'idée principale du tri par dénombrement consiste à stocker la **fréquence** de chaque valeur entière possible dans un tableau de fréquences F[0..u-l]:
  - F[0] contient le nombre de fois que la valeur l est présente dans A
  - F[1] = nombre de fois que la valeur l+1 est présente dans A
  - F[j] = nombre de fois que la valeur l+j est présente dans A
  - …
  - F[u-l] = nombre de fois que la valeur u est présente dans A

Le tri par dénombrement (suite)

- Lorsque nous avons ce tableau F[0..u-l] de fréquences:
  - La valeur l occupera les F[0] premières positions dans le tableau trié
  - La valeur l+1 occupera les F[1] positions suivantes dans le tableau trié
  - La valeur l+2 occupera les F[2] positions suivantes dans le tableau trié
  - …
- Il suffit donc d'écrire, dans un nouveau tableau S[0..n-1] les valeurs dans cet ordre

- Nous utilisons un nouveau tableau S pour contenir les éléments triés car chaque élément s'accompagne habituellement de données satellites.

Le tri par dénombrement (suite)

- Pour obtenir **directement la position** de stockage de chaque élément, il est préférable de créer un tableau D[0..u-l] de **distribution** :
  - D[0] = F[0]
  - D[1] = F[1] + D[0]
  - D[2] = F[2] + D[1]
  - …
  - D[j] = F[j] + D[j-1] pour j 2 {1..u-l}

- Alors: D[j] = nombre d'éléments de A ayant une valeur ≤ l+j

- Par exemple, supposons que A[0..5] = 13, 11, 12, 13, 12, 12
  - Donc nous avons l = 11 et u = 13
  - Alors: F[0..2] = 1, 3, 2   (pour les valeurs 11, 12, 13 resp.)
  - Et: D[0..2] = 1, 4, 6   (pour les valeurs 11, 12, 13 resp.)

Le tri par dénombrement (suite)

- Ayant le tableau de distribution D, nous pouvons écrire S comme suit:
  - S[i] = l pour i ∈ {0..D[0]-1}   (car D[0] = nombre d'élm. ≤ l + 0)
  - S[i] = l+1 pour i ∈ {D[0]..D[1]-1} (car D[1] = nombre d'élm. ≤ l + 1)
  - …
  - S[i] = l+j pour i ∈ {D[j-1]..D[j]-1} (car D[j] = nombre d'élm. ≤ l + j)
  - … jusqu'à j = u – l
- L'algorithme débute avec le dernier élément A[n-1].
  - Nous avons A[n-1] = l + j (pour un certain j ∈ {0..u-l})
- On écrit alors A[n-1] en position D[j]-1 dans S: S[D[j]-1] ← A[n-1]
  - Ensuite on fait: D[j] ← D[j] – 1 pour obtenir la position d'écriture pour la prochaine valeur de l + j rencontrée dans A
- On recommence ainsi pour A[n-2] jusqu'à A[0]

Algorithme du tri par dénombrement

Algorithme 1 : TriParDenombrement($A[0..n-1]$)

// Tri un tableau d'entiers;

Entrées : Un tableau d'entiers $A[0..n-1]$ tel que $l \leq A[i] \leq u$ pour tout $0 \leq i < n$.

Sorties : Un tableau $S[0..n-1]$ contenant les éléments de A triés en ordre non-décroissant.

pour $j = 0$ **à** $u - l$ **faire** $D[j] \leftarrow 0$; Initialise le tableau de fréquences

pour $i = 0$ **à** $n - 1$ **faire** $D[A[i] - l] \leftarrow D[A[i] - l] + 1$; Calcule le tableau de fréquences

pour $j = 1$ **à** $u - l$ **faire** $D[j] \leftarrow D[j-1] + D[j]$; Calcule le tableau de distribution

pour $i = n - 1$ **à** 0 **faire**

- $j \leftarrow A[i] - l$;
- $D[j] \leftarrow D[j] - 1$;
- $S[D[j]] \leftarrow A[i]$;

retourner S

Exemple d’exécution du tri par dénombrement

	D[0..2]			*S*[0..5]					
A [5] = 12	1	**4**	6				12		
A [4] = 12	1	**3**	6			12			
A [3] = 13	1	2	**6**						13
A [2] = 12	1	**2**	5		12				
A [1] = 11	**1**	1	5	11					
A [0] = 13	0	1	**5**					13	

FIGURE 7.2 Example of sorting by distribution counting. The distribution values being decremented are shown in bold.

Analyse du tri par dénombrement

- Pour déterminer le temps d'exécution, comptons le nombre d'affectations effectuées
- L'algorithme effectue $\Theta(u\text{-}l)$ affectations pour initialiser D à zéro
- Il effectue ensuite $\Theta(n)$ affectations pour calculer les fréquences
- Il effectue ensuite $\Theta(u\text{-}l)$ affectations pour obtenir le tableau de distribution à l'aide des fréquences
- Il effectue finalement $\Theta(n)$ affectations pour obtenir le tableau S trié à partir du tableau initial A et du tableau D
- Le temps d'exécution C(n) est donc donné par (en pire et meilleur cas):

$$\mathbf{C(n) = \Theta(\max\{u\text{-}l, n\})}$$

- C'est donc un algorithme de tri **linéaire en n**
- Mais ce n'est pas un algorithme de tri par comparaisons…

Lecture (Levitin)

- Chapitre 7 Space and Time Trade-Offs
  - 7.1 Sorting by Counting