Complexité algorithmique, TD00

Stéphane Le Roux, leroux@lsv.fr

November 23, 2022

Question 1 : Comparaison de deux entiers

Trouver des algorithmes le plus simples possibles, en terme de clareté, parmi les algorithmes en temps polynomial qui répondent aux questions suivantes. Préciser leurs complexités en temps et en espace.

1. - Entrée : deux entiers n et k représentés en base 2 séparés par ♯.
   - Question : est-ce que $n = k$?
2. - Entrée : deux entiers n et k représentés en base 2 séparés par ♯.
   - Question : est-ce que $n \leq k$?

Solution 1

Techniquement, il faut savoir si les nombres sont représentés avec bit de poids forts à gauche ou à droite. On suppose que c'est à gauche. On accepte aussi les entrées qui commencent par des 0. Si on se rend compte à un moment que l'entrée n'a pas la forme attendue, on rejette.

1. - On parcourt l'entrée jusqu'au ♯ en la recopiant sur une bande de travail.
   - Sur la bande d'entrée on se place sur le premier 1 après le ♯. Sur l'autre bande sur le premier 1.
   - On effectue alors une comparaison bit à bit.

   Cette méthode est linéaire en la taille de l'entrée, en temps et en espace.
2. - On parcourt l'entrée jusqu'au ♯ en la recopiant sur une bande de travail.
   - On se place aux dernières cases des bandes d'entrée et de travail.
   - On effectue alors une comparaison bit à bit, en gardant en mémoire sur une bande supplémentaire si $n_i \leq p_i$, où n_i et k_i sont les nombres entiers correspondant aux lectures partielles déjà effectuées à l'étape i. Cette information est mise à jour facilement à la lecture de deux nouveaux bits.

   Cette methode est linéaire en la taille de l'entrée, en temps et en espace.

Question 2 : Compteurs

1. Entrée : un entier naturel n. Tâche : le contenu d'une bande doit prendre toutes les valeurs de 0 à n dans l'ordre croissant et passer dans un état spécial après chaque changement.

2. Entrée : deux entiers naturels n, m. Tâche : les contenus de deux bandes doivent prendre toutes les valeurs de 0 à n et 0 à m, dans l'ordre croissant, et passer dans un état spécial pour chaque nouveau couple de valeurs.

Discuter de l'efficatité de vos algorithmes.

Solution 2

1. - Ajouter 1 à un nombre k écrit en base 2 est linéaire en la représentation du nombre.
   - À partir du nombre 0, on ajoute 1, on compare avec l'entrée en temps linéaire comme dans la question précédente, on ajoute 1, etc, et on s'arrête quand l'égalité est atteinte. On passe n fois dans cette boucle.

   Cet algorithme termine en temps $o(n \log n)$ (voire $O(n)$ si on regarde les détails) avec espace en $O(\log n)$, i.e. linéaire en la taille de l'entrée.

2. - On recopie m, trouvé après le $\sharp$, sur une bande de travail.
   - On effectue l'algorithme pour un seul entier avec n, mais à chaque fois que la MT passe dans son état spécial, l'algo est exécuté complètement pour m avec une seconde MT sur des bandes disjointes, etc.

   Cet algorithme termine en temps $O((n \log n)(m \log m))$ (estimation grossière) et espace $O(\log n + \log m)$.

Question 3 : Comparaison d'un bit de deux entiers

Trouver un algorithmique en temps linéaire et en espace logarithmique répondant à la question suivante.

- Entrée : trois entiers n, k, p.
- Question : les p-ième bits (en partant des poids faibles) de n et k sont-ils égaux ?

Solution 3

- L'entrée est de la forme $n\sharp k\sharp p$.
- Si $p > \log_2 n$ (et/ou $p > \log_2 k$), on rejette. Pour cela, on compte en base 2 la longueur de la représentation de n sur une bande de travail, puis on compare p à cette longueur grâce à un algorithme précédent. Cela se fait en temps et espace $O(\log n)$.
- On recopie p sur une bande de travail.

- Grâce à un compteur comme à la question 2, on place la tête de lecture de la bande d'entrée sur le p-ième bit de n : à partir du $\sharp$, on se déplace de une case vers la gauche en incrémentant le compteur, etc.
- On retient le bit en question (sur une autre bande de travail ou grâce aux états de la MT), puis on va au p-ième bit de k et on compare.

Question 4 : Comparaison de deux entiers en espace logarithmique

Trouver des algorithmes en espace logarithmique pour la Question 1.

Solution 4

1.
   - On ne copie aucun des deux entiers sur une bande de travail, on va les comparer sur place, sur la bande d'entrée.
   - Sur une bande de travail on écrit en binaire la longueur de l'entrée qu'on appelle l.
   - On lance un compteur p de 0 à l. C'est en espace $\log l$ (et temps linéaire en l).
   - À chaque nouvelle valeur de p, on compare les p-ième bits de n et k. C'est en espace $\log p$ (et temps linéaire en l)
   - L'ensemble est en espace $\log l$ et temps l^2.
2. Il suffit à nouveau de garder en mémoire (1 bit) si $n_i \leq k_i$ (n et k partiels)

Question 5 : Maximum d'une liste

Trouver un algorithme en espace logarithmique pour la tâche suivante.

- Entrée : une liste d'entiers $n_1 \sharp n_2 \sharp \dots \sharp n_k$, où k n'est pas connu à l'avance.
- Sortie : $\max_{i \in [k]} n_i$.

Solution 5

- Sur deux bandes de travail, on maintient deux compteurs. Un compteur i pour le n_i qu'on est en train de lire et un compteur $j \in [i]$ pour l'indice du dernier maximum partiel.
- Quand on lit un nouveau n_i, on le compare au maximum courant grâce à l'algorithme précédent.

Comme k est inférieur à la longueur de l'entrée, cela se fait en espace logarithmique.

Question 6 : Addition, multiplication, exponentiation

Trouver des algorithmes relativement efficaces pour les opérations suivantes.

1. - Entrée : un entier $n > 0$.
   - Sortie : $\lceil \log_2(n) \rceil$
2. - Entrée : deux entiers n et k.
   - Sortie : $n + k$
3. - Entrée : deux entiers n et k.
   - Sortie : $n \cdot k$
4. Soit $k \in \mathbb{N} \setminus \{0\}$.
   - Entrée : un entier n.
   - Sortie : n^k.
5. - Entrée : deux entiers n et k.
   - Sortie : n^k.

Solution 6

1. On calcule la longueur de l'entrée sans compter les 0 de tête. On fait ça avec un compteur, donc en espace logarithmique.
2. Méthode classique d'addition avec propagation de retenue, en temps linéaire.
   - Si on peut afficher les bits de poids faibles d'abord, pas besoin de bande de travail.
   - Sinon, c'est en espace linéaire.
3. Méthode classique de multiplication.
   - Sur deux bandes de travail T_1 et T_2, on écrit n.
   - Sur la bande d'entrée, on parcours k en commençant par les bits de poids faible.
   - À chaque étape, on multiplie par deux le contenu de T_1, en insérant un 0.
   - Si le bit courant de k est 1, on ajoute T_1 à T_2.
   - À la fin, on copie T_2 sur la bande de sortie.

   Cela se fait en espace linéaire et temps quadratique en la taille de l'entrée.
4. On effectue un nombre constant, $k - 1$, de multiplications. Chacune d'entre elle prend un nombre quadratique en la taille de l'entrée (c'est presque la question précédente).
5. La sortie est de taille exponentielle en la taille de l'entrée. On peut optimiser le calcul, mais la méthode restera en temps exponentiel.

Question 7: Évaluation de polynomes

Considérons un anneau tel que multiplication et addition soient réalisables en temps polynomial. Soit $P[X_1, \ldots, X_v]$ un polynôme. Montrer que la tâche suivante est aussi réalisable en temps polynomial.

- Entrée : des éléments $x_1, \ldots, x_v$ de l'anneau.
- Sortie : $P(x_1, \ldots, x_v)$.

Solution 7

- Chaque $x_1^{\alpha_1} \cdot \cdots \cdot x_v^{\alpha_v}$ se calcule en temps polynomial, car il y un nombre constant $k := (\sum \alpha_i) - 1$ de multiplications, dont les opérandes sont de au plus longueur linéaire en la taille de l'entrée.
- Il suffit ensuite d'additioner les résultats intermédaires, par exemple via un accumulateur. Le nombre de ces additions est aussi constant, car le polynôme est fixé.

Question 8: Addition et multiplication de matrices

On considère des matrices à coefficients dans $\mathbb{Z}$.

1. - Entrée : deux matrices M et P.
   - Sortie : $M + P$ (si les matrices sont de même taille)
2. - Entrée : deux matrices M et P.
   - Sortie : $M \cdot P$ (si les matrices le permettent)

Solution 8

Conventions : le premier bit d'un entier code son signe ; les coefficients sont séparées par un ♯. On passe à la ligne suivante avec ♯♯. On sépare deux matrices par ♯♯♯.

1. - On vérifie que les entrées sont deux matrices en stockant (en base 2) sur une bande de travail T_1 le nombre de coefficients sur la première ligne. On vérifie ligne à ligne que les autres lignes ont la même longueur : soit en parcourant la ligne et en comparant à un compteur lié à T_1, soit en écrivant la longueur de la ligne courante sur une bande T_2 et en comparant les deux entiers sur T_1 et T_2. Cela se fait en temps linéaire et espace logarithmique.
   - On vérifie que les matrices ont la même taille, en temps linéaire et espace logarithmique. Plus précisément,
     - on vérifie que la première ligne des deux matrices ont la même taille.
     - on vérifie que les matrices ont le même nombre de lignes, en comptant les ♯♯.

- On calcule ensuite le nombre de coefficients dans la première matrice (ou la deuxième). Puis on écrit la somme des deux matrices directement sur la bande de sortie, en calculant la somme case à case. Cela se fait en temps quadratique ou cubique (à cause de va-et-vient) et espace linéaire.

2.
- On vérifie comme précédemment que les deux entrées sont des matrices.
- On vérifie que M a le même nombre de colonnes que P de ligne.
- On calcule le nombre l de lignes de M et le nombre c de colonne de P.
- On lance deux compteurs imbriqués $i \in [l]$ et $j \in [c]$. Pour chaque (i, j) on parcourt la ligne i de M et la colonne j de P, en "parallèle" grâce à un compteur k pour effectuer des multiplications, puis addition dans un acculmulateur. Un quatrième compteur peut servir à faire des va-et-vient entre les matrices.

Cela est en temps polynomial, peut-être de degré 4, et en espace linéaire.