ALGORITHMES ET COMPLEXITE - MINI PROJET

PRESENTE PAR

NDEYE SOKHNA SECK

1. Algorithme de la machine 1

Dans la machine 1 décrite dans le fichier '***Machine1.txt***', deux bandes infinies sont utilisées.

La première contient la longueur n suivi du séparateur 'r' puis du mot.

L'algorithme choisi consiste à recopier la longueur n sur la seconde bande. Et lorsqu'on lit le séparateur 'r', initialiser la recherche du nième bit. A ce stade, le pointeur se trouve sur la première lettre du mot sur la première bande, et sur le dernier bit de la longueur sur la seconde. La longueur étant donnée en unaire, le mot et la longueur sont parsés simultanément (en se déplacement vers la gauche pour la seconde bande et vers la droite pour la première à chaque itération).

Le résultat est la lettre lue sur la bande 2.

Si long(n) < long(m), nous arrivons sur un blanc dans la seconde bande lors d'une lecture d'une lettre sur la première. Cette lettre est recopiée sur la seconde et est le résultat.

Si long(n)>long(m), nous arrivons sur un blanc dans la première bande lors d'une lecture d'un '1' sur la seconde. On écrit 'X' sur cette dernière, ce qui constitue le résultat.

Si long(n)==long(m), nous lisons simultanément un blanc sur la première et sur la seconde bande. Nous retournons donc à gauche sur la première bande pour lire la dernière lettre, que nous écrivons sur la seconde bande, constituant ainsi le résultat.

2. Algorithme de la machine 2

➔ Description de l'algorithme

Dans cette machine, une bande infinie est utilisée. Elle contient l'expression '***NrW***' avec N, la longueur du mot en unaire, suivi du séparateur 'r' et enfin le mot W.

L'algorithme implémenté consiste à parser chaque caractère de la longueur (en unaire), le remplacer par du blanc puis de vérifier s'il reste une lettre non lue. Si oui, on la remplace par un blanc et reboucle. Cette lettre a donc été comptabilisée.

- Au point de départ de chaque boucle, le pointeur se trouve sur la première case non blanche contenant un caractère (‘1’) de n, longueur du mot. Il est remplacé par du blanc avant de se déplacer vers la droite.

  Si la case suivante est blanche, cela signifie que la longueur n est atteinte : la lettre sur le bit numéro n du mot est celle dans la case après le séparateur. Il s’agit du cas long(n)<=long(m). Sinon, on itère à droite jusqu’à atteindre un blanc ou le séparateur.

- Lorsqu’on atteint le séparateur, il est remplacé par du blanc et sur la prochaine case lue, on replace le séparateur.

  (Vérification sur la case suivante)

  - Si cette case (celle sur laquelle on place le séparateur) contenait une lettre, on vérifie si la prochaine case contient toujours une lettre. Dans ce cas, la recherche n’est pas terminée, on retourne vers le point de départ . Autrement, on trouve une case blanche et il s’agit du cas long(n)>long(m). En effet, si nous sommes à cette case, cela signifie qu’il reste toujours des caractères de la longueur n à parser. On peut donc déjà s’assurer que la longueur n est supérieur à celle du mot. La case blanche est remplacée par ‘X’, ce qui constitue le résultat.
  - Si cette case était vide (exemple mot vide), la prochaine est remplacée par ‘X’, le résultat.

Le résultat est la lettre lue sur la bande (suite au séparateur ‘r’).

➔ Analyse de la complexité

A chaque boucle :

- On évalue le nombre de caractères (‘1’) non lues (non blanches) de longueur n => C1
- Puis on prend en compte les cases qui ont été qui ont été remplacées par du blanc jusqu’au séparateur => C2
- Ensuite, on comptabilise le séparateur (+1), puis la vérification sur la prochaine case (+1). Dans le pire des cas, ces deux cases contiennent des lettres; la recherche n’est donc pas terminée. Retour vers le séparateur (+1) => C3 = 3

- Enfin nous retournons vers le point de départ. => C4 = C2+C1.

Ainsi C= 2(C1+C2)+3. '***n***' étant donné en unaire, long(n) = n.

Dans le pire des cas :

Lors de la première boucle B1, on a: C1 = n; C2 = 0 => CB1 = 2((n-0)+0)+3 = 2n+3

A la seconde boucle B2, on a: C1 = n-1; C2 = 1 => CB2 = 2((n-1)+1)+3 = 2n +3

A la troisième, B3, on a: C1 = n-2; C2 = 2 => CB3 = 2((n-2)+2)+3 = 2n +3

....

A la kième, Bk, on a C1 = n-k+1; C2 = k-1 => CBk = 2((n-k+1)+k-1)+3 = 2n+3

D'où le nombre d'itération dans une boucle pour un mot de longueur n est de C(n) = 2n+3. Cette opération est répétée p fois; p = min(long(m), long(n)) = min(long(m),n).

En effet, si m<=n, la recherche se termine quand le mot est intégralement lue. Sachant que chaque lettre est parsée lors d'une boucle, p = long(m).

Si m>n, la recherche se termine quand la longueur **'n'** est intégralement lue. De même, p=long(n)=n.

Ainsi, la complexité de l'algorithme est de p(2n+3) < N(2N+3); N=n+long(m)+1.

Nous en concluons que la complexité de cet algorithme est quadratique. **$C_A = O(N^2)$.**

3. Algorithme de la machine 3

➔ Description de l'algorithme

Dans cette machine, la longueur de n est donnée en binaire. Afin d'améliorer la complexité de la machine, on se limite à effectuer un nombre d'opération inférieur à cste*log(N), pour atteindre une complexité globale égale à O(NlogN) sur l'ensemble de l'algorithme.

Pour se faire, à chaque boucle, un bit de la longueur n est diminué à chaque passage. Il s'agit du dernier bit lu. Lorsque le tout dernier bit de cette longueur n (avant le séparateur est lu), la lettre la plus proche du séparateur (à droite) est la lettre recherchée.

A chaque passage, on vérifie s'il s'agit du dernier bit, dans quel cas, on cherche la prochaine lettre après le séparateur, ce qui constitue notre résultat. Il s'agit du cas où long(n)<=long(m). Sinon, on évalue ce dernier bit. S'il est égal à 0, la lettre recherchée est dans une case d'indice paire. On simplifie donc une lettre sur deux juste après le séparateur. S'il est égal à 1, la case la contenant est d'indice paire. On simplifie d'abord la première lettre après le séparateur avant de simplifier une lettre sur deux. Puis on retourne vers le point de départ. Ainsi, il ne reste plus que la moitié de la longueur de n à chaque passage. Lors de ce retour, dans le cas où long(n)>long(m), on peut voir qu'on ne parse que du caractères simplifiés ('S') jusqu'au séparateur. Le résultat est donc X.

➔ Analyse de la complexité

Il s'agit bien de la méthode utilisée dans la recherche dichotomique. Ceci nous garantit une complexité logarithmique lors de la lecture de longueur du mot; ce qui détermine le nombre de passage.

Pour p, le nombre de passages effectués; p est égal à log(n) si long(n)<long(m) car dès que le dernier bit de la longueur est reconnu, l'analyse prend fin. Dans le cas où long(n)>long(m), on sait que long(n) = log(n). D'où p <= log(n). Le nombre d'opération est donc bien bien inférieur à cste*log(N).

Lors de chaque passage, le mot bien que simplifié est entièrement lu. On parse donc l'ensemble toute la bande, en aller et en retour => C = 2 (long(m) + long(n) + 1)

Sachant que long(n) = log(n) on a C = 2(m + log(n) + 1) < 2 (m + n + 1) <2N

Pour p passages, on a donc C <= 2Nlog(n); avec log(n)<log(M);

D'où une complexité de **Nlog(N) : C_A = O(Nlog(N))**.

4. Algorithme de la machine 4

➔ Description de l'algorithme

Dans cette machine, la longueur de n est donnée en unaire. Afin de conserver la complexité obtenue précédemment, on s'applique à effectuer un nombre d'opération égale. Pour cela, lors de la lecture du mot de longueur n, on simplifie un caractère '1' sur deux. Si le dernier '1' avant le séparateur n'est pas simplifié, la case contenant la lettre recherchée est d'indice paire. Autrement, elle est d'indice impaire. Les mêmes simplifications sont faites que précédemment; le même raisonnement est tenu.

➔ Analyse de la complexité

Par analogie, la longueur de n est également divisée par 2. Ce qui détermine tout aussi pareillement le nombre de passages effectués sur la bande : elle est logarithmique.

A chaque passage, toute la bande est lue en aller/retour d'où une complexité de 2N pour chaque passage.

La complexité de cette machine est de **$C_A = O(N\log(N))$**.