Projet d'Informatique Parallèle et Distribuée

E. Goubault & M. Martel

2003

Le but de ce projet est de réaliser une application distribuée, en utilisant JavaParty, afin de rechercher rapidement et de manière approchée une séquence d'ADN particulière dans une base de données contenant un grand nombre de séquences. Plus précisément, nous souhaitons permettre aux "utilisateurs" de trouver la séquence de la base ressemblant le plus à un motif fourni en entrée. La ressemblance entre deux séquences est donnée par la *distance d'édition* dont le calcul est décrit à la section 1. Les sources de parallélisme sont nombreuses : on pourra s'intéresser à la parallélisation du calcul de la distance d'édition entre deux chaînes de caractères, à la comparaison d'un même motif à plusieurs séquences de la base en parallèle, ainsi qu'au traitement simultané de plusieurs requêtes (c.à.d. la comparaison de plusieurs motifs aux éléments de la base en parallèle). De plus, ces différentes possibilités peuvent être exploitées simultanément, mais, dans ce cas, se posent des problèmes de partage des ressources...

1 Distance d'édition

La distance d'édition est une technique couramment employée pour mesurer le niveau de ressemblance entre deux chaînes de caractères u et v. Le principe consiste à modifier la première séquence u à l'aide de certaines opérations jusqu'à ce que l'on obtienne la seconde séquence v. Un coût est donné à chacune des opérations de la transformation et la somme des coûts indique la distance séparant les deux séquences. Il existe généralement plusieurs solutions et la distance correspond au coût de la plus petite d'entre elles. Les opérations autorisées sont :

- la *substitution* : une lettre est remplacée par une autre afin de faire correspondre u et v,
- la *suppression* : une lettre de u est supprimée,
- l'*insertion* : on ajoute une lettre à l'endroit souhaité dans la chaîne u.

Des coûts différents peuvent être donnés à chacune des transformations. Appelons c la fonction de coût : $c(x, y)$ indique le coût du remplacement du caractère x par le caractère y. Les coûts des suppressions et insertions d'un caractère x seront indiqués par $c(x, \varepsilon)$ et $c(\varepsilon, x)$.

Soient u_i et v_i les *i-ièmes* caractères des chaînes u et v et soit $\mathcal{D}(i, j)$ la distance entre les séquences $u_0 u_1 \dots u_i$ et $v_0 v_1 \dots v_j$. Si l'on suppose connues les distances $\mathcal{D}(i-1, j)$, $\mathcal{D}(i, j-1)$ et $\mathcal{D}(i-1, j-1)$, il existe trois manières permettant de transformer $u_0 \dots u_i$ en $v_0 \dots v_j$:

- soit on transforme $u_0 \dots u_{i-1}$ en $v_0 \dots v_j$ puis on supprime u_i,
- soit on transforme $u_0 \dots u_i$ en $v_0 \dots v_{j-1}$ puis on insère le suffixe v_j,
- soit on transforme $u_0 \dots u_{i-1}$ en $v_0 \dots v_{j-1}$ puis on substitue v_j à u_i.

Ainsi, le coût de la transformation de $u_0 \dots u_i$ en $v_0 \dots v_j$ sera le minimum des coûts des trois transformations énumérées ci-dessus, ce qui nous amène à la récursion :

$$\mathcal{D}(i,j) = \begin{cases} \min \begin{cases} \mathcal{D}(i-1,j-1) + c(u_i, v_j) \\ \mathcal{D}(i-1,j) + c(u_i, \varepsilon) \\ \mathcal{D}(i,j-1) + c(\varepsilon, v_j) \end{cases} & si\ i > 0\ et\ j > 0 \\ 0 \quad si\ i = 0\ et\ j = 0 \\ \mathcal{D}(i,j-1) + c(\varepsilon, v_j) \quad si\ i = 0\ et\ j > 0 \\ \mathcal{D}(i-1,j) + c(u_i, \varepsilon) \quad si\ i > 0\ et\ j = 0 \end{cases} \tag{1}$$

Les cas $i = 0$ ou $j = 0$ traitent les bords du tableau. Dans le cas $i = j = 0$, les mots considérés sont donc ε et ε. La distance les séparant est toujours considérée nulle : D(0,0) = 0. Pour les cas où seul i ou j est nul, le résultat est lié au fait que la seule manière de transformer ε en $u_0 \dots u_i$ (resp. $v_0 \dots v_j$ en ε) est d'insérer (resp. supprimer) toutes les lettres requises. Un exemple de calcul est donné à la figure 1.

	ε	C	T	A	C	C	G	A	C	G	A
ε	0	1	2	3	4	5	6	7	8	9	10
T	1	1	1	2	3	4	5	6	7	8	9
A	2	2	2	1	2	3	4	5	6	7	8
C	3	2	3	2	1	2	3	4	5	6	7
C	4	3	3	3	2	1	2	3	4	5	6
G	5	4	4	4	3	2	1	2	3	4	5
C	6	5	5	5	4	3	2	2	2	3	4
G	7	6	6	6	5	4	3	3	3	2	3
T	8	7	6	7	6	5	4	4	4	3	2
A	9	8	7	6	7	6	5	4	5	4	3

Figure 1: Exemple de calcul de distance d'édition.

2 Parallélisation

Les questions suivantes sont destinées à vous aider à paralléliser l'algorithme de calcul de la distance d'édition.

1. Déduire de la formule (1) un programme de calcul de $\mathcal{D}(n,n)$ sous forme d'une imbrication de boucles `for`.

2. Tracer dans le plan (O,i,j) les dépendances entre les calculs des différents $\mathcal{D}(i,j)$. Proposer un ordonnancement pour calculer ces valeurs en parallèle, c.à.d. une fonction (linéaire) $\theta(i,j)$ qui associe à chaque point $\mathcal{D}(i,j)$ sa date de calcul ($\theta(i,j) = (n+1)i + j$ est probablement une solution triviale mais pas très parallèle! Trouvez mieux!). Quelles sont les valeurs de $\mathcal{D}$ calculées aux instants 0, 1, n, etc. ?

3. En déduire une nouvelle écriture du programme de la forme:

```
for t = 0 to ...
  for k = ... to ...
    D[...t...,...k...] = ....
```

4. Ecrire un algorithme pour PRAM permettant de calculer la distance d'édition.
5. Ecrire un algorithme calculant en parallèle sur p processeurs la distance d'édition entre des chaînes de longueur n, avec $p << n$.

Pour tester vos programmes, il est possible de trouver des exemples de séquences par exemple aux URL suivantes:

`http://www.genoscope.cns.fr/`
`http://flybase.bio.indiana.edu/`