Programmation Dynamique: TD

Tronc Commun Scientifique
Recherche Opérationnelle
École des Mines de Nancy

1 La distance d'édition

…aussi appelée *distance de Levenshtein*.

Soient x et y deux mots sur un alphabet, de longueurs respectives m et n.

On note $d(x, y)$ le nombre minimal d'insertions ou de suppressions de caractères pour passer de x à y. L'entier $d(x, y)$ est appelé *distance d'édition* entre les mots x et y.

Par exemple, on peut passer de `mines` à `mimes` des deux manières suivantes :

- `mines` → `mies` (suppression) → `mimes` (insertion)
- `mines` → `mins` (suppression) → `mimns` (insertion) → `mimens` (insertion) → `mimes` (suppression)

La première solution nécessite 2 insertions/suppressions, la seconde 4.
Dans cet exemple, $d($`mines`,`mimes`$) = 2$.

Questions.

1. Montrez que d établit bien une distance entre mots.
2. Montrez que $|m - n| \leqslant d(x, y) \leqslant m + n$.
3. Pour $i \in \{0, 1, 2, \ldots, m\}$ et $j \in \{0, 1, 2, \ldots, n\}$, on note x_i et y_j les préfixes de x et y de longueurs respectives i et j, avec la convention $x_0 = y_0 = \varepsilon$, où ε est le mot vide.

   Quelles sont les valeurs de $d(x_i, y_0)$ et $d(x_0, y_j)$?
4. Soient $i \in \{1, \ldots, m\}$ et $j \in \{1, \ldots, n\}$. Montrez que :

$$d(x_i, y_j) = \min\big\{d(x_{i-1}, y_{j-1}) + 2\,\delta_{i,j},\ d(x_{i-1}, y_j) + 1,\ d(x_i, y_{j-1}) + 1\big\}, \quad (1)$$

   où $\delta_{i,j}$ vaut 0 si la i-ème lettre de x et la j-ème lettre de y sont identiques, et 1 sinon.
5. Estimez la complexité de l'algorithme récursif implémentant « directement » l'équation (1). On pourra raisonner sur deux mots de même longueur n.

   Remarquez que dans cet algorithme les mêmes calculs sont faits plusieurs fois.

6. Déduisez de l'équation (1) un algorithme de calcul de $d(x, y)$ tel que le nombre d'opérations et l'occupation mémoire soient en $\mathcal{O}(mn)$.
7. Calculez $d(\texttt{ingenieur}, \texttt{igneneur})$.
8. Trouvez une suite d'opération réalisant le nombre minimal d'opérations dans le cas précédent.

Remarque : On définit aussi la distance d'édition comme le nombre minimal d'insertions, suppressions ou *substitutions* (ces dernières sont bien sûr possibles dans le cadre de l'énoncé, mais « coûtent » ici 2 opérations). On peut aussi donner des coûts différents à chaque opération. Le principe de la résolution par programmation dynamique reste le même.

La distance d'édition est utilisée dans de nombreux contextes (par exemple les correcteurs d'orthographe des logiciels de traitement de texte). Une variante est également utilisée dans les problèmes d'*alignement de génomes* en biologie moléculaire.

2 Problème de location de skis

On cherche à attribuer m paires de skis (de longueurs $l_1, l_2, \ldots, l_m$) à n skieurs (de tailles $t_1, t_2, \ldots, t_n$), de manière à minimiser la somme des différences entre la longueur des skis et la taille des skieurs.

Bien sûr, $n \leqslant m$.

On suppose sans perte de généralité que les l_i et les t_i sont classés par ordre croissant.

Le problème consiste à trouver une fonction (d'allocation) a telle que le skieur n° i se voit attribué la paire de skis n° $a(i)$ et qui minimise :

$$\sum_{i=1}^{n} \left| t_i - l_{a(i)} \right|$$

La même paire de skis n'est pas attribuée à deux skieurs, donc a est injective.

Questions.

1. Montrez que l'optimum est atteint par une fonction a croissante. On supposera donc que l'affectation des skis des i premiers skieurs se fait parmi les j premières paires de skis.
2. Soit $S_{i,j}$ le minimum de la fonction-objectif du problème restreint aux i premiers skieurs et j premières paires de skis.
   Montrez que
   $$S(i, j) = \min\big\{S(i, j-1), S(i-1, j-1) + |t_i - l_j|\big\}.$$
3. Comment procéderiez-vous pour calculer la solution optimale ?
4. Quelle est la complexité de cet algorithme ?

3 Équilibrage de charge sur deux machines en parallèle

On considère deux machines (par exemple des machines industrielles, mais aussi des processeurs) sur lesquelles sont exécutées n tâches de durées $t_1, t_2, \ldots t_n$.

On souhaite répartir les tâches sur chacune des machines de manière à ce qu'elles s'arrêtent au plus tôt. Il s'agit d'un problème d'*ordonnancement*.

Questions.

1. Justifiez que le but est de partitionner l'ensemble des n tâches en deux sous-ensembles A_1 et A_2, l'un s'exécutant sur la machine 1, l'autre sur la machine 2, tels que
$$DT(A_1, A_2) = \max\left(\sum_{i\in A_1} t_i, \sum_{i\in A_2} t_i\right)$$
soit minimum.
2. Notons minDT(k, T) le temps d'arrêt minimal réalisable en plaçant les $n-k$ dernières tâches, lorsque l'on impose 1) que les k premières tâches ont été réparties sur les deux machines (pas forcément de manière optimale) et 2) que la différence (absolue) entre les temps d'arrêt des machines 1 et 2 est $T > 0$ si on considère uniquement les k premières tâches.
   Que vaut minDT(n, T) ?
   Établissez une formule de récurrence (ascendante en k) sur minDT.
3. Application numérique :

k	1	2	3	4
t_k	1	2	2	6

   Quel est l'ordonnancement optimal ?

4 Programmation dynamique et plus court chemin

Proposez un algorithme utilisant la programmation dynamique pour calculer la distance entre deux sommets fixés d'un graphe valué.

Application : trouvez le chemin le plus court de A à B, pour le graphe dont les arêtes sont valuées par :
d(A,C) = 5 ; d(A,D) =3 ; d(A,G) = 14 ; d(D,C) = 11 ; d(D,G) = 6 ; d(D,E) = 7 ; d(C,E) = 3 ; d(C, F) = 2 ; d(F,B) = 7 ; d(E,F) =2 ; d(G,E) = 7 ; d(E,B) = 5 ; d(G,B) = 6.