Feuille d'exercices 3, techniques algorithmiques

Math 312, L3 Université Paris-Sud 11

1er semestre 2008-2009

Voici quelques exercices pour vous entrainer sur les différentes techniques algorithmiques. Ils sont pour l'instant très peu détaillés et donc très difficiles (plus que le niveau prévu de l'examen). Une version plus détaillée vous sera fournie plus tard (et sera sans doute plus proche de ce qu'on attendra de vous à l'examen)...

1 Programmation dynamique

On cherche à calculer une distance particulière entre deux chaînes de caractères.

Cette distance est définie par le nombre minimum d'opérations nécessaires pour passer d'une chaîne à l'autre. Les opérations suivantes sont les seules autorisées :

- substitution : on remplace une lettre par une autre. Par exemple, à partir de "chapeau", on peut obtenir "crapeau".
- délétion : on peut supprimer une lettre de la chaîne initiale. Par exemple, de "crapeau", on peut obtenir "crapau".
- insertion : on peut ajouter une lettre à la chaîne initiale. Par exemple, de "crapau", on peut obtenir "crapaud".

Nous venons de vérifier que la distance de "chapeau" à "crapaud" est d'au plus 3. Intuitivement, on se doute que c'est le meilleur nombre de pas nécessaire au changement de chaîne. Il est certainement plus difficile de se convaincre de la distance de "chalumeau" à "cannibale"...

1. Pouvez vous donner une borne supérieure à la distance entre deux chaînes ? Déduisez-en l'idée d'un algorithme de type "brute-force" pour calculer la distance entre deux chaînes. Quel est l'ordre de grandeur de sa complexité ?
2. Soient $A[1..n]$ et $B[1..m]$ deux chaînes de caractères. Proposez une relation de récurrence entre la distance entre A et B et la distance de leurs sous chaînes $A[1..n-1]$ et $B[1..m-1]$. Appuyez vous sur les trois opérations possibles.
3. Proposez un algorithme de programmation dynamique pour ce problème. Quelle est sa complexité ?

2 Algorithmes gloutons : arbres couvrants de poids min

On a trouvé dans un bout de désert un nouveau gisement d'hydrocarbures. Il a été décidé de placer les puits en n endroits. On connait les longueurs de tuyau nécessaires à la connection de toutes paires de puits (stockées dans une matrice $n \times n$ M). On cherche à connecter tous les puits au dépot en utilisant un minimum de tuyauterie. Pour limiter les problèmes d'étanchéité (et simplifier le problème), on ne s'autorise à connecter des tuyaux qu'au niveau d'un puit.

1. Proposez un algorithme glouton qui calcule la longueur minimale de tuyau nécessaire et propose une solution au problème.
2. Cet algorithme est-il optimal ? Pouvez vous l'améliorer / prouver son optimalité ?

3 Diviser pour régner

Soit $A[1..n]$ un tableau de n entiers (eventuellement négatifs), par exemple $A = [|2, -5, 1, 3, 2, -4, 1|]$. On cherche une portion $A[i..j]$ du tableau dont la somme des éléments est maximale, c'est à dire :

$$\max_{i,j \in \{1..n\}} \sum_{k=i}^{j} A[k]$$

.

1. Décrivez un algorithme naif. Quelle est sa compléxité ?
2. Décrivez un algorithme diviser pour régner.
3. Quelle est la compléxité de l'algorithme précédent ? Que remarquez vous ?