1 TD 10 : Programmation dynamique

(correction page ??)

Abordé lors de cette séance	
programmation	calcul numérique
algorithme	programmation dynamique

La programmation dynamique [1] est une façon de résoudre de manière similaire une classe de problèmes d'optimisation qui vérifie la même propriété. On suppose qu'il est possible de découper le problème P en plusieurs parties P_1, P_2, ... Si S est la solution optimale du problème P, alors chaque partie S_1, S_2 de cette solution appliquée aux sous-problèmes est aussi optimale.

Par exemple, on cherche le plus court chemin $c(A, B)$ entre les villes A et B. Si celui-ci passe par la ville M alors les chemins $c(A, M) + c(M, B) = c(A, B)$ sont aussi les plus courts chemins entre les villes A, M et M, B.

Première demi-heure : Le plus court chemin .

On récupère le fichier `matrix_distance_7398.txt` qui contient des distances entre différentes villes (pas toutes).

```
import pyensae
pyensae.download_data("matrix_distance_7398.zip", website  = "xd")
```

Il contient des données du type :

```
Boulogne-Billancourt        Beauvais        85597
Courbevoie        Sevran        26564
Colombes        Alfortville        36843
Bagneux        Marcq-En-Baroeul        233455
...
```

On lit les données comme ceci :

```
import pandas
df = pandas.read_csv("matrix_distance_7398.txt", sep="\t")
print (df.head())
```

Exercice 1

Construire la liste des villes sans doublons ?

Exercice 2

Constuire un dictionnaire $\{(\texttt{a}, \texttt{b}) : \texttt{d}, (\texttt{b}, \texttt{a}) : \texttt{d}\}$ où $\texttt{a}, \texttt{b}$ sont des villes et `d` la distance qui les sépare ?

On veut calculer la distance entre la ville de `Charleville – Mezieres` et `Bordeaux` ? Est-ce que cette distance existe dans la liste des distances dont on dispose ?

1. http://fr.wikipedia.org/wiki/Programmation_dynamique

Seconde demi-heure : L'algorithme du plus court chemin (Dikjstra)

On créé un tableau d[v] qui contient la distance optimale entre les villes v et Charleville − Mezieres. La valeur qu'on cherche est d[′Bordeaux′]. On initialise le tableau comme suit :

- d[′Charleville − Mezieres′] = 0
- d[v] = infini pour tout $v \neq' Charleville - Mezieres'$.

Exercice 3

Quelles sont les premières cases qu'on peut remplir facilement ?

Exercice 4

Soit une ville v et une autre w, on s'aperçoit que $d[w] > d[v] + dist[w, v]$. Que proposez-vous de faire ? En déduire un algorithme qui permet de déterminer la distance la plus courte entre Charleville-Mezieres et Bordeaux.

Troisième demi-heure : La répartition des skis

$N = 10$ skieurs rentrent dans un magasins pour louer 10 paires de skis (parmi $M > N$). On souhaite leur donner à tous une paire qui leur convient (on suppose que la taille de la paire de skis doit être au plus proche de la taille du skieurs. On cherche donc à minimiser :

$$\arg\min_{\sigma} \sum_{i=1}^{N} \left| t_i - s_{\sigma(i)} \right|$$

Où σ est un ensemble de N paires de skis parmi M (un arrangement [2] pour être plus précis).

A première vue, il faut chercher la solution du problème dans l'ensemble des arrangements de N paires parmi M. Mais si on ordonne les paires et les skieurs par taille croissantes : $t_1 \leqslant t_2 \leqslant ... \leqslant t_N$ (tailles de skieurs) et $s_1 \leqslant s_2 \leqslant ... \leqslant s_M$ (tailles de skis), résoudre le problème revient à prendre les skieurs dans l'ordre croissant et à les placer en face d'une paire dans l'ordre où elles viennent. C'est comme si on insérait des espaces dans la séquence des skieurs sans changer l'ordre :

t_1		t_2	t_3			t_4	...	t_{N-1}		t_N	
s_1	s_2	s_3	s_4	s_5	s_6	s_7	...	s_{M-3}	s_{M-2}	s_{M-1}	s_M

Exercice facultatif

Il faut d'abord prouver que l'algorithme suggéré ci-dessus permet bien d'obtenir la solution optimale.

Exercice 5

Après avoir avoir trié les skieurs et les paires par tailles croissantes. On définit :

$$p(n, m) = \sum_{i=1}^{n} \left| t_i - s_{(}\sigma_m^* i) \right|$$

Où σ_m^* est le meilleur choix possible de n paires de skis parmi les m premières. Exprimer $p(n, m)$

2. http://fr.wikipedia.org/wiki/Arrangement

par récurrence (en fonction de $p(n, m-1)$ et $p(n-1, m-1)$. On suppose qu'un skieur sans paire de ski correspond au cas où la paire est de taille nulle.

Quatrième demi-heure : La répartition des skis (suite et fin)

Exercice 6

Ecrire une fonction qui calcule l'erreur pour la distribution optimale ? On pourra choisir des skieurs et des paires de tailles aléatoires par exemple.

```
import random
skieur = [ random.gauss(1.75, 0.1) for i in range(0,10) ]
paires = [ random.gauss(1.75, 0.1) for i in range(0,15) ]
skieurs.sort()
paires.sort()
```

Exercice 7

Quelle est la meilleure distribution ?

Pour aller plus loin ou pour ceux qui ont fini plus tôt

Quels sont les coûts des deux algorithmes (plus court chemin et ski) ?

Remarques