



DEVOIR D'INFORMATIQUE 6

D.Malka – MPSI 2016-2017 – Lycée Saint-Exupéry

19.06.2017

Durée de l'épreuve : 3h00.

L'usage de la calculatrice n'est pas autorisé.

L'énoncé de ce devoir comporte 6 pages.

- *Si, au cours de l'épreuve, vous repérez ce qui vous semble être une erreur d'énoncé, signalez le sur votre copie et poursuivez votre composition en expliquant les raisons des initiatives que vous êtes amené à prendre.*
- *Il ne faudra pas hésiter à formuler des commentaires. Le barème tiendra compte de ces initiatives ainsi que des qualités de rédaction de la copie.*
- *La numérotation des exercices doit être respectée. Les résultats doivent être systématiquement encadrés.*
- *Les pages doivent être numérotées de la façon suivante : n°page courante/nombre total de pages.*

Exercice 1 – Données générales sur le monde

On considère la base de données `world` relative aux pays et principales villes du monde. Elle est constituées des relations suivantes :

- `City`(Id, Name, *CountryCode*, District, Population)
- `Country`(Code, Name, Continent, SurfaceArea, Population, GNP, LifeExpectancy, *Capital*)
- `CountryLanguage`(*CountryCode*, *Language*, isOfficial, Percentage)

Les clés primaires sont soulignées, les *clés étrangères* sont en italique. Des extraits des différentes tables sont fournis en annexe.

Traduire en langage SQL les requêtes fournissant :

1. le nom et le GNP de tous les pays dont le GNP est supérieur à 100 000 ;
2. le nom et la population de toutes les villes françaises référencées dans la base de données ;
3. le nombre de personnes parlant anglais dans le monde ;
4. la population totale de chaque continent ;
5. le nom et la superficie des pays dont la population moyenne dans les grandes villes est supérieure à 100 000.

Code	Name	Continent	SurfaceArea	Population	LifeExpectancy	GNP	Capital
DOM	Dominican Republic	North America	48511.00	8495000	73.2	15846.00	587
DZA	Algeria	Africa	2381741.00	31471000	69.7	49982.00	35
ECU	Ecuador	South America	283561.00	12646000	71.1	19770.00	594
EGY	Egypt	Africa	1001449.00	68470000	63.3	82710.00	608
ERI	Eritrea	Africa	117600.00	3850000	55.8	650.00	652
ESH	Western Sahara	Africa	266000.00	293000	49.8	60.00	2453
ESP	Spain	Europe	505992.00	39441700	78.8	553233.00	653
EST	Estonia	Europe	45227.00	1439200	69.5	5328.00	3791
ETH	Ethiopia	Africa	1104300.00	62565000	45.2	6353.00	756
FIN	Finland	Europe	338145.00	5171300	77.4	121914.00	3236
FJI	Fiji Islands	Oceania	18274.00	817000	67.9	1536.00	764
FLK	Falkland Islands	South America	12173.00	2000	NULL	0.00	763
FRA	France	Europe	551500.00	59225700	78.8	1424285.00	2974
FRO	Faroe Islands	Europe	1399.00	43000	78.4	0.00	901
FSM	Micronesia, Federated States of	Oceania	702.00	119000	68.6	212.00	2689
GAB	Gabon	Africa	267668.00	1226000	50.1	5493.00	902
GBR	United Kingdom	Europe	242900.00	59623400	77.7	1378330.00	456
GEO	Georgia	Asia	69700.00	4968000	64.5	6064.00	905
GHA	Ghana	Africa	238533.00	20212000	57.4	7137.00	910
GIB	Gibraltar	Europe	6.00	25000	79.0	258.00	915

(a) Table Country

CountryCode	Language	IsOfficial	Percentage	ID	Name	CountryCode	District	Population
ABW	Dutch	T	5.3	1	Kabul	AFG	Kabol	1780000
ABW	English	F	9.5	2	Qandahar	AFG	Qandahar	237500
ABW	Papiamento	F	76.7	3	Herat	AFG	Herat	186800
ABW	Spanish	F	7.4	4	Mazar-e-Sharif	AFG	Balkh	127800
AFG	Balochi	F	0.9	5	Amsterdam	NLD	Noord-Holland	731200
AFG	Dari	T	32.1	6	Rotterdam	NLD	Zuid-Holland	593321
AFG	Pashto	T	52.4	7	Haag	NLD	Zuid-Holland	440900
AFG	Turkmenian	F	1.9	8	Utrecht	NLD	Utrecht	234323
AFG	Uzbek	F	8.8	9	Eindhoven	NLD	Noord-Brabant	201843
AGO	Ambo	F	2.4	10	Tilburg	NLD	Noord-Brabant	193238
AGO	Chokwe	F	4.2	11	Groningen	NLD	Groningen	172701
AGO	Kongo	F	13.2	12	Breda	NLD	Noord-Brabant	160398
AGO	Luchazi	F	2.4	13	Apeldoorn	NLD	Gelderland	153491
AGO	Luibemangwela	F	5.4	14	Nijmegen	NLD	Gelderland	152463
AGO	Luvale	F	3.6	15	Enschede	NLD	Overijssel	149544
AGO	Mbundu	F	21.6	16	Haarlem	NLD	Noord-Holland	148772
AGO	Nyaneka-nkhumbi	F	5.4	17	Almere	NLD	Flevoland	142465
AGO	Ovimbundu	F	37.2	18	Arnhem	NLD	Gelderland	138020
AIA	English	T	0.0	19	Zaanstad	NLD	Noord-Holland	135621
ALB	Albanian	T	97.9	20	's-Hertogenbosch	NLD	Noord-Brabant	129170

(b) Table CountryLanguage

(c) Table City

FIGURE 1 – Extraits de la base de données World

Exercice 2 – Gestion du trafic aérien

1. PLAN DE VOL

Afin d'éviter les collisions entre avions, les altitudes de vol en croisière sont normalisées. Dans la majorité des pays, les avions volent à une altitude multiple de 1000 pieds (un pied vaut 30,48 cm) au-dessus de la surface isobare à 1013,25 hPa. L'espace aérien est ainsi découpé en tranches horizontales appelées niveaux de vol et désignées par les lettres « FL » (flight level) suivies de l'altitude en centaines de pieds : « FL310 » désigne une altitude de croisière de 31000 pieds au-dessus de la surface isobare de référence. Eurocontrol est l'organisation européenne chargée de la navigation aérienne, elle gère plusieurs dizaines de milliers de vol par jour. Toute compagnie qui souhaite faire traverser le ciel européen à un de ses avions doit soumettre à cet organisme un plan de vol comprenant un certain nombre d'informations : trajet, heure de départ, niveau de vol souhaité, etc. Muni de ces informations, Eurocontrol peut prévoir les secteurs aériens qui vont être surchargés et prendre des mesures en conséquence pour les désengorger : retard au décollage, modification de la route à suivre, etc. Nous modélisons (de manière très simplifiée) les plans de vol gérés par Eurocontrol sous la forme d'une base de données comportant deux tables :

— la table **vol** qui répertorie les plans de vol déposés par les compagnies aériennes ; elle contient les colonnes

- ◆ **id_vol** : numéro du vol (chaîne de caractères) ;
- ◆ **depart** : code de l'aéroport de départ (chaîne de caractères) ;
- ◆ **arrivee** : code de l'aéroport d'arrivée (chaîne de caractères) ;
- ◆ **jour** : jour du vol (de type date, affiché au format aaaa-mm-jj) ;
- ◆ **heure** : heure de décollage souhaitée (de type time, affiché au format hh :mi) ;
- ◆ **niveau** : niveau de vol souhaité (entier).

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	07 :35	300
AF1205	FCO	CDG	2016-05-02	10 :25	300
AF1504	CDG	FCO	2016-05-02	10 :05	310
AF1505	FCO	CDG	2016-05-02	13 :00	310

FIGURE 2 – Extrait de la table vol : vols de la compagnie Air France entre les aéroports Charles-de- Gaule (Paris) et Léonard-de-Vinci à Fiumicino (Rome)

— la table **aeroport** qui répertorie les aéroports européens ; elle contient les colonnes

- ◆ **id_aero** : code de l'aéroport (chaîne de caractères) ;
- ◆ **ville** : principale ville desservie (chaîne de caractères) ;
- ◆ **pays** : pays dans lequel se situe l'aéroport (chaîne de caractères).

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie

FIGURE 3 – Extrait de la table aeroport

Les types SQL date et time permettent de mémoriser respectivement un jour du calendrier grégorien et une heure du jour. Deux valeurs de type date ou de type time peuvent être comparées avec les opérateurs habituels (=, <, <=, etc.). La comparaison s'effectue suivant l'ordre chronologique. Ces valeurs peuvent également être comparées à une chaîne de caractères correspondant à leur représentation externe ('aaaa-mm-jj' ou 'hh :mi').

- 1.1 Écrire une requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi.
- 1.2 Écrire une requête SQL qui fournit la liste des numéros de vols au départ d'un aéroport desservant Paris le 2 mai 2016.

1.3 Que fait la requête suivante ?

```

1 SELECT id_vol
2 FROM vol
3 JOIN aeroport AS d ON d.id_aero = depart
4 JOIN aeroport AS a ON a.id_aero = arrivee
5 WHERE
6 d.pays = 'France' AND
7 a.pays = 'France' AND
8 jour = '2016-05-02'

```

1.4 Certains vols peuvent engendrer des conflits potentiels : c'est par exemple le cas lorsque deux avions suivent un même trajet, en sens inverse, le même jour et à un même niveau. Écrire une requête SQL qui fournit la liste des couples (Id1 , Id2) des identifiants des vols dans cette situation.

2. ALLOCATION DES NIVEAUX DE VOL

Lors du dépôt d'un plan de vol, la compagnie aérienne doit préciser à quel niveau de vol elle souhaite faire évoluer son avion lors de la phase de croisière. Ce niveau de vol souhaité, le RFL pour requested flight level, correspond le plus souvent à l'altitude à laquelle la consommation de carburant sera minimale. Cette altitude dépend du type d'avion, de sa charge, de la distance à parcourir, des conditions météorologiques, etc.

Cependant, du fait des similitudes entre les différents avions qui équipent les compagnies aériennes, certains niveaux de vols sont très demandés ce qui engendre des conflits potentiels, deux avions risquant de se croiser à des altitudes proches. Les contrôleurs aériens de la région concernée par un conflit doivent alors gérer le croisement de ces deux avions.

Pour alléger le travail des contrôleurs et diminuer les risques, le système de régulation s'autorise à faire voler un avion à un niveau différent de son RFL. Cependant, cela engendre généralement une augmentation de la consommation de carburant. C'est pourquoi on limite le choix aux niveaux immédiatement supérieur et inférieur au RFL.

Ce problème de régulation est modélisé par un graphe dans lequel chaque vol est représenté par trois sommets. Le sommet 0 correspond à l'attribution du RFL, le sommet + au niveau supérieur et le sommet - au niveau inférieur. Chaque conflit potentiel entre deux vols sera représenté par une arête reliant les deux sommets concernés. Le coût d'un conflit potentiel (plus ou moins important en fonction de sa durée, de la distance minimale entre les avions, etc.) sera représenté par une valuation sur l'arête correspondante.

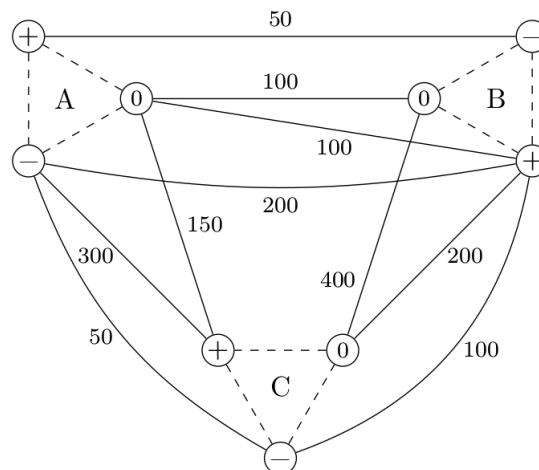


FIGURE 4 – Exemple de conflits potentiels entre trois vols

Dans l'exemple de la figure 3, faire voler les trois avions à leur RFL engendre un coût de régulation entre A et B de 100 et un coût de régulation entre B et C de 400, soit un coût total de la régulation de 500 (il n'y a pas de conflit entre A et C). Faire voler l'avion A à son RFL et les avions B et C au-dessus de leur RFL engendre un conflit potentiel de coût 100 entre A et B et 150 entre A et C, soit un coût total de 250 (il n'y a plus de conflit entre B et C).

On peut observer que cet exemple possède des solutions de coût nul, par exemple faire voler A et C à leur RFL et B au-dessous de son RFL. Mais en général le nombre d'avions en vol est tel que des conflits potentiels sont inévitables. Le but de la régulation est d'imposer des plans de vol qui réduisent le plus possible le coût total de la résolution des conflits.

2.1 Implantation du problème

Chaque vol étant représenté par trois sommets, le graphe des conflits associé à n vols v_0, v_1, \dots, v_{n-1} possède $3n$ sommets que nous numérotions de 0 à $3n - 1$. Nous conviendrons que pour $0 \leq k < n$:

- le sommet $3k$ représente le vol v_k à son RFL ;
- le sommet $3k + 1$ représente le vol v_k au-dessus de son RFL ;
- le sommet $3k + 2$ représente le vol v_k au-dessous de son RFL ;

Le coût de chaque conflit potentiel est stocké dans une liste de $3n$ listes de $3n$ entiers (tableau $3n \times 3n$) accessible grâce à la variable globale `conflit` : si i et j désignent deux sommets du graphe, alors `conflit[i][j]` est égal au coût du conflit potentiel (s'il existe) entre les plans de vol représentés par les sommets i et j . S'il n'y a pas de conflit entre ces deux sommets, `conflit[i][j]` vaut 0. On convient que `conflit[i][j]` vaut 0 si les sommets i et j correspondent au même vol (figure 5). On notera que pour tout couple de sommets (i, j) , `conflit[i][j]` et `conflit[j][i]`, représentent un seul et même conflit et donc `conflit[i][j] == conflit[j][i]`.

```
conflit = [ [ 0, 0, 0, 100, 100, 0, 0, 150, 0 ],
            [ 0, 0, 0, 0, 0, 50, 0, 0, 0 ],
            [ 0, 0, 0, 0, 200, 0, 0, 300, 50 ],
            [ 100, 0, 0, 0, 0, 0, 400, 0, 0 ],
            [ 100, 0, 200, 0, 0, 0, 200, 0, 100 ],
            [ 0, 50, 0, 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 0, 400, 200, 0, 0, 0, 0 ],
            [ 150, 0, 300, 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 50, 0, 100, 0, 0, 0, 0 ] ]
```

FIGURE 5 – Tableau des coûts des conflits associé au graphe représenté figure 4

- 2.1.1 Écrire en Python une fonction `nb_conflits()` sans paramètre qui renvoie le nombre de conflits potentiels, c'est-à-dire le nombre d'arêtes de valuation non nulle du graphe.
- 2.1.2 Exprimer en fonction de n la complexité de cette fonction.

2.2 Régulation

Pour un vol v_k on appelle niveau relatif l'entier r_k valant 0, 1 ou 2 tel que :

- $r_k = 0$ représente le vol v_k à son RFL ;
- $r_k = 1$ représente le vol v_k au-dessus de son RFL ;
- $r_k = 2$ représente le vol v_k au-dessous son RFL.

On appelle régulation la liste $(r_0, r_1, \dots, r_{n-1})$. Par exemple, la régulation $(0, 0, \dots, 0)$ représente la situation dans laquelle chaque avion se voit attribuer son RFL. Une régulation sera implantée en Python par une liste d'entiers. Il pourra être utile d'observer que les sommets du graphe des conflits choisis par la régulation r portent les numéros $3k + r_k$ pour $0 \leq k < n$. On remarque également qu'au sommet s du graphe correspond le niveau relatif $r_k = s \bmod 3$ et le vol v_k tel que $k = \lfloor s/3 \rfloor$.

- 2.2.1 Écrire en Python une fonction `nb_vol_par_niveau_relatif(regulation)` qui prend en paramètre une régulation (liste de n entiers) et qui renvoie une liste de 3 entiers $[a, b, c]$ dans laquelle a est le nombre de vols à leurs niveaux RFL, b le nombre de vols au-dessus de leurs niveaux RFL et c le nombre de vols au-dessous de leurs niveaux RFL.

2.2.2 Coût d'une régulation

On appelle coût d'une régulation la somme des coûts des conflits potentiels que cette régulation engendre.

- 2.2.2.1 Écrire en Python une fonction `cout_regulation(regulation)` qui prend en paramètre une liste représentant une régulation et qui renvoie le coût de celle-ci.
- 2.2.2.2 Évaluer en fonction de n , la complexité de cette fonction.
- 2.2.2.3 Dédurre de la question 2.2.2.1 une fonction `cout_RFL()` qui renvoie le coût de la régulation pour laquelle chaque vol est à son RFL.
- 2.2.3 Combien existe-t-il de régulations possibles pour n vols ? Est-il envisageable de calculer les coûts de toutes les régulations possibles pour trouver celle de coût minimal ?

Opérations et fonctions Python disponibles

Fonctions

- `exp(x)` calcule l'exponentielle du nombre `x`
- `randint(n)` (`n` entier) renvoie un entier aléatoire compris entre 0 et `n-1` inclus
- `random()` renvoie un nombre flottant tiré aléatoirement dans $[0, 1[$ suivant une distribution uniforme
- `sqrt(x)` calcule la racine carrée du nombre `x`
- `time()` renvoie l'heure sous la forme d'un nombre de secondes depuis un instant de référence

Opérations sur les listes

- `u + v` construit une liste constituée de la concaténation des listes `u` et `v` : `[1, 2] + [3, 4, 5] → [1, 2, 3, 4, 5]`
- `n * u` construit une liste constitué de la liste `u` concaténée `n` fois avec elle-même : `3 * [1, 2] → [1, 2, 1, 2, 1, 2]`
- `u.append(e)` ajoute l'élément `e` à la fin de la liste `u`
- `del(u[i])` supprime de la liste `u` son élément d'indice `i`
- `u.insert(i, e)` insère l'élément `e` à la position d'indice `i` dans la liste `u` (en décalant les éléments suivants); si `i >= len(u)`, `e` est ajouté en fin de liste
- `len(u)` donne le nombre d'éléments de la liste `u` :
- `u[i], u[j] = u[j], u[i]` permute les éléments d'indice `i` et `j` dans la liste `u`