Correction du DS 2

Informatique pour tous, deuxième année

Julien Reichert

Exercice 1

Question de cours, pas de correction.

Exercice 2

```
def tri_lyes(l):
    n = len(l)
    ll = [None] * n
    for i in range(len(l)): # quadratique car...
        nbinf = 0
        nbeg = 0
        for j in range(len(l)): # ... linéaire
            if l[i] > l[j]:
                nbinf += 1
            elif l[i] == l[j] and i > j:
                nbeg += 1
        ll[nbinf+nbeg] = l[i] # ... et constant
    return ll
```

Exercice 3

```
def tri_dubos(l):
    n = len(l)
    ll = [None] * (2**n-1)
    for element in l:
        deb, fin = 0, 2**n-2
        mil = (deb+fin)//2
        while ll[mil] != None:
            if ll[mil] < element:
                deb = mil + 1
            else:
                fin = mil-1
            mil = (deb+fin)//2
        ll[mil] = element
    return [x for x in ll if x != None] # plus court qu'une boucle
```

Exercice 4

```
def tri_dijkstra(l):
    n = len(l)
    nb_0 = 0 # l'indice du dernier est donc un de moins
```

```
    bn_2 = n-1 # l'indice du premier sera toujours un de plus que ceci
    for i in range(n):
        while l[i] == 2:
            if i <= bn_2:
                l[i], l[bn_2] = l[bn_2], l[i]
                bn_2 -= 1
            else:
                return # on sait qu'on ne rencontrera plus que des 2, donc on arrête le tri
        if l[i] == 0:
            if i > nb_0:
                l[i], l[nb_0] = l[nb_0], l[i]
            nb_0 += 1
```

Exercice 5

Question 1

```
SELECT * FROM Resultats WHERE Score = (SELECT MAX(Score) FROM Resultats)
```

Éventuellement LIMIT 1 pour n'avoir qu'un enregistrement (arbitraire) en cas d'égalité.

... ou plus simplement :

```
SELECT * FROM Resultats ORDER BY Score DESC LIMIT 1 (un seul enregistrement)
```

L'étoile est la bienvenue vu qu'on récupère tout.

Question 2

```
SELECT MAX(Score) FROM Resultats WHERE Date = d
```

Puisqu'on ne récupère qu'un champ agrégé, on n'a pas besoin de tant d'artifices.

Bien entendu, on peut une fois de plus passer par le tri.

```
SELECT Score FROM Resultats WHERE Date = d ORDER BY Score DESC LIMIT 1
```

Question 3

```
SELECT COUNT(*) FROM Resultats WHERE Score > s AND Date = d
```

Même remarque qu'à la question 2.

L'inégalité peut être large, ce n'est pas pénalisé, mais on prépare le terrain pour la suite ici.

Question 4

```
SELECT COUNT(*) FROM Resultats WHERE Score > (SELECT Score FROM Resultats WHERE Id = n AND Date =
d) AND Date = d
```

On récupère un seul résultat, donc tout va relativement bien.

Question 5

```
SELECT R1.Id, R2.Id FROM Resultats AS R1 JOIN Resultats AS R2 ON R1.Score > R2.Score WHERE R1.Date
= d AND R2.Date = d
```

Ici, si on est rebuté à l'idée d'utiliser autre chose qu'une égalité dans la jointure, on peut inclure la comparaison à la clause WHERE voire remplacer aussi le JOIN par une virgule.

Le mot-clé AS n'est pas nécessaire, mais je n'envisage pas de le retirer, c'est une histoire de présentation.

Question 6

```
SELECT Id, 1 AS Place FROM Resultats WHERE Score = (SELECT MAX(Score) FROM Resultats WHERE Date =
d) AND Date = d UNION SELECT R2.Id, COUNT(*)+1 AS Place FROM Resultats AS R1 JOIN Resultats AS R2
ON R1.Score > R2.Score WHERE Date = d GROUP BY R2.Id ORDER BY Place
```

C'est vrai que ce serait plus facile avec des $<=$ car on n'aurait pas les maxima qui disparaissent, mais le vrai classement s'obtient en connaissant le nombre de personnes qui ont strictement plus (sinon c'est compliqué de le récupérer, et on préfère alors utiliser un programme annexe).

L'ordre n'importe pas ici (comme partout sauf à la question 5 faute de symétrie).

Pour le cas où on voudrait le classement général sur les totaux, une table dérivée s'impose, et donc on ne prendra pas les résultats dans Resultats WHERE Date = d mais dans `(SELECT SUM(Score) AS Total FROM Resultats GROUP BY Id)`, une table dérivée qui doit avoir un alias.

La requête devient alors

```
SELECT Id, 1 AS Place FROM (SELECT SUM(Score) AS Total FROM Resultats GROUP BY Id) AS td WHERE Total
= (SELECT MAX(Total) FROM (SELECT SUM(Score) AS Total FROM Resultats GROUP BY Id) AS td2) UNION SELECT
R2.Id, COUNT(*)+1 AS Place FROM (SELECT SUM(Score) AS Total FROM Resultats GROUP BY Id) AS R1 JOIN
(SELECT SUM(Score) AS Total FROM Resultats GROUP BY Id) AS R2 ON R1.Total > R2.Total GROUP BY R2.Id
ORDER BY Place
```