Correction du DS 2bis

Julien Reichert

Partie A

Exercice A1

```
def pppd(n): # Plus petite puissance de dix
    rep = 1
    while rep <= n:
        rep *= 10
    return rep
```

Exercice A2

Attention à ne pas muter g, ni une copie qui ne serait que superficielle. Par ailleurs, les couples ne sont pas mutables donc il faudrait remplacer le couple en entier.

```
def encodage(g):
    g2 = []
    dpk = pppd(len(g)) # dix puissance k
    for liste in g:
        liste2 = []
        for (v, p) in liste:
            liste2.append((v, dpk * p + 1))
        g2.append(liste2)
    return g2
```

Exercice A3

Les lignes manquantes, toutes dans le corps de boucle avec une indentation relative en plus pour les deux dernières :

```
if distances[t] is None or distances[t] > mini + p:
    distances[t] = mini + p
    distancesbis[t] = mini + p
```

Exercice A4

```
def optimal_et_court(g, s, t):
    g2 = encodage(g)
    dpk = pppd(len(g))
    distances_s = dijkstra(g2, s)
    d_s_t = distances_s[t]
    return d_s_t // dpk, d_s_t % dpk
```

Exercice A5

Sur le principe, tout fonctionne de la même façon, mais l'encodage des poids est changé : au lieu d'avoir $p \mapsto 10^k p+1$, on peut faire $p \mapsto 10^h + p$ avec 10^h la plus petite puissance de dix strictement supérieure à la somme de tous les poids des arcs, de sorte de ne pas avoir là non plus de problèmes de débordement. On peut évidemment raffiner la valeur de 10^h (par exemple la somme des poids maximaux par liste d'adjacence), mais l'intérêt est limité.

Partie B

Exercice B1

La ligne manquante triait la liste `distances` en place, par exemple `distances.sort()`.

Exercice B2

```
def dist_eucl_dim_d(pt1, pt2):
    assert len(pt1) == len(pt2) # cela ne peut pas faire de mal
    rep = 0
    for i in range(len(pt1)):
        rep += (pt1[i] - pt2[i]) ** 2
    return rep ** .5 # ou rep si on prend la distance au carré (le tri sera identique)
```

Un élément de `les_points` sera alors par exemple `((4, 2), "rouge")`, donc un point dans un espace de dimension deux, de coordonnées $(4, 2)$ et de couleur rouge.

Exercice B3

La variable `etiq` correspond à sa valeur au dernier tour de boucle, donc à l'étiquette du `k`-ième plus proche voisin (avec la gestion des éventuelles égalités faite par le tri).

Exercice B4

Changement au niveau de la boucle, avec une indentation relative comme dans la partie A :

```
for d, etiq in distances[:k]:
    if d == 0:
        return etiq
    if etiq in etiquettes:
        etiquettes[etiq] += 1 / d
    else:
        etiquettes[etiq] = 1 / d
```

Partie C

Exercice C1

```
SELECT Score FROM Resultats WHERE Id=19 AND Date="2023-03-01"
```

Exercice C2

On pourra mettre ou non `DISTINCT`. La clé primaire suggère que c'est inutile, mais si cette clé est retirée ultérieurement pour permettre des doublons, la requête resterait correcte avec.

```
SELECT Date FROM Resultats WHERE Id=5
```

Exercice C3

Cette fois-ci le `DISTINCT` est nécessaire. Une autre possibilité serait de faire une sous-requête avec un `GROUP BY`, mais c'est plus lourd.

```
SELECT COUNT(DISTINCT Date) FROM Resultats
```

Exercice C4

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

Si on accepte de n'avoir qu'un résultat en cas d'égalité, on peut aussi écrire :

```
SELECT * FROM Resultats ORDER BY Score DESC LIMIT 1
```

Exercice C5

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

Autre possibilité :

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

Exercice C6

Attention à ne pas considérer `d` comme une constante, idem pour tout ce qui est analogue par la suite.

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

Exercice C7

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

Exercice C8

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

Exercice C9

```
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.Date = R2.Date
WHERE R1.Date = d AND R1.Score > R2.Score GROUP BY R2.Id
ORDER BY Place
```

Exercice C10

```
SELECT Id FROM Resultats GROUP BY Id HAVING COUNT(*) =
  (SELECT COUNT(DISTINCT Date) FROM Resultats)
```