SGBLE

Zeste de savoir

Chercher dans un CSV de 3 Go en quelques instants

16 mars 2021

Table des matières

Votre premier cours d'algorithmique introduisait sûrement la notion de dichotomie pour jouer efficacement au *Plus ou Moins*, le jeu dans lequel le sphinx choisit un nombre entier entre n et m à deviner. La solution optimale étant alors de proposer $\frac{n+m}{2}$ afin d'éliminer à chaque coup la moitié des possibilités totales dans l'intervalle $[n, m]$. *Très* rapidement, on converge vers la solution.

Aussi étonnant que cela puisse paraître, je n'ai eu que très peu l'occasion d'implémenter ce grand classique dans de véritables projets professionnels.
Eh oui! il faut dire qu'il y a toujours une lib à portée de main (style `bisect`) qui implémente déjà le bon algorithme pour le problème, ou bien parce que l'on se repose souvent sur une base de données qui utilise déjà des structures efficaces en interne.

Mais récemment, j'ai enfin eu l'occasion d'implémenter l'algorithme du sphinx. L'exercice: une recherche dans une base de données CSV de 3 Go, 1 524 580 lignes. Il s'agit d'une base de données d'Open Food Facts dans laquelle il faut trouver les informations d'un produit avec comme clef d'accès son code barre, en première colonne.

Évidemment, il n'est pas concevable de charger 3 Go dans un programme à chaque fois que l'on a besoin d'une entrée. Il faut donc procéder autrement, sans jamais charger le fichier en mémoire.

1. Solution

On va chercher dans un intervalle entre `mini` et `maxi`. Au début, `mini = 0` et `maxi = file_size` (en octets).

À chaque itération on se place sur l'octet du milieu grâce à `f.seek()`. Comme on ne peut pas être sûr de tomber exactement en début de ligne CSV, on fait un `f.readline()` dans le vide pour se placer au début de la ligne suivante. Dès lors, il n'y a plus qu'à lire et *parser* cette ligne CSV pour voir si l'on tombe sur le code barre recherché. Si c'est le cas, on peut retourner la ligne CSV complète, on a fini. Si non, on compare simplement le code barre courant avec le code barre cible pour ré-ajuster les bornes `mini` et `maxi` en retranchant le précédent intervalle de moitié.

Le code explique tout ça mieux:

```python
def lookup_product(target):
    filename = DATAFILE_OPENFOODFACTS  # Le fichier CSV de 3 Go
    file_size = os.stat(filename).st_size
    max_steps = int(math.log(file_size) / math.log(2)) + 1

    with open(filename, 'r') as f:
        mini = 0
        maxi = file_size
        for _ in range(max_steps + 1):
            mid = (mini + maxi) // 2
            f.seek(mid)  # Se placer au milieu de l'intervalle
            f.readline()  # Avancer à la prochaine ligne
            line = f.readline()  # Consommer la ligne courante
            code = line.split('\t', 1)[0]  # lire le code barre,
                1re colonne du CSV
            if target < code:
                maxi = mid
            elif target > code:
                mini = mid
            else:
                return line.split('\t')  # Trouvé
```

Pour être sûr de ne pas tomber dans une boucle infinie, le `for _ in range(max_steps + 1)` impose une limite du nombre d'itérations. Avec un savant calcul, j'ai estimé au feeling que c'était $log_2(s) + 1$, avec $s =$ `file_size`. (c'est sûrement un peu brouillon, je suis une quiche en maths…)

On peut remarquer un problème dans la fonction: on ne peut jamais atteindre la toute première ligne du CSV. En effet, même si l'on tombait sur le premier octet, le `f.readline()` nous amènerait aussitôt sur la deuxième ligne. En fait, ce n'est pas un problème puisque la première ligne est le *header* du CSV qu'on cherchera à éviter.

Sur un dédié aux capacités plutôt modestes (disque dur SATA) les accès avec `seek` successifs sont plutôt satisfaisants, en cache froid. Cela ne dépasse guère quelque millisecondes.

Voilà comment, après des années dans le domaine, j'ai enfin implémenté l'exercice 1 du chapitre 1 de n'importe quel livre d'algo dans un cas réel 🍊

2. Crédits

Crédits pour l'image: `Bisection_method.svg` de *Tokuchan*, sous licence CC BY-SA 3.0 (source: https://commons.wikimedia.org/w/index.php?curid=9382140)