TP 6
Dask Array
Dask Dataframe

Master SID 1 SD
Benoist GASTON
benoist.gaston@univ-rouen.fr

Dask

Exercice 01
Pi avec Monte Carlo

- L'objectif est de paralléliser le calcul de Pi avec Monte Carlo à l'aide de Dask array.
- Le notebook `TP06_Ex01_DaskArrayPiMC.ipynb` reprend le calcul de π par méthode de Monte Carlo en générant l'ensemble des coordonnées des points tirés dans un ndarray.
- **Questions**
    1. Observer les temps d'exécution en augmentant progressivement le nombre de tir (et donc la taille du ndarray).
    2. Adapter l'algorithme afin d'utiliser un dask array au lieu d'un ndarray.
    3. Observer les temps d'exécution et la consommation mémoire.
    4. Augmenter le nombre de tirs.

Exercice 02
wordCount avec dataFrame

- Le notebook `TP06_02_wc.ipynb` réalise un algorithme de word counting sur un fichier en utilisant pandas.
- Le fichier est lu tel un fichier `csv` par **`pandas`** chaque ligne est un index. Chaque line est découpé en colonne (une par mot) et les colonnes sont ensuite empilées. Finalement la méthode `value_count` est appliquée.
- **Questions**
    1. Évaluer le temps de restitution (commande magique `%%time`) de l'algorithme pour les fichiers `data/wordcount0.txt` et `data/wordcount1.txt`
    2. Écrire le même algorithme en utilisant `dask.dataframe` au lieu de pandas (attention sur la méthode stack).
    3. Évaluer le temps de restitution pour les fichiers `data/wordcount0.txt` and `data/wordcount1.txt`.
    4. Comparer les temps de restitutions.
    5. Essayer la solution pandas sur le fichier `data/wordcount2.txt`. Quelles observations ?
    6. Même question avec la solution Dask

TP 6
Dask Array
Distance de Levenshtein

Master SID 1 SD
Benoist GASTON
benoist.gaston@univ-rouen.fr

Distance de Levenshtein

La distance de Levenshtein est une métrique permettant de mesurer la différence entre deux séquences de caractères. Cette distance correspond au nombre minimal d'opérations élémentaires (insertion, suppression ou substitution de caractères) nécessaires pour transformer une séquence en une autre.

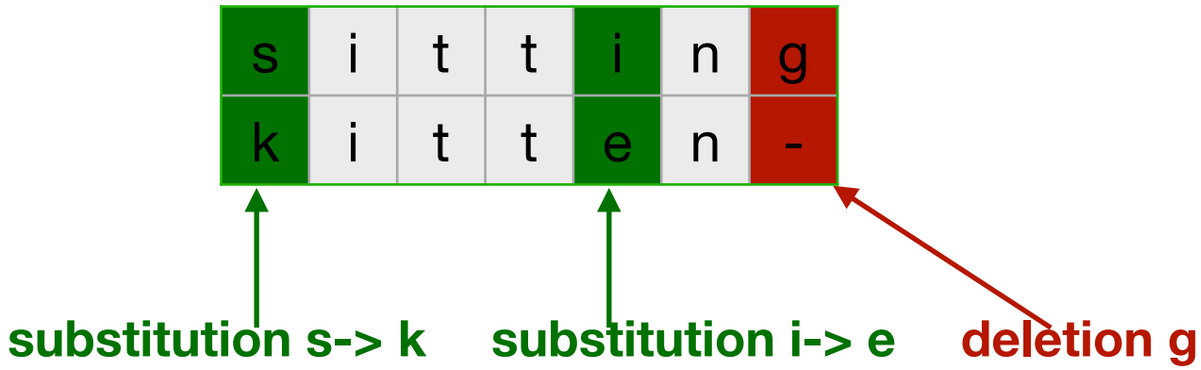
Par exemple, la distance entre `sitting` et `kitten` est de 3 (2 substitutions, 1 suppression).

Formellement, la distance peut être définie de manière récursive selon la définition à droite, où :

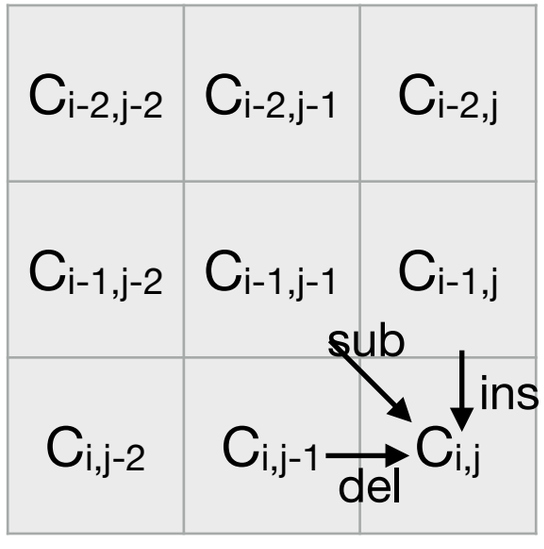
Pour une chaîne de caractères s, $|s|$ représente sa longueur et $s - 1$ désigne la chaîne s sans son premier caractère.

À partir de cette définition, il en résulte un algorithme naïf permettant de calculer la distance de Levenshtein entre deux séquences en remplissant une matrice de coûts.

L'objectif de cet exercice est d'établir un algorithme distribué du calcul de la distance de Levenshtein et, si possible, de le mettre en oeuvre avec Dask



$$\mathrm{lev}(a,b) = \begin{cases} \max(|a|,|b|) & \text{si } \min(|a|,|b|) = 0, \\ \mathrm{lev}(a-1,b-1) & \text{si } a[0] = b[0], \\ 1 + \min \begin{cases} \mathrm{lev}(a-1,b) \\ \mathrm{lev}(a,b-1) \\ \mathrm{lev}(a-1,b-1) \end{cases} & \text{sinon.} \end{cases}$$



		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

Distance de Levenshtein

Le notebook `M1SD-TP06-DaskArray-Levenshtein.ipynb` propose une implémentation du calcul de la distance de Levenshtein par utilisation de `ndarray`.

Questions

1. **Version naïve.**

a) Tester l'algorithme sur différentes séquences en augmentant la taille des séquences et relever les temps de restitution.

b) Sur la base du code existant écrire un version utilisant des dask array.

c) Tester sur de petites séquences, observer le graphe des tâches et relever les temps de restitution.

2. **Optimisations**

a) Sur la base de la version numpy, écrire un algorithme non parallèle où la matrice principale est remplie en la parcourant par bloc de taille fixe (par exemple 100×100) : on remplit le premier bloc puis on passe au suivant.

b) Relever les temps de restitution.

c) Sur la base de cette version établir un algorithme parallèle et tenter de le mettre en oeuvre à l'aide de Dask array, pour cela regarder les méthodes map_blocks et map_overlap.

d) Relever les temps de restitution