DNAlgo: Mini-projet, Alignement de séquences (LU3IN003 - Sorbonne Université) 2019

Description:

Ce projet porte sur un problème de génomique : l'alignement de séquences. D'un point de vue biologique, il s'agit de mesurer la similarité entre deux séquences d'ADN, que l'on voit simplement comme des suites de nucléotides. Cela permet, lorsqu'un nouveau génome est séquencé, de comparer ses gênes à ceux de génomes précédemment séquencés, et de repérer ainsi des homologies, c'est-à- dire les ressemblances dues au fait que les deux espèces ont un ancêtre commun qui leur a transmis ce gêne, même si ce gêne a pu subir des mutations (évolutions) au cours du temps. D'un point de vue informatique, les séquences de nucléotides sont vues comme des mots sur l'alphabet { A,T,G,C } et l'on est ramené à deux problèmes d'algorithmique du texte : le calcul de la distance d'édition entre deux mots quelconques et la production d'un alignement réalisant cette distance. Pour chacun de ces problèmes, on s'intéresse d'abord à un algorithme naïf, puis à un algorithme de programmation dynamique. Enfin, on utilise la méthode diviser pour régner pour améliorer la complexité spatiale de ces algorithmes. En ouverture, on s'intéresse à un problème légèrement différent : l'alignement local de séquences.

Installation:

To install the necessary libraries in order to run the scripts run:

pip3 install -r requirements.txt

Environment:

This repository was tested in Debian based distribution, with Python 3.7.4+.
CPU: Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
RAM: 8 GBs

Usage:

Tasks:

`To run see the results of a given task, run one of the task_?.py files. (? = {a, b, c, d})`

Main.py:

`To construct a new sequences with gaps, run :`
`python3 main.py -p SEQUENCE_TXT NUMBER_OF_GAPS_TO_ADD`
`Exemple:`

```
shetsecure@pararap:~/DNAlgo$ py main.py -p TGCA 3
Predicted answer : 35.0
Constructed 35 permutations.
{'TG-C-A-', 'TG--C-A', 'TGC---A', '-TGC-A-', '-T-GC-A', 'TG---CA', '-TGC--A', 'TGC-A--', '-TG--CA', 'T-GC--A', 'T--G-CA', '--TG-CA', 'T
G-CA--', '--T-GCA', 'T-G-C-A', 'TGC--A-', '-T-G-CA', '--TGC-A', '-TGCA--', 'TG--CA-', '-TG-C-A', '---TGCA', 'T-GCA--', 'TG-C--A', 'TGCA
---', 'T-G-CA-', 'T-G--CA', 'T--GCA-', 'T---GCA', 'T--GC-A', '-T-GCA-', '--TGCA-', 'T-GC-A-', '-T--GCA', '-TG-CA-'}
```

[+] To run dist algorithm on a file:
python3 main.py -f - dist PATH_TO_FILE
Exemple:

```
shetsecure@pararap:~/DNAlgo$ py main.py -f -dist Instances_genome/Inst_0000010_44.adn
10
Time and memory used respectively :   0.1 secs  36.5 KBytes
```

[+] To run dist1 algorithm on a file:
python3 main.py -f -dist1 PATH_TO_FILE
Exemple:

```
shetsecure@pararap:~/DNAlgo$ py main.py -f -dist1 Instances_genome/Inst_0000010_44.adn
10
Time and memory used respectively :   0.0 secs  36.4 KBytes
```

[+] To run the prog_dyn method on a file:
python3 main.py -f -dyn PATH_TO_FILE
Exemple:

```
shetsecure@pararap:~/DNAlgo$ py main.py -f -dyn Instances_genome/Inst_0000010_44.adn
[10, ('TATATGAGTC', 'TAT-T---T-')]
Time and memory used respectively :   0.0 secs  36.4 KBytes
```

[+] Pour voir le graph des appels des fonctions, ajouter l'argument -graph a la fin:
Exemple:

```
shetsecure@pararap:~/DNAlgo$ py main.py -f -dist Instances_genome/Inst_0000010_44.adn -graph
A functions's calls graph will be generated and saved in the cwd.
10
Time and memory used respectively :   1.1 secs  39.1 KBytes
shetsecure@pararap:~/DNAlgo$
```