Arnau Claramunt 17/05/2024

CDI - FIB

Práctica Compresión ADN

Explicación del código:

Para hacer la primera versión del compresor sencilla y rápida, he optado por asignar dos bits por cada letra de ADN.
De esta sencilla forma, conseguimos codificar en vez de 8 bits por cada carácter, solo con 2 bits.
Los valores de los bits han sido escogidos arbitrariamente y hay formas mejor de hacerlo, como por ejemplo con una codificación de Huffman y otras técnicas vistas en clase.

Uso el módulo *argparse* para tratar los argumentos que se introducen al ejecutar el programa. Este ya viene con el python a partir de la versión 3.2 y superiores. Por lo tanto, supongo que en el entorno Linux de los laboratorios, debe funcionar bien.

Información extra codificada:
Al codificar de dos en dos bits, hay veces que no son múltiplos de 8 para convertirlos a bytes al codificar. Por tanto, se tendría que añadir un padding. En mi caso voy a aprovechar el bit extra de padding para indicar si hay que quitar bits que no forman parte del adn. En el peor caso, es cuando es múltiplo de 8 y en el byte extra envío todo 0s. En los otros casos que habría 2, 4 o 6 de padding, pero aprovecho ese último byte y congo un código que cuando lo lea sepa que tenga que quitar ese número de bits. (Por ejemplo si serían 2 bits de padding, añadir al final “01” y al decodificar solo habrá que quitar estos 2 bits).

Formato del fichero comprimido:
Una vez se comprime el fichero de ADN, este está en el formato **.bin**
(nombre: nombre_fichero_adn.bin)
Decidí poner esta extensión, ya que el contenido es en bytes. Considero que así queda claro que internamente hay datos binarios y otra idea de formato que pensé, sería .dna

Luego al descomprimir tiene el mismo nombre que el fichero de ADN original y es un .txt.