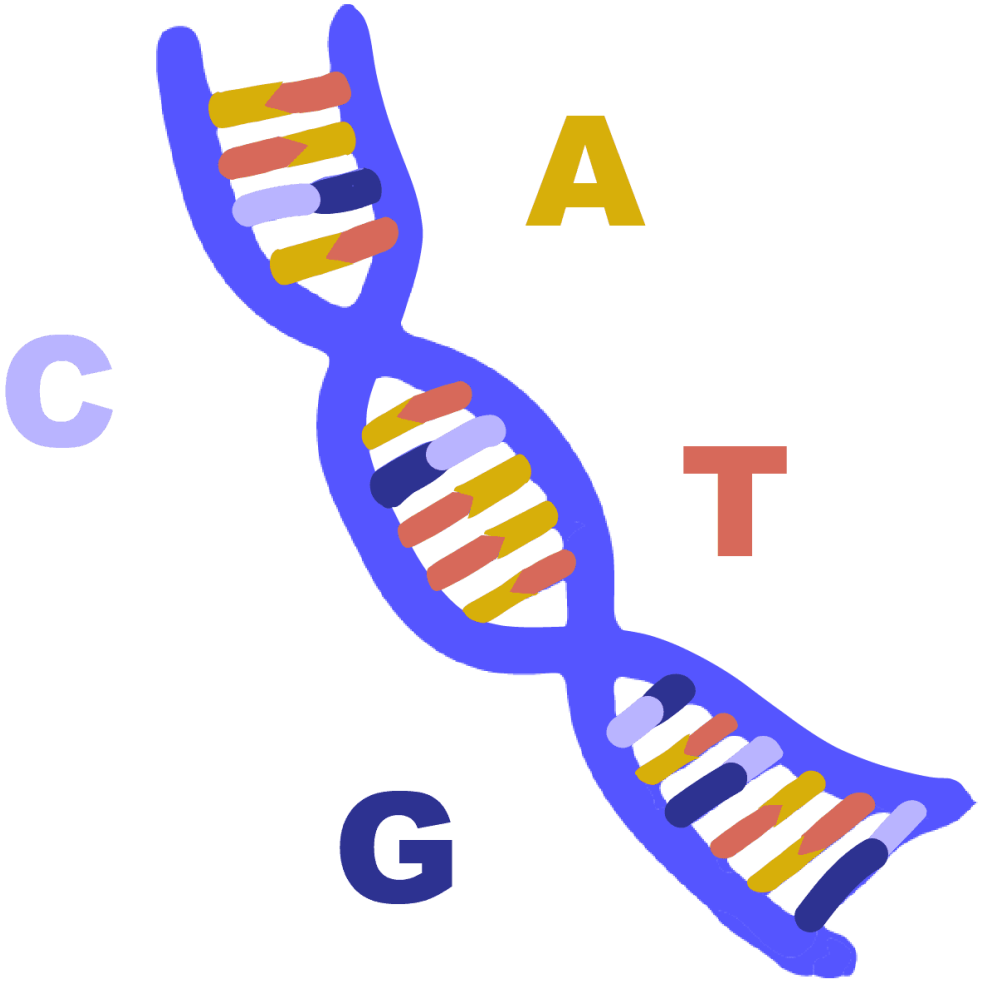
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Documentación de la práctica de ADN



1. Composición y funcionamiento del programa:

El compresor está compuesto por dos algoritmos de compresión:

- Códigos de Rice
- Codificación aritmética con reescalados y números enteros

Para comprimir, el programa hace lo siguiente:

1. Obtiene el contenido del archivo indicado en la entrada .
2. Obtiene la frecuencia de cada carácter de ADN (A, T, G, C).
3. Codifica las frecuencias de cada carácter con Rice.
4. Codifica el contenido del archivo con codificación aritmética y usando las frecuencias de cada carácter para hacer la tabla de probabilidades.
5. Junta el código creado en el paso 3 y 4, procede a pasar el código binario a bytes y, por último, insertarlo en un fichero con el mismo nombre que el archivo sin descomprimir, pero con terminación ".ch".

Para descomprimir, el programa hace lo siguiente:

1. Obtiene el contenido del archivo indicado en la entrada y convierte el código de bytes a un string binario.
2. Se descomprime parte del código con Rice para obtener la lista de frecuencias.
3. Se descomprime el código restante con codificación aritmética.
4. Se inserta el mensaje descomprimido en un fichero con el mismo nombre que el archivo comprimido, pero sin la terminación ".ch".

2. Explicación del código

2.1. Variables globales

2.1.1. Alfabeto

Es una lista que almacena el alfabeto utilizado en el programa.

2.1.2. Alfdict

Es un diccionario que almacena pares (Carácter, Número) y es utilizado en la compresión con codificación aritmética para encontrar rápidamente el límite inferior y superior del intervalo de búsqueda.

2.2. Funciones

2.2.1. do_alfabeto_and_dict

Su propósito es crear el alfabeto y diccionario de forma automática utilizado en el programa.
Fue creado principalmente para hacer pruebas con la codificación aritmética codificando dos o más carácteres a la vez, aunque no resultaron mejores que codificar carácter por carácter, ya que no proporcionaban mejores resultados.

2.2.2. obtener_frecuencias

Su propósito es obtener el número de ocurrencias de cada carácter con la función count() que es más rápida que intentar hacerlo con un bucle recorriendo el mensaje.

2.2.3. num2rice

Su propósito es codificar un número dado con Rice con el parámetro k que es pasado por parámetro, que és 10, a la función.

2.2.4. listnum2rice

Su funcionamiento es comprimir toda una lista de números en un código de Rice. Su propósito es codificar la lista de frecuencias. Este se hace a partir de un bucle que codifica número por número.

2.2.5. rice2listnum

Su funcionamiento es descomprimir todo un código de Rice en una lista de números. Su propósito es descodificar la lista de frecuencias comprimidas. Igualmente se hace a partir de un bucle que descodifica número por número.

2.2.6. tabla_de_frec

Su propósito es obtener la tabla de frecuencias utilizada en la codificación aritmética.
Este se hace a partir de una lista de comprensión que crea un diccionario de pares (letra, probabilidad)

2.2.7. Decimal_to_binary

Su función es convertir un número de decimal a binario. Se usa para el último paso de la codificación aritmética.

2.2.8. IntegerArithmeticEncode

Su función es codificar el mensaje pasado por parámetro a un número que lo represente y pasarlo a binario a partir del algoritmo de codificación aritmética.

Este se implementa de la siguiente manera:

1. Se obtiene la suma de frecuencias
2. Se obtiene el límite superior inicial del intervalo
3. Se obtiene la tabla de probabilidades a partir de la suma de frecuencias
4. Se obtiene el intervalo de probabilidades acumuladas de cada carácter
5. Se obtiene el límite inferior inicial del intervalo
6. Por cada carácter del mensaje:
    a. Se va ajustando el límite superior e inferior del intervalo
    b. Si el intervalo es contenido entre la primera o segunda mitad del intervalo inicial o entre un tercio y tres tercios del intervalo, se aplican las reajustes a los límites y el número que representa el mensaje y añade bits al código por cada iteración del reajuste.
7. Se añaden los bits acumulados al código
8. Se pasa el número que representa el mensaje a binario y se añade al código.
9. Se devuelve el código

2.2.9. findPos

Su función es, a partir del intervalo de probabilidades acumulada, número a decodificar e intervalos superiores e inferiores, calcular los nuevos intervalos inferiores, superiores y posición del intervalo, para insertar el carácter decodificado en el mensaje, y serán utilizados en el decodificador de la codificación aritmética.

2.2.10. IntegerArithmeticDecode

Su función es descodificar el código pasado por parámetro al mensaje que representa a partir del algoritmo de codificación aritmética.

Este se implementa de la siguiente manera:

1. Se obtiene la suma de frecuencias
2. Se obtiene el límite superior inicial del intervalo
3. Se obtiene la tabla de probabilidades a partir de la suma de frecuencias
4. Se obtiene el intervalo de probabilidades acumuladas de cada carácter
5. Se obtiene el límite inferior inicial del intervalo
6. Por cada número del código:

   a. Mientras que el intervalo no encaje en ninguna de las mitades del intervalo inicial, ni entre el primer y tercer tercio del intervalo inicial, se va decodificando el mensaje.
   b. Se reajusta los límites del intervalo y el número que representa el código
7. Se devuelve el mensaje decodificado

2.2.11. entradaEn

Su función es leer el fichero del mensaje a codificar, a partir del path pasado por parámetro, y devolver su contenido.

2.2.12. entradaDe

Su función es leer el fichero del mensaje a decodificar, a partir del path pasado por parámetro, convertir su contenido en un string binario y devolverlo.

2.2.13. salidaEn

Su función es introducir el código codificado en un fichero, el cual su dirección es indicada por un string pasado por parámetro y que se convierte de string binario a bytes.

2.2.14. salidaDe

Su función es introducir el mensaje descodificado en un fichero, el cual su dirección es indicada por un string pasado por parámetro.