UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de sistemas informáticos y computación

Computación natural

Búsqueda de tándems de repetición en secuencias genómicas

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor
Francisco Javier Gil-Terrón Rodríguez

2021 - 2022

El trabajo a desarrollar seleccionado de entre los propuestos en la asignatura ha sido el de búsqueda de tándems de repetición en secuencias genómicas, el cuál será desarrollado en un archivo de Jupyter Notebook que será adjuntado con este mismo documento.

Este notebook consta de tres secciones: Definición de funciones, pruebas realizadas sobre estas funciones, y cadenas de ADN como lenguajes de duplicación con lazos de compensación, que se tratará con más detalle al final del documento.

La primera de las funciones desarrolladas consiste simplemente en un generador de cadenas aleatorio que dada una longitud de entrada devuelve una cadena de esa longitud de nucleótidos codificados con su base nitrogenada (A – Adenina, G – Guanina, C – Citosina y T – Timina). Esta función ha sido creada para poder trabajar con comodidad sobre cadenas de distintos tamaños, pero el programa funciona correctamente con cualquier cadena introducida por el usuario.

Después de esta, se desarrollan las tres funciones principales del programa: repeticion_tandem_especifica(), todo_tandem_posible() y tandems_longitud(). Cada una de estas funciones se comporta de la siguiente manera.

La primera busca un tándem de repetición específico en una cadena y si lo encuentra devuelve cuantas ocurrencias seguidas hay de la sub-cadena de entrada y su posición por cada vez que se produzca el tándem de repetición. Es decir, dada una cadena y una sub-cadena de entrada, se buscará a lo largo de la cadena todas las veces que la sub-cadena se encuentra repetida, por lo que se considerará que se produce un tándem de repetición con esa sub-cadena cuando al menos haya dos ocurrencias seguidas de esa sub-cadena sobre la cadena original.

Por ejemplo, hacer la llamada a esta función sobre la cadena ‘AAGTCAAA’ y siendo la búsqueda la sub-cadena ‘A’, el programa devolvería lo siguiente:

Tándem A
2 ocurrencias en la posición 0
3 ocurrencias en la posición 5

Mientras que si la búsqueda sobre esa cadena fuera la sub-cadena ‘AA’ no se devolvería nada, ya que no hay dos o más ocurrencias seguidas de ‘AA’ en la cadena.

Cabe destacar que se ha hecho hincapié en el rendimiento del programa, de manera que la forma de buscar una sub-cadena en otra cadena se ha llevado a cabo haciendo uso de la función split() y calculando las longitudes de los resultados de este split(), que es mucho más eficiente que realizar comparaciones.

Por ejemplo, el resultado de hacer ('CGTAAG').split('A') devolvería la siguiente lista ['CGT','','G'], cuyas longitudes serían [3,0,1], a partir de lo que se puede obtener que se producen dos ocurrencias de 'A' (ya que hay una longitud 0) en la posición 3.

Las otras dos funciones, todo_tandem_posible() y tandems_longitud() respectivamente, busca y devuelve todos los tándems de repetición posibles de cualquier longitud en una cadena, y busca y devuelve todos los tándems de repetición de una determinada longitud en una cadena. De esta manera, usando estas dos funciones en conjunto con la anterior, podemos obtener a partir de una cadena todos los tándems de repetición de cualquier longitud existentes en una cadena, todos los tándems de repetición de una longitud determinada, o bien un tándem de repetición especificado por el usuario (en caso de no existir, no devolvería nada el programa)

La última de las funciones sería un 'main' que invoca a las anteriormente expuestas en función de los argumentos que se le dé. En la sección de pruebas se ha experimentado con estos tres tipos de llamadas a la función, es decir, con las tres funciones principales que se han comentado con anterioridad.

Además, en la última sección del Notebook, Cadenas de ADN como lenguajes de duplicación con lazos de compensación, se han tratado de demostrar de forma práctica algunas de las propiedades vistas durante las sesiones de la asignatura.

Para la primera de estas propiedades, que para cualquier alfabeto arbitrario V y cualquier cadena w perteneciente a V^+, $h(D^*_{cl}(w))$ es regular, se han obtenido las expresiones regulares que denotan $h(D^*_{cl}(w))$ para todas las permutaciones posibles de las cuatro bases nitrogenadas que componen los nucleótidos.

En el caso de la segunda propiedad, que se compone de dos afirmaciones, la primera de ellas, que para cualquier alfabeto V con card(V)=1 y cualquier w perteneciente a V^+, $h(D^*_{cl}(w)) = D^*(w)$, es trivial tal como se vio en clase, dado que si $w = a^n$ entonces, $h(D^*_{cl}(a)) = D^*(a^n) = a^n a^+$.

Sin embargo, no lo es la segunda afirmación, que sostiene que para cualquier alfabeto V con card(V)>1 existe *w* perteneciente a V^+ tal que $h(D^*_{cl}(w))$ no contiene $D^*(w)$.

Para demostrarlo se ha generado la cadena 'CGTCGTGTCGTA' a partir de secuencias de duplicación de 'CGTA' cuya $h(D^*_{cl}(CGTA))$ se puede denotar con la expresión regular $CC^*G(G+CG)^*T(T+GT+CGT)^*A(A+TA+GTA+CGTA)^*$ y que no contiene a la cadena generada con secuencias de duplicación, es decir, que la cadena 'CGTCGTGTCGTA' generada con secuencias de duplicación de CGTA no pertenece al lenguaje $h(D^*_{cl}(CGTA))$.