Parte de Algoritmos, de la asignatura de Programación
Máster de Bioinformática

Cadenas y expresiones regulares

Domingo Giménez Cánovas

Departamento de Informática y Sistemas
Universidad de Murcia
http://dis.um.es/~domingo/algbio.html
domingo@um.es

Contenido

Definición

Una **cadena de caracteres** (*string*) es una secuencia ordenada de longitud arbitraria (finita) de elementos que pertenecen a un alfabeto.
Los **caracteres** pueden ser letras, números u otros símbolos.
Normalmente se almacenan en un vector (un *array* de una dimensión).
Las operaciones más comunes son:

- **Asignación** de una cadena a otra. ($S \rightarrow S$, con S el conjunto de strings del alfabeto con el que se trabaja)
- **Concatenación**: formar una cadena añadiendo otra al final. ($S \times S \rightarrow S$)
- **Búsqueda** de carácter. ($S \times C \rightarrow N$, con C el conjunto de caracteres del alfabeto)
- **Búsqueda** de subcadena. ($S \times S \rightarrow N$)
- **Extracción**: quitar de una cadena una parte a partir de la posición que se indica. ($S \times N \times N \rightarrow S$)
- **Comparación**. ($S \times S \rightarrow Bool$)
- **Longitud**.($S \rightarrow N$)

Cuestiones

→ Hacer un programa que pregunte en un bucle mientras que no se quiera acabar el tipo de operación de las anteriores que queremos hacer, pida los datos de entrada a la función y escriba el resultado. Se trabajará directamente con arrays de caracteres, sin utilizar las funciones de Perl para trabajar con strings.
→ Con las operaciones vistas hasta ahora de Perl, ¿cuáles de las operaciones anteriores se simplifican?, ¿cómo?
→ ¿Qué otras operaciones sobre cadenas se pueden definir? Añadir alguna de ellas al programa anterior.

Funciones con cadenas en Perl

- El operador `.` (punto) se usa para concatenar cadenas.
- `chomp(CADENA)`. Elimina el último carácter de `CADENA`. `chop(CADENA)` hace lo mismo pero además devuelve el carácter eliminado.
- `chr(NUMERO)`. Devuelve el carácter que tiene NUMERO como valor ASCII.
- `index(CADENA,SUBCADENA,POSICION)`. Devuelve la posición de la primera aparición de `SUBCADENA` en la `CADENA`, empezando en `POSICION`, si no se da `POSICION` la búsqueda se hace desde el principio de `CADENA`. `rindex` hace lo mismo desde el final.
- `join(CADENA,ARRAY)`. Devuelve una cadena formada por todos los elementos del `ARRAY` unidos por la `CADENA`.

Funciones con cadenas en Perl II

- `lc(CADENA)`. Devuelve la `CADENA` en minúsculas. `lcfirst(CADENA)` devuelve la `CADENA` con la primera letra en minúscula. `uc` y `ucfirst` hacen lo mismo pero convirtiendo a mayúscula.
- `length(CADENA)`. Devuelve la longitud de la `CADENA`.
- `split(PATRON,CADENA,LIMITE)`. Divide una `CADENA` de acuerdo a un `PATRON`, si se asigna a un array devuelve todos los elementos resultado de la división, si se asigna a un escalar devuelve el número de partes en que se divide.
- `substr(CADENA,DESPLAZAMIENTO,LONGITUD)`. Devuelve la porción de la cadena entre `DESPLAZAMIENTO` y `LONGITUD`. Si no se da `LONGITUD` llega hasta el final de `CADENA`. Con un `DESPLAZAMIENTO` negativo se empieza por la derecha de la `CADENA`.

→ Hacer programas para comprobar el funcionamiento de las funciones que no se comprendan.

Conceptos

Las expresiones regulares se usan para representar, buscar, hacer cambios... en strings.
Por ejemplo, con `$dna =~ /CT[CGT]ACG/` se comprueba si la variable `dna` está compuesta por seis caracteres, los dos primeros `CT`, los tres últimos `ACG`, y el tercero `C`, `G` o `T`.

En las expresiones regulares se usan caracteres especiales que se llaman **metacaracteres**. Por ejemplo:

- `*` representa varias repeticiones del carácter que lo precede.
- `[ab]` representa una aparición del carácter `a` o del `b`.

Enzimas de restricción

Cortan una cadena de ADN donde aparece una cadena.
Se conocen unas mil. Por ejemplo:

- `EcoRI`: `G^AATTC`
- `HindIII`: `A^AGCTT`

Queremos encontrar en una cadena de ADN las posiciones donde aparece una cadena de restricción (**mapa de restricción**).

Ficheros de enzimas

Podemos tener la enzimas almacenadas en ficheros con un cierto formato, por ejemplo:

```
REBASE version 104
bionet.104
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
REBASE, The Restriction Enzyme Database
http://rebase.neb.com
Copyright (c) Dr. Richard J. Roberts, 2001.
All rights reserved.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Rich Roberts Mar 30 2001

AaaI (XmaIII)    C^GGCCG
...
AarI    CACCTGCNNNN^
...
```

Lectura de las enzimas

- Hay que pasar todas las líneas de cabecera (hasta llegar a la que tiene `Rich Roberts`) y las en blanco.
- Por cada enzima tenemos: el nombre, posiblemente entre paréntesis otro nombre, y la cadena.
- En la cadena tenemos los cuatro caracteres básicos (`A`, `C`, `G`, `T`) y otros que representan varios caracteres básicos, que habrá que convertir en cadenas de caracteres básicos.

Eliminación de líneas (ejemplo 9.2)

```
# Discard header lines
( 1 .. /Rich Roberts/ ) and next;

# Discard blank lines
/^\s*$/ and next;
```

Con el operador `..` se determina un rango. En este caso las líneas de la 1 a la que contiene `/Rich Roberts/`, lo que produce que se salten los pasos correspondientes del bucle `foreach ( @rebasefile )`.
Con el operador `and` se evalúa la expresión de la izquierda, y si es verdad se ejecuta la de la derecha (`next`).

Obtención de enzima y cadena (ejemplo 9.2)

Se genera un array con los campos de cada línea separados por espacio en blanco:

```
my @fields = split( " ", $_)
```

Se guarda el nombre de la enzima (sin la posible alternativa en paréntesis):

```
$name = shift @fields
```

Y se obtiene la cadena:

```
$site = pop @fields
```

→ Consultar otras funciones sobre cadenas (`defined`, `undef`, `join`, `reverse`, `scalar`...)

Conversión de cadenas (ejemplo 9.1)

La función del ejemplo 9.1 recibe una cadena y usa una tabla hash

```
my %iub2character_class = (
A => 'A',
...
R => '[GA]',
...
);
```

para obtener la cadena con caracteres básicos.
Obtiene el carácter en cada posición, lo transforma, y concatena el resultado en la nueva cadena:

```
$regular_expression
.= $iub2character_class{substr($iub, $i, 1)};
```

Ejemplo 9.3

Hace el trabajo completo: lee fichero de cadena de ADN y fichero de enzimas, pide la enzima a buscar y devuelve las posiciones.
Utiliza una tabla hash con un primer campo el nombre de la enzima, y otro campo formado por dos: la cadena original y la expresión regular con caracteres básicos.
Cuando identifica una expresión regular, la variable $& contiene la cadena con la que ha coincidido, la $` contiene los caracteres de la cadena hasta la coincidencia, y $´ los caracteres posteriores a la coincidencia.

→ Hacer funcionar el programa, y modificarlo para que tome de un fichero las enzimas para las que quiere obtener el mapa, y genere los mapas en un fichero con formato:

```
ADN
enzima1 mapa_enzima_1
enzima2 mapa_enzima_2
...
```