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

# Asociaciones y tablas hash

Domingo Giménez Cánovas

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

Asociaciones y tablas hash

# Contenido

# Conceptos generales

Frecuentemente es necesario establecer asociaciones entre datos, por ejemplo:

- En un array o una cadena hay una asociación entre la posición y el dato almacenado.
- En un diccionario se asocia a una palabra su definición.
- Puede haber asociaciones múltiples, como el nombre de una persona, que tiene asociado el DNI y la dirección.

Cuando se implementa una asociación hay que implementar también las operaciones que se realizan sobre ella:

- Insertar una nueva asociación.
- Modificar una asociación.
- Eliminar una asociación...

# Conjuntos y Diccionarios

Un **conjunto** es una colección no ordenada de elementos.
Un **elemento** puede ser un elemento primitivo o compuesto, como otro conjunto, y suelen ser del mismo tipo.
Cuando puede haber elementos repetidos se habla de **bolsas**.
Las **operaciones** más frecuentes **entre conjuntos** son: Unión, Intersección, Diferencia, Anular, Asignar, Igual?.
Las **operaciones** más frecuentes **entre elementos y conjuntos**: Miembro?, Inserta, Suprime.
Un **diccionario** es un conjunto de asociaciones, con las operaciones Inserta, Suprime, Miembro y Anula.

# Cuestiones

→ Dados los caracteres A, C, G y T, ¿Cuántos conjuntos distintos podemos formar con ellos?
→ Representamos uno de estos conjuntos con cuatro valores 0 o 1, con 0 indicando que el carácter no está en el conjunto y 1 que sí pertenece. Hacer un programa que pregunte en un bucle mientras que no se quiera acabar el tipo de operación que queremos hacer (Miembro?, Inserta y Suprime), la haga y escriba el conjunto después de cada operación.

# Tablas hash (o de dispersión)

El trabajo sobre conjuntos o diccionarios con listas o arrays tiene un tiempo lineal: para encontrar un elemento hay que recorrer la estructura.

Con **tablas hash** se usa un array de $B$ posiciones, y dada una clave $x$ se obtiene su posición mediante una función de dispersión: $h(x) \in (0, \ldots, B-1)$.

Se presentan problemas de: colisión, llenado de la tabla...

No veremos cómo resolver estos problemas (se puede consultar la bibliografía no básica) pero veremos la estructura correspondiente en Perl.

# Hashes

- Las variables de tipo **hash** en Perl empiezan con %: `%english_dictionary`.
- Se puede acceder al valor asociado a una clave: `$definition = $english_dictionary{'recreant'}`
- o asociar un valor a una clave: `$english_dictionary{'recreant'} = "One who calls out in surrender"`
- Se puede obtener un array de todas las claves: `@keys = keys %my_hash`
- o de todos los valores: `@values = values %my_hash`
- y comprobar si hay una clave: `defined $myhash{'mykey'}`
- o ordenar las claves: `@sorted_keys = sort keys %my_hash`

# Hashes (II)

Se inicializan de forma similar a los arrays:

```
%classification = (
    'dog',        'mammal',
    'robin',     'bird',
    'asp',        'reptile',
);
```

o en forma más clara:

```
%classification = (
    'dog'    =>    'mammal',
    'robin'  =>   'bird',
    'asp'     =>   'reptile',
);
```

# Código genético

| First Position | Second Position: U | | Second Position: C | | Second Position: A | | Second Position: G | | Third Position |
|---|---|---|---|---|---|---|---|---|---|
| U | UUU | Phe | UCU | Ser | UAU | Tyr | UGU | Cys | U |
| | UUC | | UCC | | UAC | | UGC | | C |
| | UUA | Leu | UCA | | UAA | Stop | UGA | Stop | A |
| | UUG | | UCG | | UAG | Stop | UGG | Trp | G |
| C | CUU | Leu | CCU | Pro | CAU | His | CGU | Arg | U |
| | CUC | | CCC | | CAC | | CGC | | C |
| | CUA | | CCA | | CAA | Gln | CGA | | A |
| | CUG | | CCG | | CAG | | CGG | | G |
| A | AUU | Ile | ACU | Thr | AAU | Asn | AGU | Ser | U |
| | AUC | | ACC | | AAC | | AGC | | C |
| | AUA | | ACA | | AAA | Lys | AGA | Arg | A |
| | AUG | Met (start) | ACG | | AAG | | AGG | | G |
| G | GUU | Val | GCU | Ala | GAU | Asp | GGU | Gly | U |
| | GUC | | GCC | | GAC | | GGC | | C |
| | GUA | | GCA | | GAA | Glu | GGA | | A |
| | GUG | | GCG | | GAG | | GGG | | G |

# Uso de condicionales (página 177 PERL)

```
sub codon2aa {
    my($codon) = @_;
    if ( $codon =~ /TCA/i )    { return 'S' } #Serine
    elsif ( $codon =~ /TCC/i ) { return 'S' } #Serine
    ...
    else {
        print STDERR "Bad codon \"$codon\"!!\n";
        exit;
    }
}
```

Hay redundancia, y se pueden usar expresiones regulares para eliminarla.

# Condicionales y expresiones regulares (página 180 PERL)

```
sub codon2aa {
    my($codon) = @_;
    if ( $codon =~ /GC./i)    { return 'A' } #Alanine
    elsif ( $codon =~ /TG[TC]/i) { return 'C' } #Cysteine
    ...
    elsif ( $codon =~ /TT[AG]|CT./i) { return 'L' }
    ...
    }
}
```

Más compacto pero siguen evaluándose condiciones hasta llegar a la que buscamos.

# Uso de hash (página 182 PERL)

```
sub codon2aa {
    my($codon) = @_;
    $codon = uc $codon; #Transformar a mayusculas
    my(%genetic_code) = (
    'TCA' => 'S',
    'TCC' => 'S',
    ...
    );
    if(exists $genetic_code{$codon}) {
        return $genetic_code{$codon};
    }else{
        print STDERR "Bad codon \"$codon\"!!\n";
        exit;
    }
}
```

¿Qué hace `exists`?, ¿y `STDERR`?

# Transformación de ADN a proteínas. Ejemplo 8.1 de PERL)

Se hace la transformación llamando a la rutina anterior.
→ Analizar el ejemplo 8.1 e identificar elementos nuevos (`substr`...)
→ Transformar el programa en una rutina, y comprobar que funciona correctamente llamando a ella desde un programa.

# Ficheros en bioinformática

Hay ficheros con bancos de datos en distintos formatos.
Los más usuales son:

- FASTA. Lo usan los programas Basic Local Alignment Search Technique (BLAST).
  Contiene ADNs, con una o varias líneas (que empiezan con >) cabecera:

  ```
  > sample dna | (This is a typical fasta header.)
  agatggcggcgctgaggggtcttgggggctctaggccggccacctactgg
  tttgcagcggagacgacgcatggggcctgcgcaataggagtacgctgcct
  ```

- GenBank (Genetic Sequence Data Bank). Es una colección de datos genéticos públicos.
  Contiene mucha más información además de los ADNs.

# Lectura de un fichero en formato FASTA ( página 192 de PERL)

- Con `open(GET_FILE_DATA, $filename)` se abre un fichero de nombre `$filename` y se le asocia el identificador `GET_FILE_DATA`.
- `open` devuelve un código (verdadero si se ha podido abrir y falso si no), y con `unless` de evalúa su valor.
- Una vez abierto, se lee su contenido en una cadena con `@filedata = <GET_FILE_DATA>`
  Si el fichero es muy grande será preferible no copiar todo su contenido en una cadena, sino ir leyendo trozos del fichero, quizás línea a línea o cada ADN por separado.

# Tratamiento de los datos

- Una vez los datos del fichero en una cadena, se puede eliminar las líneas cabecera, vacías, con comentarios... para quedarnos con el ADN (página 193 PERL),
- y escribir la salida con un cierto formato (ejemplo 8.2 PERL), para lo que se utilizan las subrutinas anteriores,
- o escribirlo transformado a proteínas (ejemplo 8.3 PERL), para lo que se usan las subrutinas que trabajan con hash.

# Modificaciones

→ Modificar las rutinas correspondiente para que se lean los datos de un fichero en formato FASTA, un ADN cada vez, se transforme en proteína y se guarde en un fichero. El fichero resultado tendría los ADN del original transformados a proteínas, alguna forma de separación entre proteínas (puede ser una línea en blanco, > o la misma línea que había en el fichero original).
→ Dado que los codons están formados por tres caracteres y no se sabe dónde empiezan, se pueden transformar a proteínas empezando en distintas posiciones. Analizar cómo se hace en el ejemplo 8.4 de PERL. Modificar el programa para que guarde en un fichero de salida, para cada ADN en el fichero original, las seis secuencias de proteínas correspondientes, indicando para cada secuencia al número de ADN que corresponde y quÂº'e secuencia es.