
Nested Loops

```
while (condition) {  
    statements;  
}
```

But loops are statements, so it is possible to “nest” them:

```
while (condition) {  
    statements;  
    while (condition2) {  
        statements;  
    }  
    statements;  
}
```

Nested Loops: example

```
int row, column;
final int MAX_ROW = 6;

row = 1;
while (row <= MAX_ROW)
{
    column = 1;
    while (column <= row)
    {
        System.out.print("*");
        column++;
    }
    System.out.println();
    row++;
}
```

Case study: Bioinformatics and Computational Biology

- The use of computational techniques to solve problems in Biology such as
 - Small scale Biology:
 - * Analyzing DNA,
 - * Analyzing the structure of proteins,
 - Large scale Biology:
 - * Simulating eco-systems,
 - ...etc

DNA

- DNA is a large molecule encoding information about the structure and functions of organisms.
- DNA is made of two long chains or strings of molecules called *nucleotides*, which are twisted so it has an helix shape.
- There are four types of nucleotides, called Adenine, Cytosine, Guanine and Thymine.
- The two chains are complementary in the sense that
 - if there is Adenine in one chain, in the opposite chain there is Thymine in the same position, and viceversa, and
 - if there is Guanine in one chain, there is Cytosine in the opposite, and viceversa
 - For example
AGGTAC
TCCATG

Problem

Given a host DNA sequence and a gene sequence, find out if the gene occurs in the host or not, and if so, then say in which position.

For example, given a the host

AGGTACGCC

and the gene

ACG

we say that the gene does occur in the host at position 4 (counting from 0.)

But the gene

ATCA

does not occur in the host.

Analysis

- Input: two strings: the host, and the gene
- Output: a boolean which is true if the gene occurs in the host. If so, then we also produce a natural number which is the starting position of the gene in the host.
- Data representation: Abstraction
 - The internal chemical structure of A, T, G and C is irrelevant, so we *abstract* it.
 - We do not need to represent both chains, because they are redundant.
 - The shape of the molecules is also irrelevant for this problem, so we abstract it too.
 - The gene and the host are represented as strings,
 - The gene and the host are made up of only four characters: A, T, G, and C.
- If the gene is larger than the host, then obviously it cannot occur in it.

Design: General algorithm

Compare the gene with the host from left to right:

1. Compare the gene with the host starting at the first position of the host
2. If it matches then stop,
3. Otherwise, compare it, starting at the second position of the host
4. If it matches then stop,
5. Otherwise, compare it, starting at the third position of the host
6. ... etc.

Design: General algorithm (contd.)

0
AGGTACGCTAGGCA
TAGG

No match, so we move on...

01
AGGTACGCTAGGCA
TAGG

No match, so we move on...

012
AGGTACGCTAGGCA
TAGG

...

012345678
AGGTACGCTAGGCA
TAGG

Match!

Design: General algorithm (contd.)

```
          1111
01234567890123
AGGTACGCTATGCA
          TAGG
```

No match, so the gene doesn't occur in the host.

Design: A bit more precise...

1. Set *position* to 0
2. While *position* \leq length of the *host* - length of the *gene*, repeat:
 - (a) Compare the gene with the host starting from *position* in the host
 - (b) If the gene matches, then we found it, so stop looking
 - (c) Otherwise, increment *position* by 1 and continue

Design: Comparing the gene with the host

1. Compare the character *position* of the *host* with the first of the *gene*
2. If they are different then the gene doesn't match, so stop
3. Otherwise, compare the character *position+1* of the *host* with the second of the *gene*
4. If they are different then the gene doesn't match, so stop
5. Otherwise, compare the character *position+2* of the *host* with the third of the *gene*
6. ... etc
7. If we reach the end of the gene, then it matches

Design: Comparing the gene and the host (cont.)

0123
AGGTACGCTAGGCA
TAGG

We compare the first character of the gene...

3
AGGTACGCTAGGCA
TAGG
0

They match so we continue...

34
AGGTACGCTAGGCA
TAGG
01

They match so we continue...

345
AGGTACGCTAGGCA
TAGG
012

They don't match so we stop the comparison and continue where we left...

01234
AGGTACGCTAGGCA
TAGG

Design: A bit more precise...

1. Set *host_index* to *position*
2. Set *gene_index* to 0
3. Set *occurs* to true
4. While *host_index* < the length of the *host* and *gene_index* < the length of the *gene*, repeat:
 - (a) If the host nucleotide at *host_index* is different than the gene nucleotide at *gene_index*, then:
 - i. Set *occurs* to false, and
 - ii. stop testing this position
 - (b) Increment the *host_index* by 1 and the *gene_index* by 1

Design: Putting it all together

1. Set *occurs* to false
2. Set *position* to 0
3. While *position* < the length of the *host* - length of the *gene*, repeat:
 - (a) Set *host_index* to *position*
 - (b) Set *gene_index* to 0
 - (c) Set *occurs* to true
 - (d) While *host_index* < the length of the *host* and *gene_index* < the length of the *gene*, repeat:
 - i. If the host nucleotide at *host_index* is different than the gene nucleotide at *gene_index*, then:
 - A. Set *occurs* to false, and
 - B. stop testing this position
 - ii. Increment the *host_index* by 1 and the *gene_index* by 1
 - (e) If *occurs* is true then stop the main loop
 - (f) otherwise, increment the *position* by 1 and continue

Implementation

```
public class GeneFinder
{
    public static void main(String[] args)
    {
        String host, gene;
        boolean occurs;
        int position, host_index, gene_index;
        char host_nucleotide, gene_nucleotide;

        System.out.print("Enter a host DNA seq: ");
        host = Keyboard.readString();

        System.out.print("Enter a gene DNA seq: ");
        gene = Keyboard.readString();

        // Continues below...
```

Implementation (cont.)

```
occurs = false;
position = 0;
while (position <= host.length()
      - gene.length()) {
    host_index = position;
    gene_index = 0;
    occurs = true;
    while (host_index < host.length()
          && gene_index < gene.length()) {
        host_nucleotide = host.charAt(host_index);
        gene_nucleotide = gene.charAt(gene_index);
        if (gene_nucleotide != host_nucleotide) {
            occurs = false;
            break;
        }
        host_index++;
        gene_index++;
    } // End of inner while
    if (occurs) break;
    position++;
} // End of outer while
```

```
if (occurs) {
    System.out.println("The gene " + gene
        + " occurs at " + position
        + " in the host " + host);
}
else {
    System.out.println("The gene " + gene
        + " does not occur in the host " + host);
}
} // End of main
} // End of class GeneFinder
```