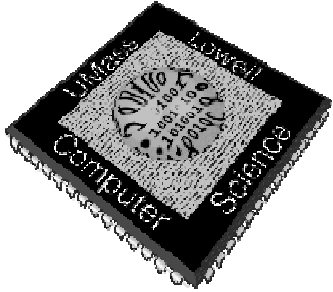


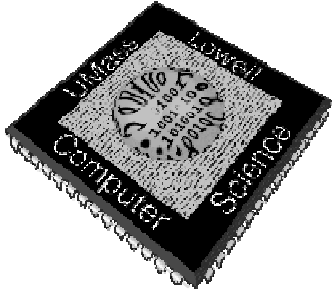
Analyzing Algorithms

Text
Chapters 2



Analyzing Algorithms

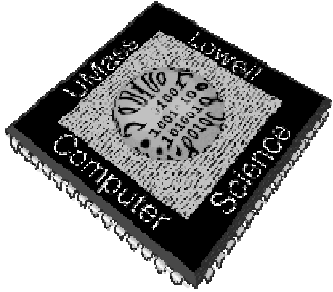
- goal: predicting resources that an algorithm requires
 - memory, communication bandwidth, hardware, **computational time**
 - compare several candidate algorithms, identify most efficient one



Analyzing Algorithms

- ❑ one-processor, random-access machine (RAM)
 - instructions executed one after another, no concurrent operations
 - common instructions:
 - arithmetic (+, -, *, /, %, [], [])
 - data movement (load, store, copy)
 - control (branch, subroutine call and return)
 - do not consider memory hierarchy

- ❑ elementary (primitive) operation: execution time can be bounded above by a constant depending only on the particular implementation—the machine, the programming language, etc.



Efficiency of an algorithm

□ Efficiency

- **Time**, space, energy
- Measured as a function of the size of the instances considered

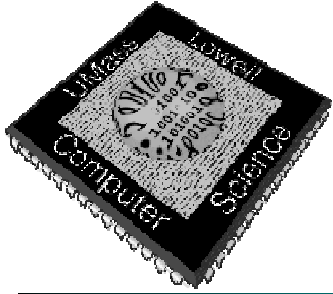
□ Input Size

- The *size* of an instance/input
 - ↗ corresponds formally the number of the bits needed to represent the instance on a computer
 - ↗ A less formal definition: any integer that in some way measures the number of components in an instance
 - ↗ For example, sorting, graphs
 - ↗ For problems involving integers, we use *value* rather than size

□ Running time

- The number of primitive operations executed in terms of input size.

□ mathematic tools include combinatorics, probability theory, identify most significant terms in a formula

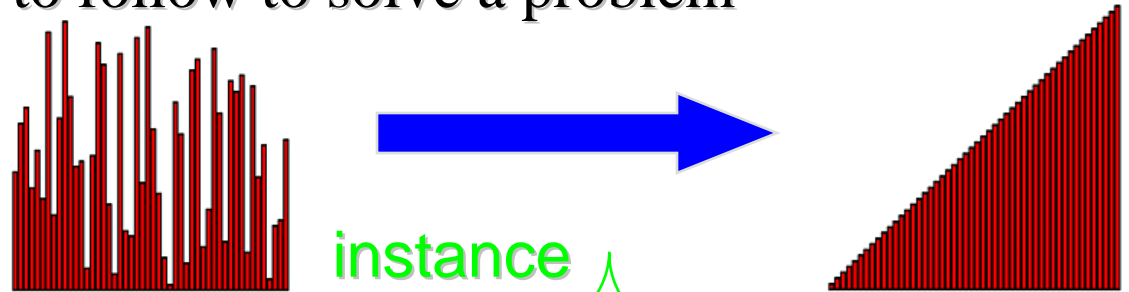


Sorting as Example

Algorithm:

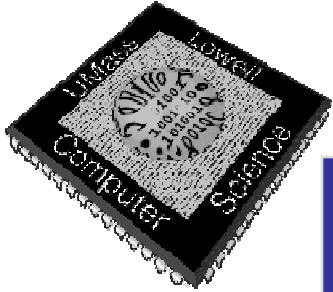
well-defined computational procedure that transforms input into output

steps for the computer to follow to solve a problem



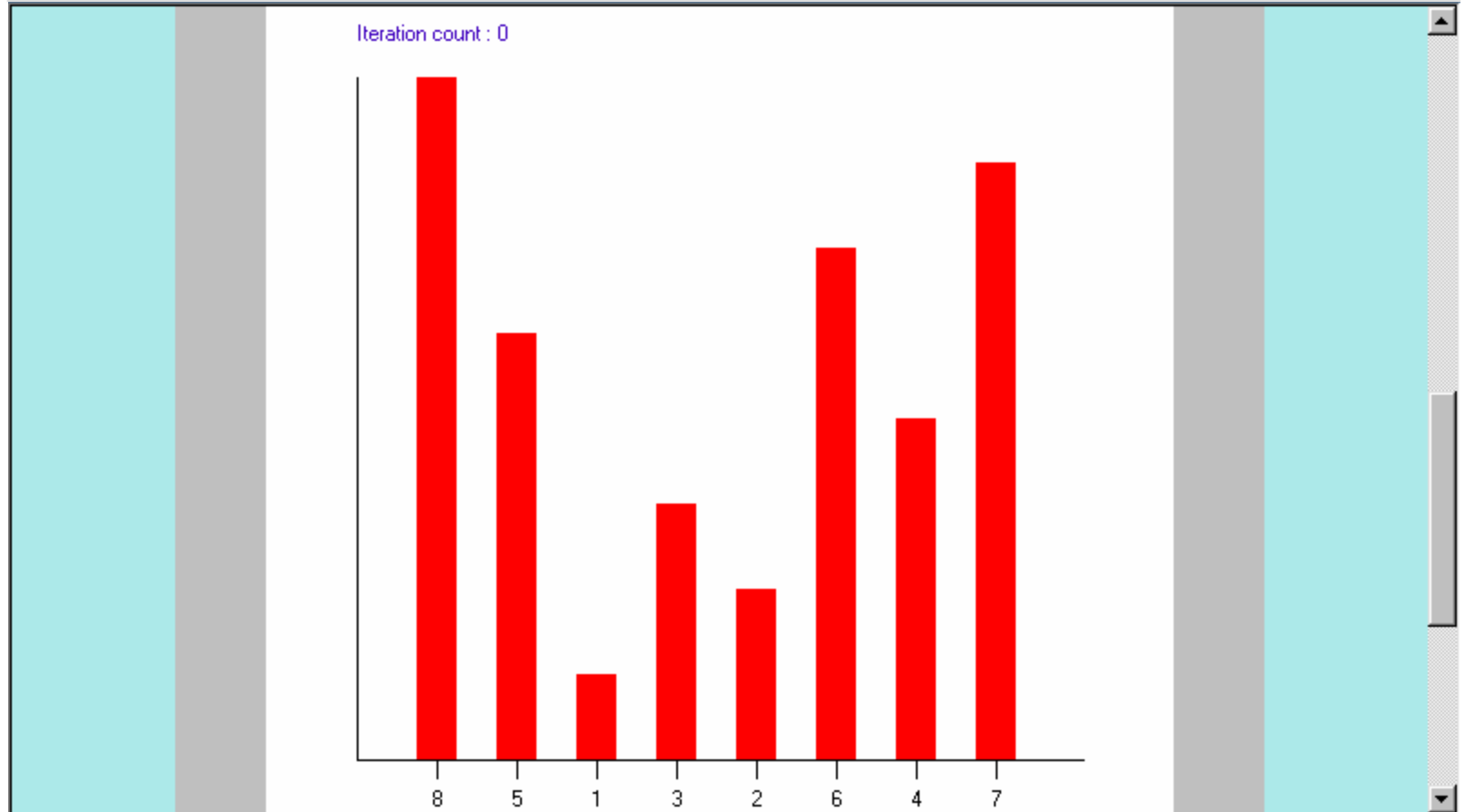
Sorting Problem:

- Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$
- Output: A permutation (reordering) $\langle a'_1, a'_2, \dots, a'_n \rangle$ of the input sequence such that: $a'_1 \leq a'_2 \leq \dots \leq a'_n$

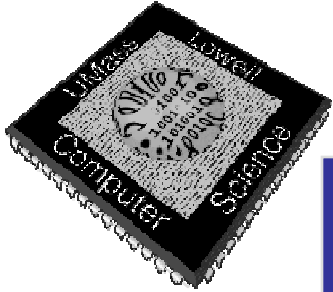


Insertion Sort Animation

Finding a place for item with value 5 in position 1:
Swap item in position 0 with item in position 1.

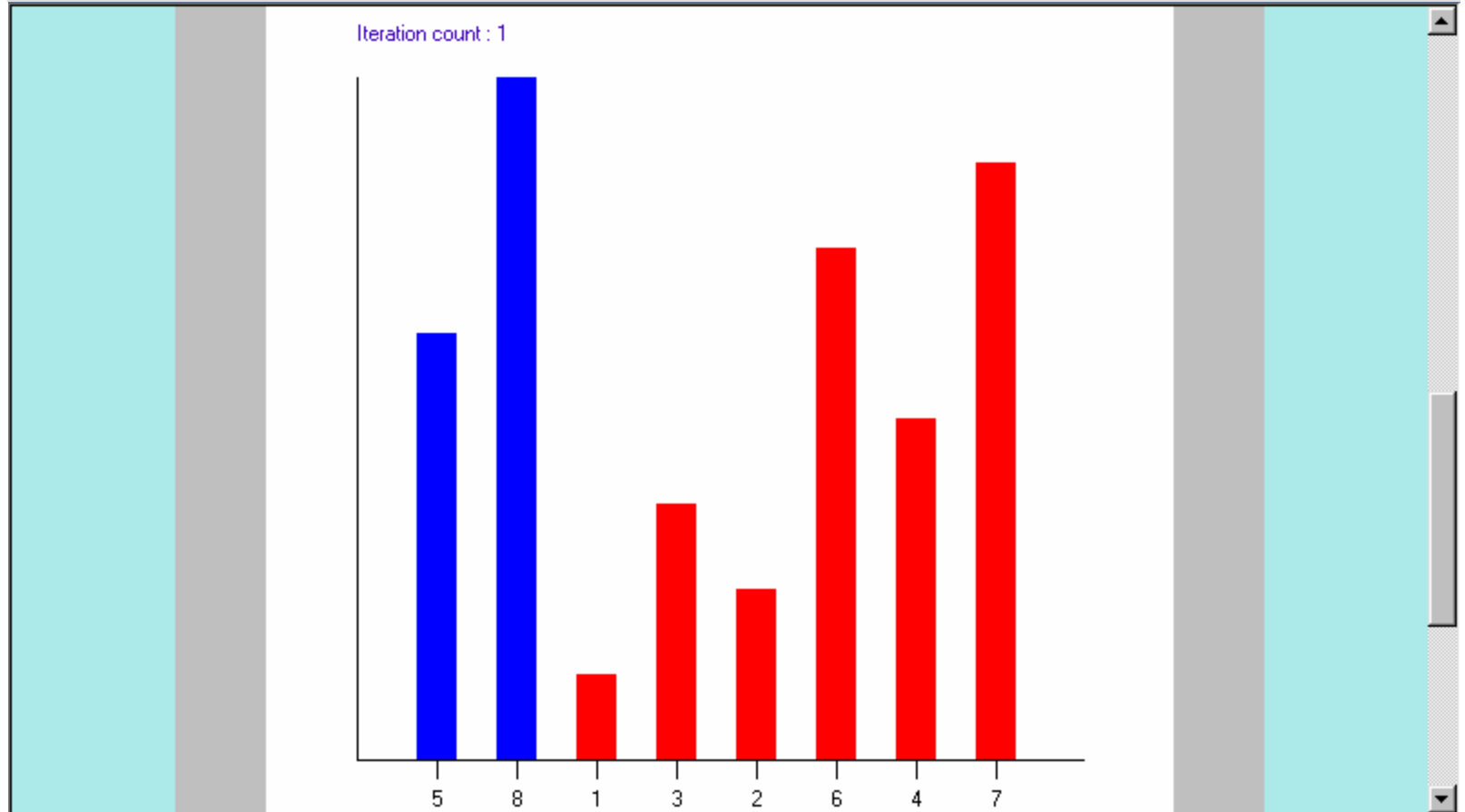


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

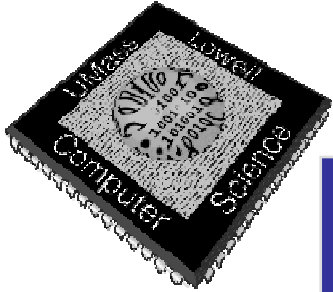


Insertion Sort Animation

Positions 0 through 1 are now in non-decreasing order.

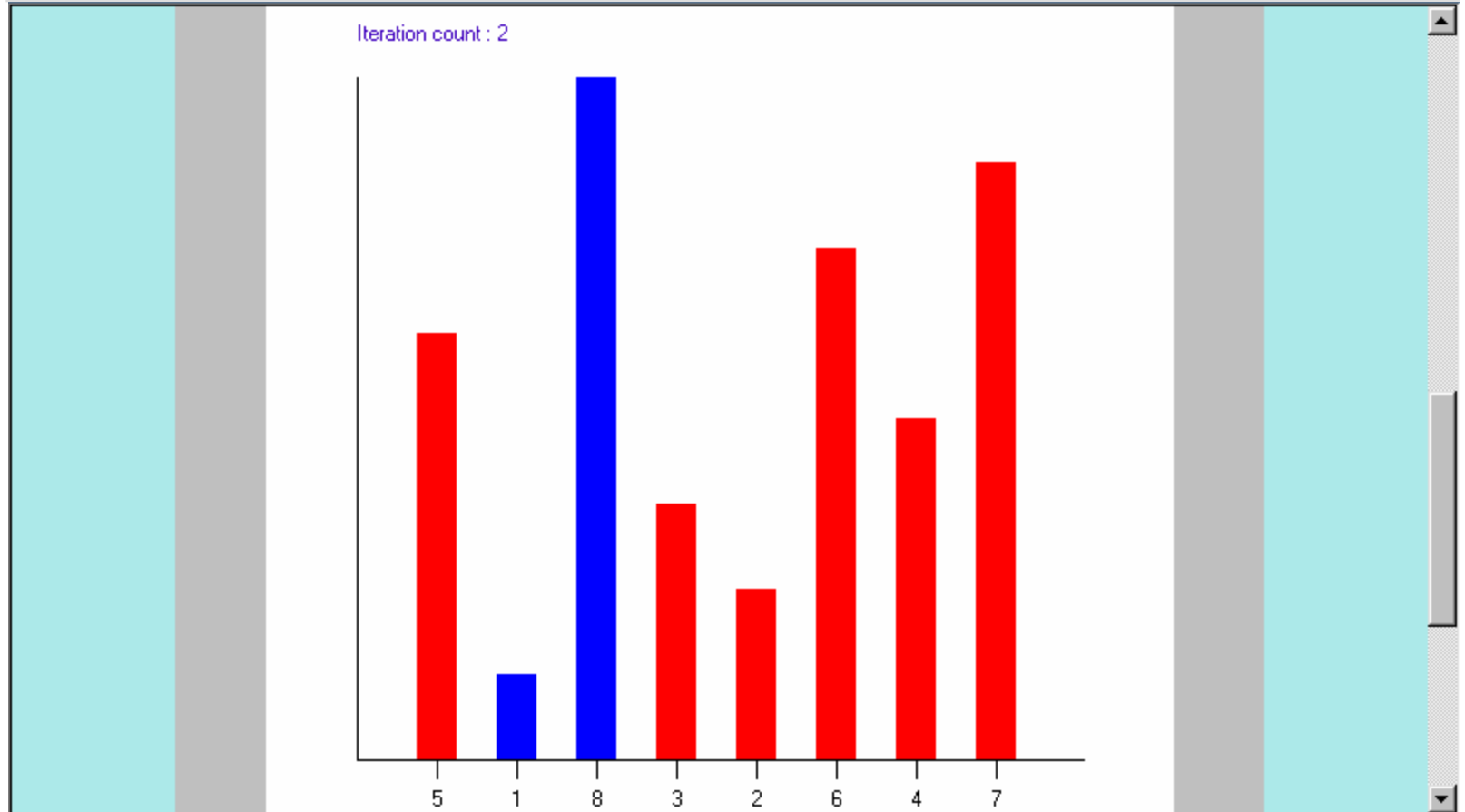


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

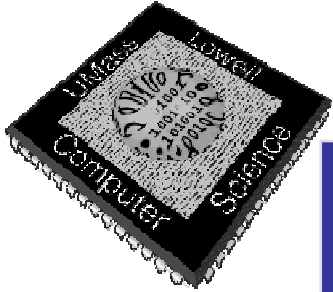


Insertion Sort Animation

Finding a place for item with value 1 in position 2:
Swap item in position 1 with item in position 2.

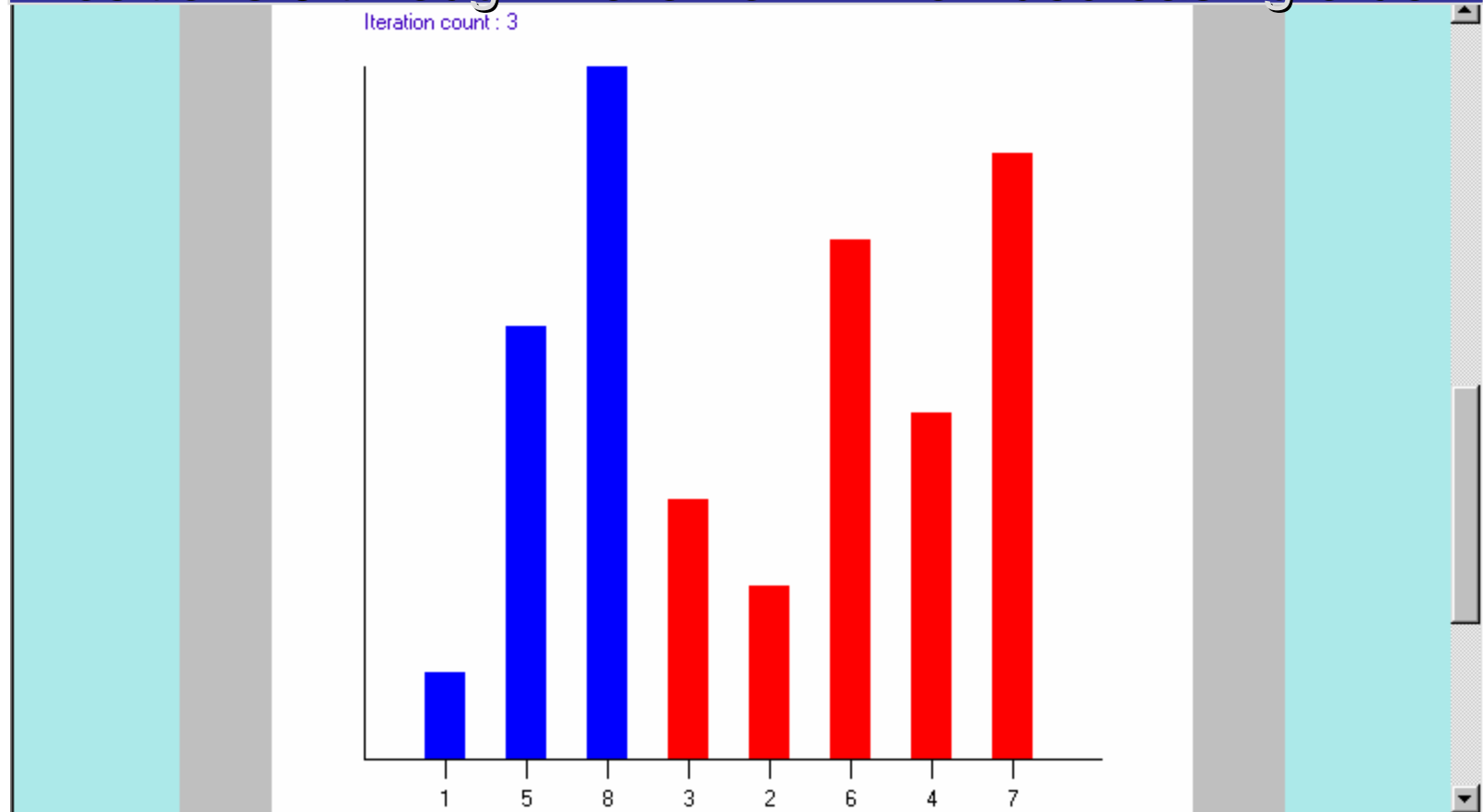


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



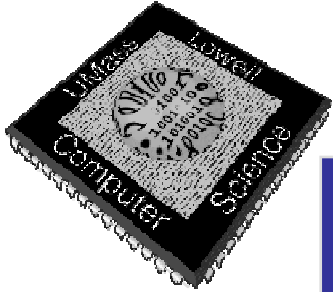
Insertion Sort Animation

Finding a place for item with value 1:
Swap item in position 0 with item in position 1.
Positions 0 through 2 are now in non-decreasing order.



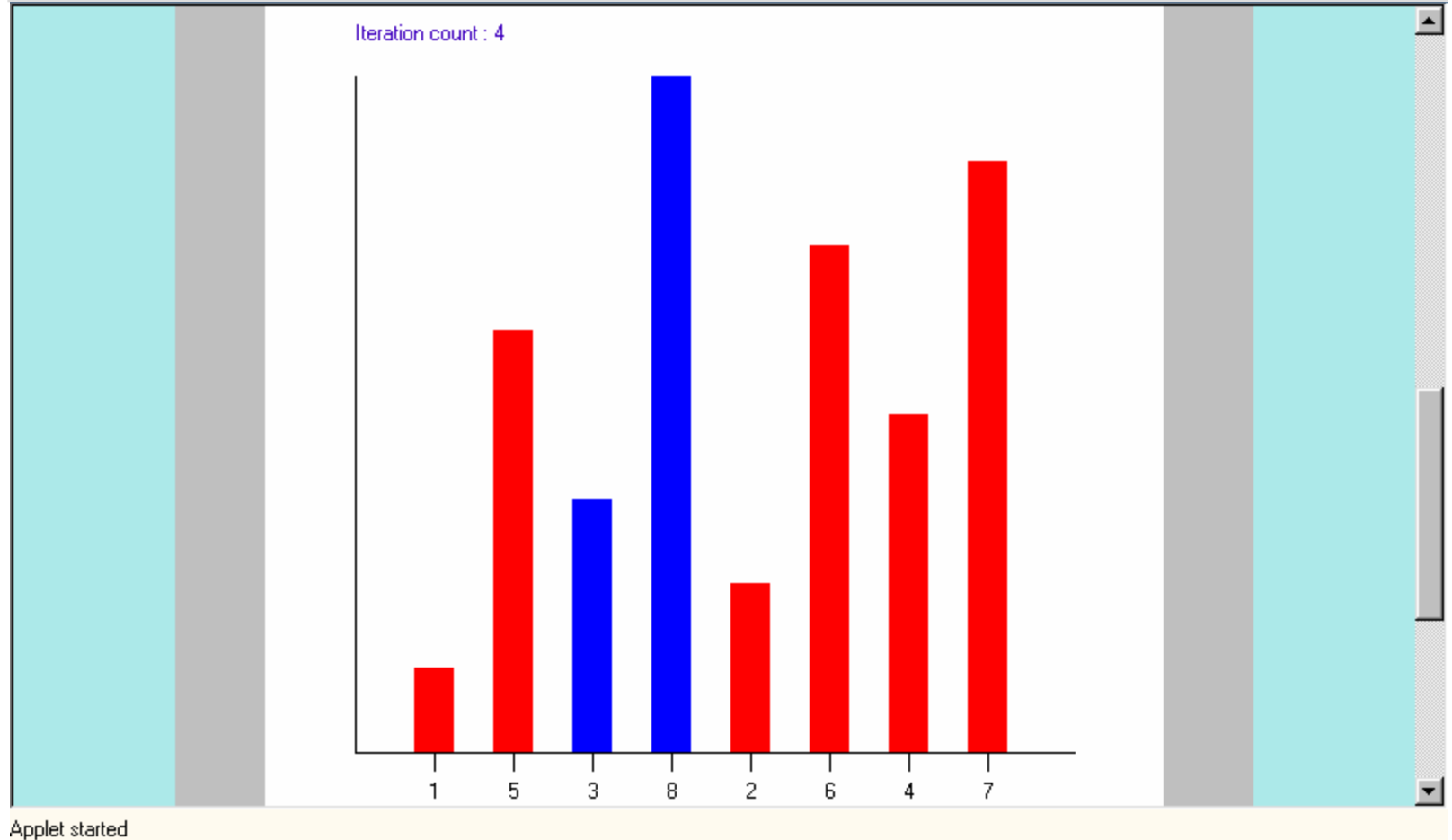
Applet started

<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

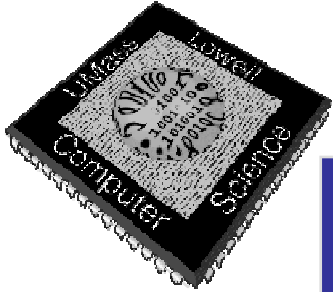


Insertion Sort Animation

Finding a place for item with value 3 in position 3:
Swap item in position 2 with item in position 3.

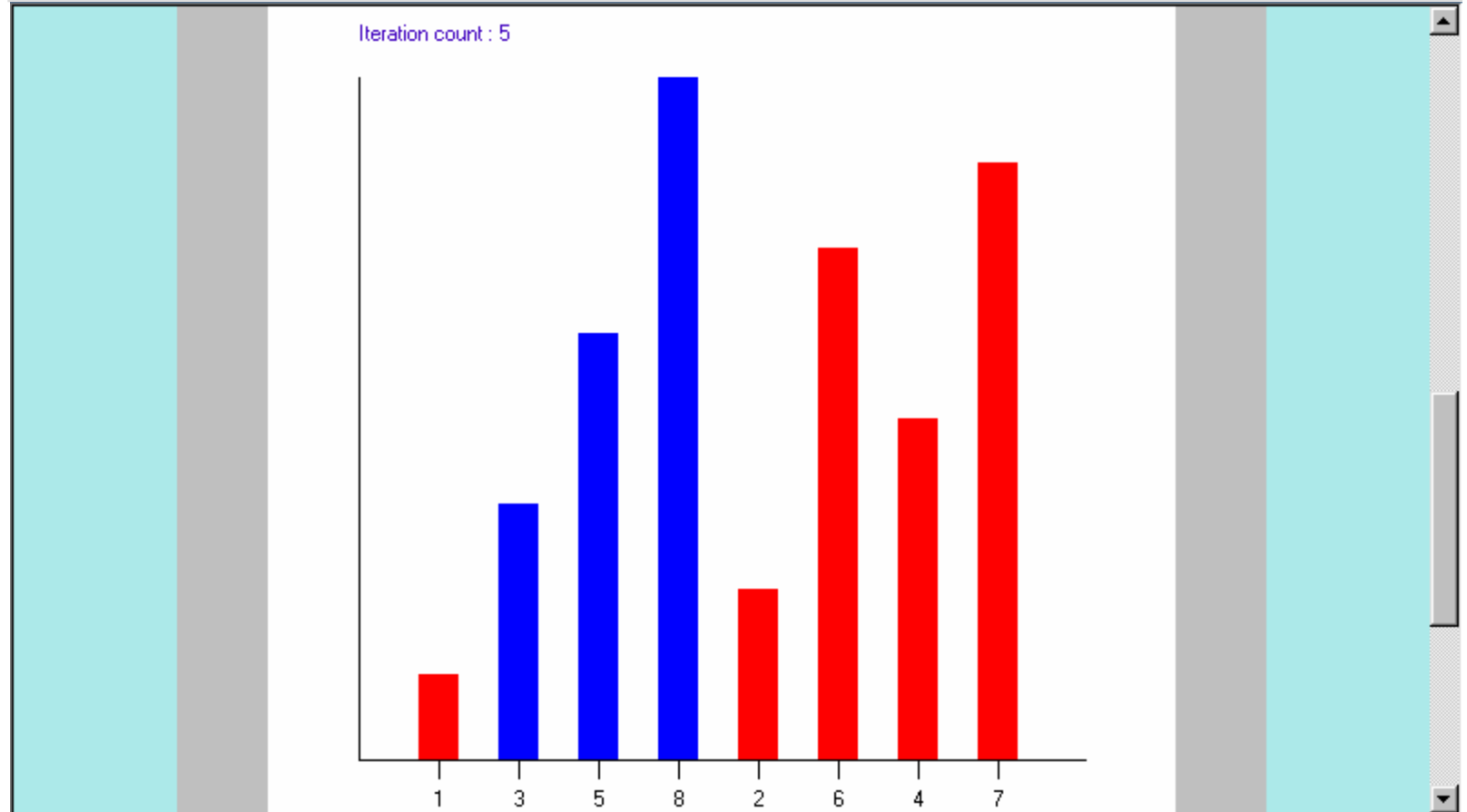


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



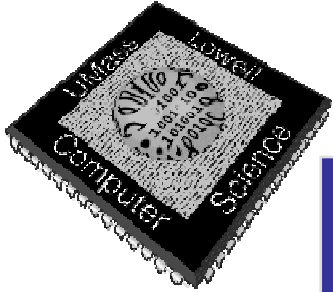
Insertion Sort Animation

Finding a place for item with value 3:
Swap item in position 1 with item in position 2.



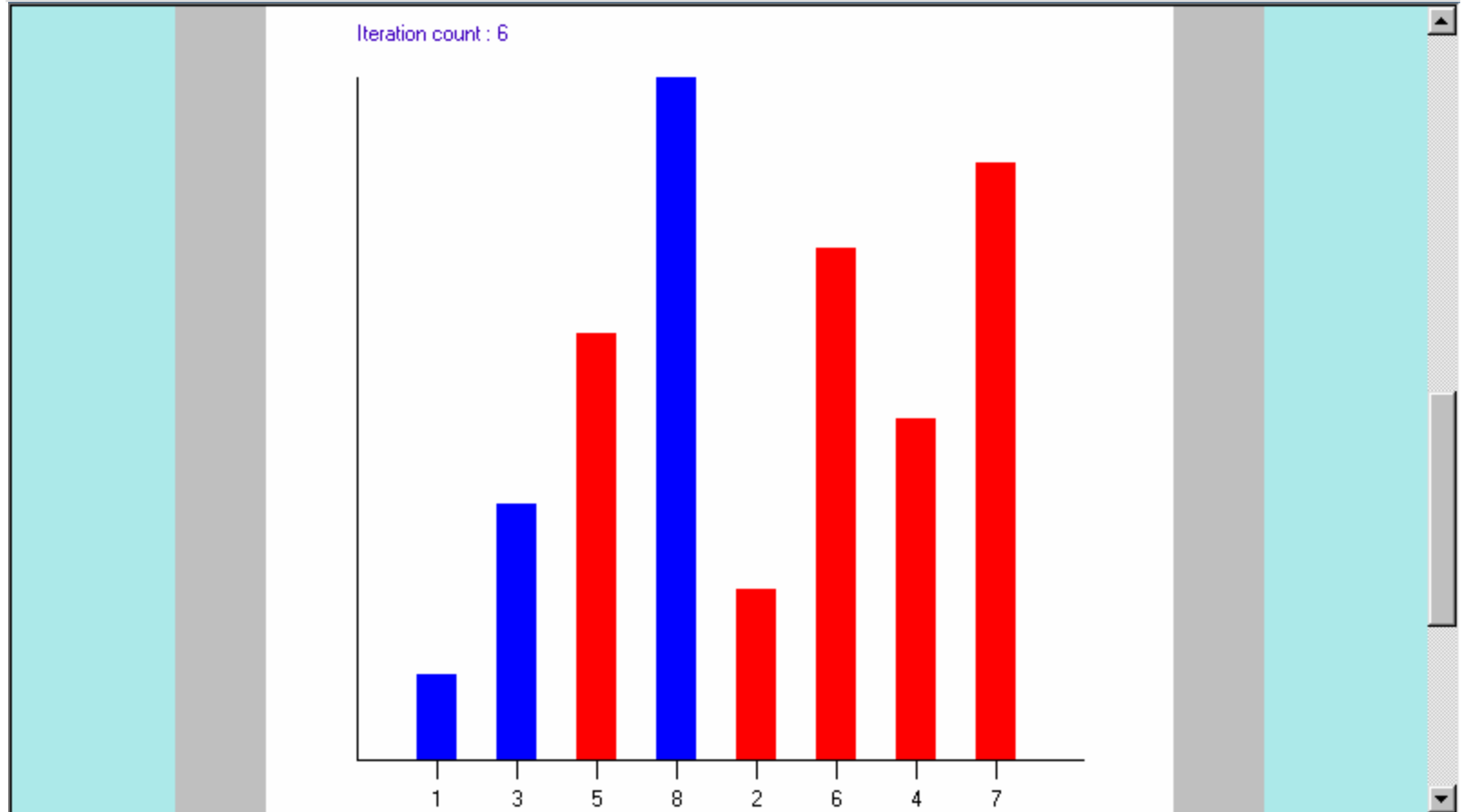
Applet started

<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

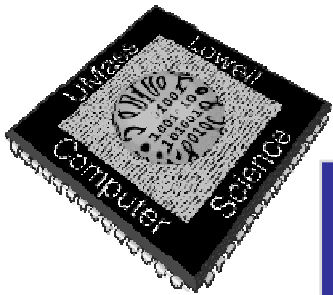


Insertion Sort Animation

Positions 0 through 3 are now in non-decreasing order.

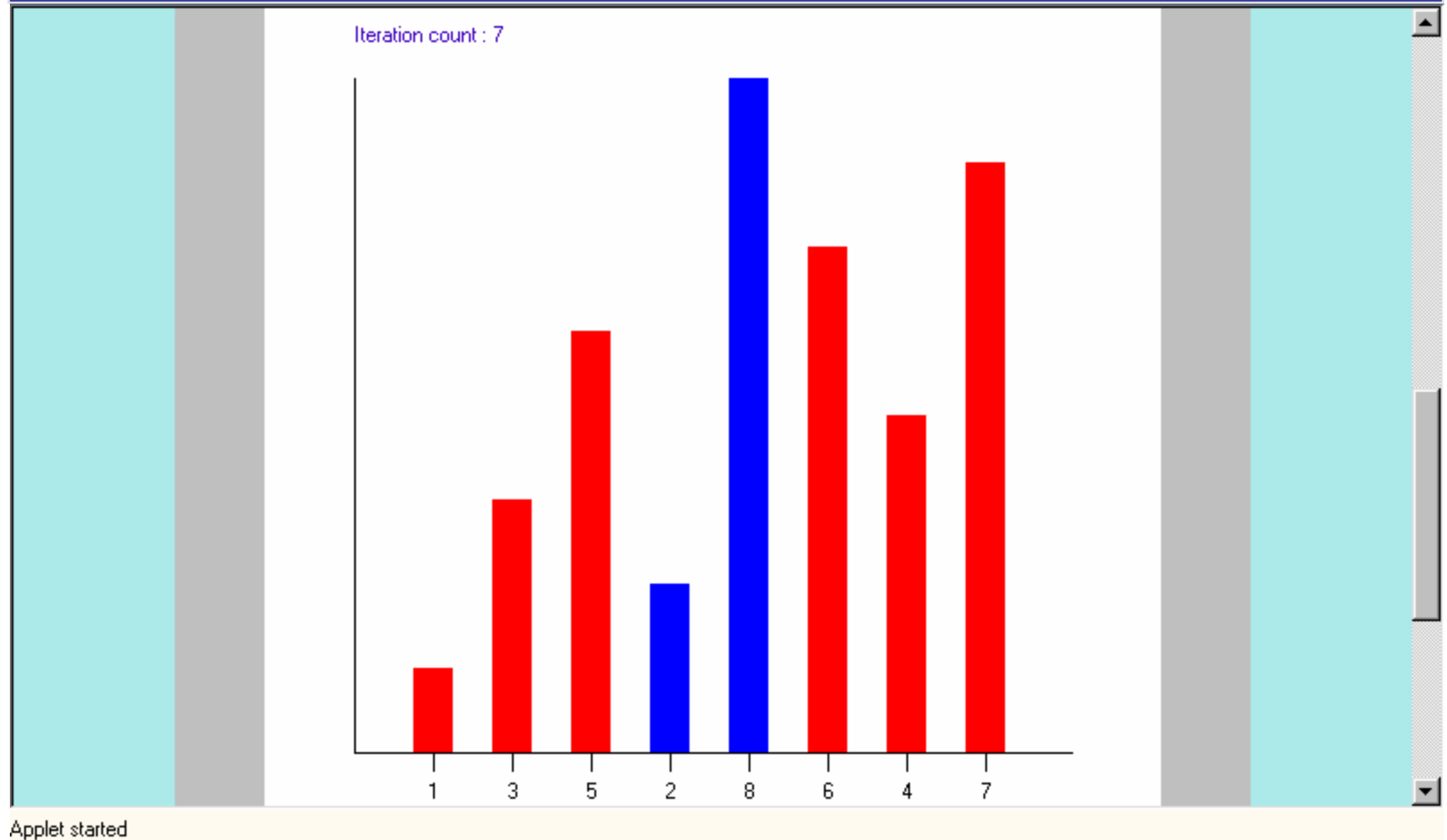


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

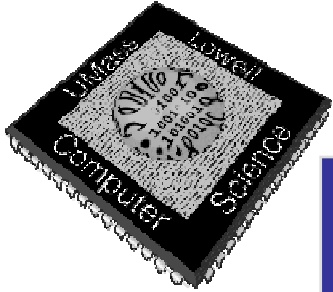


Insertion Sort Animation

Finding a place for item with value 2 in position 4:
Swap item in position 3 with item in position 4.

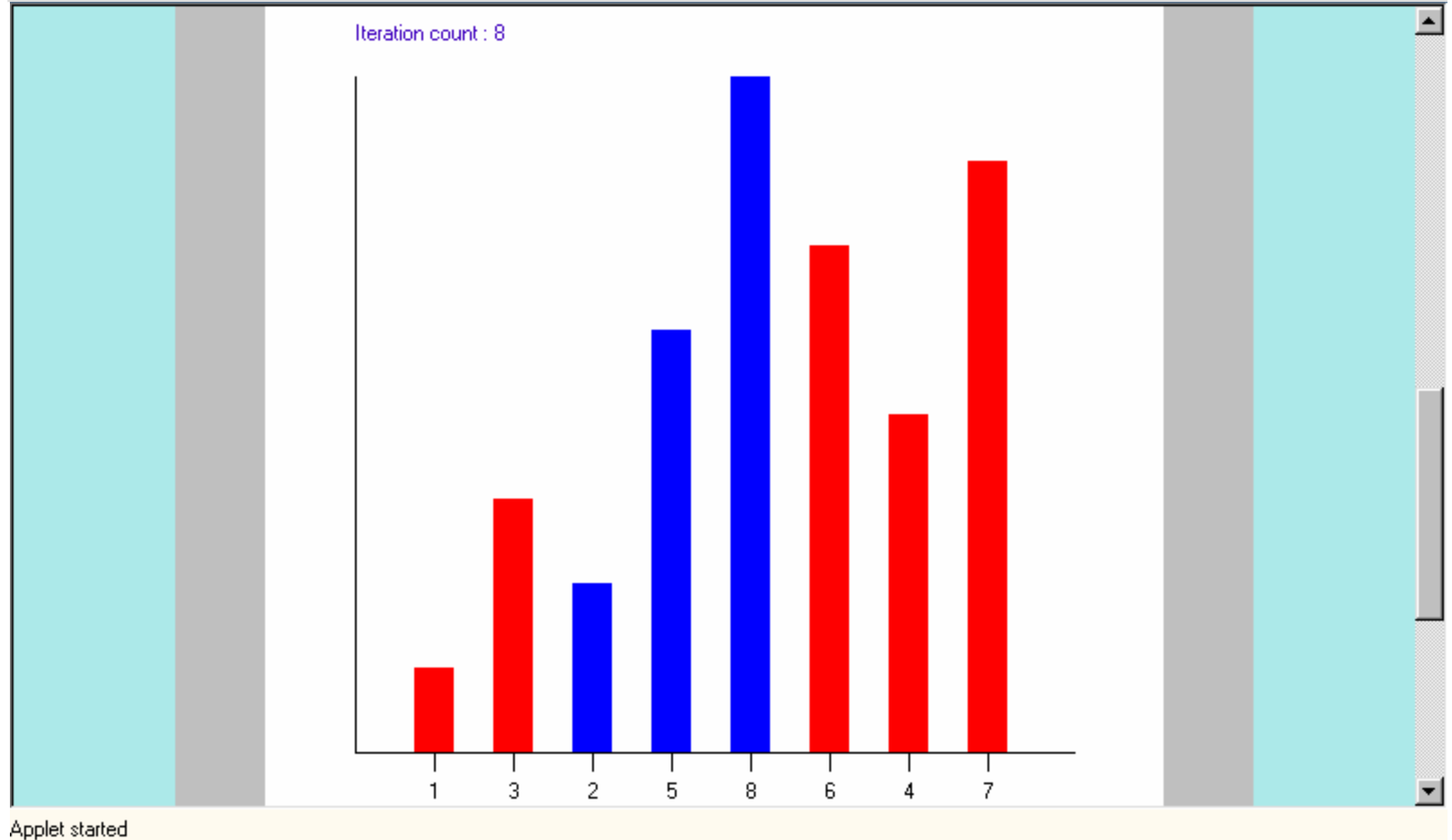


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

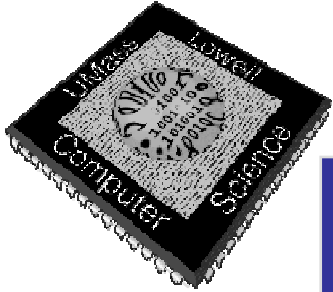


Insertion Sort Animation

Finding a place for item with value 2:
Swap item in position 2 with item in position 3.

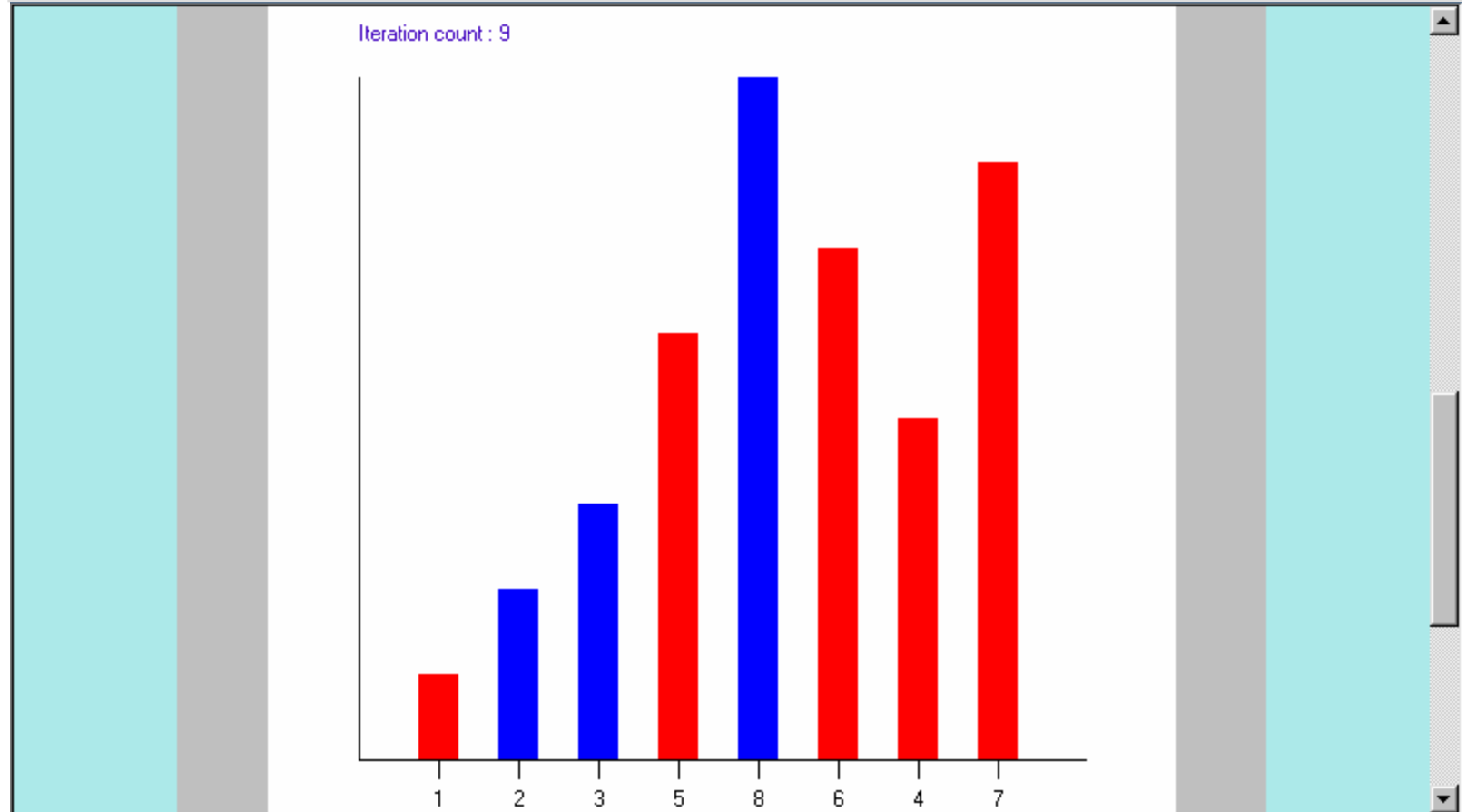


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



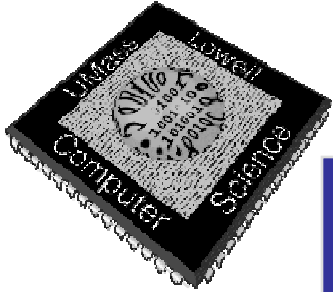
Insertion Sort Animation

Finding a place for item with value 2:
Swap item in position 1 with item in position 2.



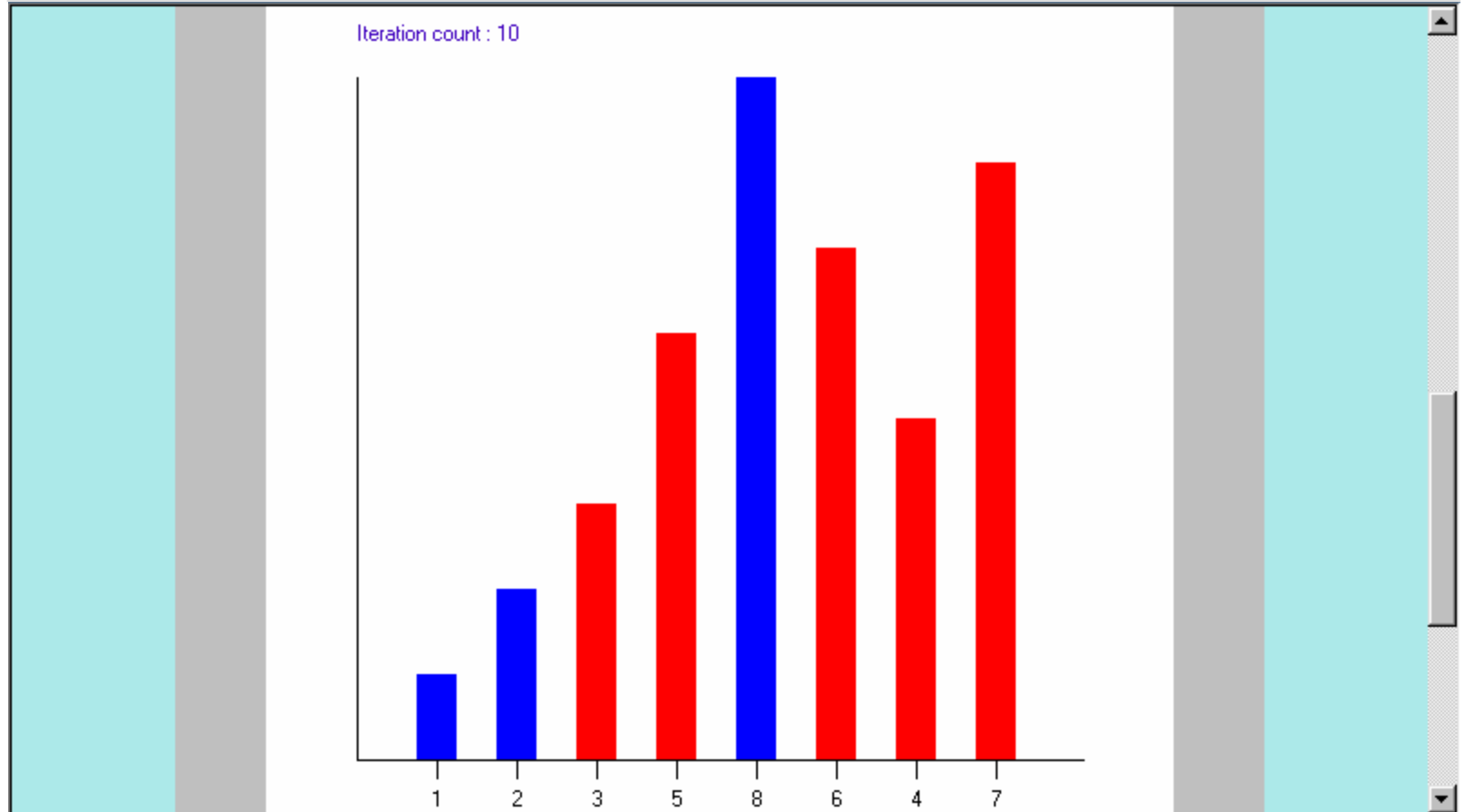
Applet started

<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



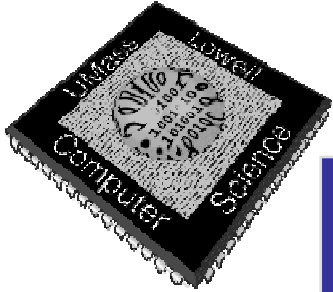
Insertion Sort Animation

Positions 0 through 4 are now in non-decreasing order.



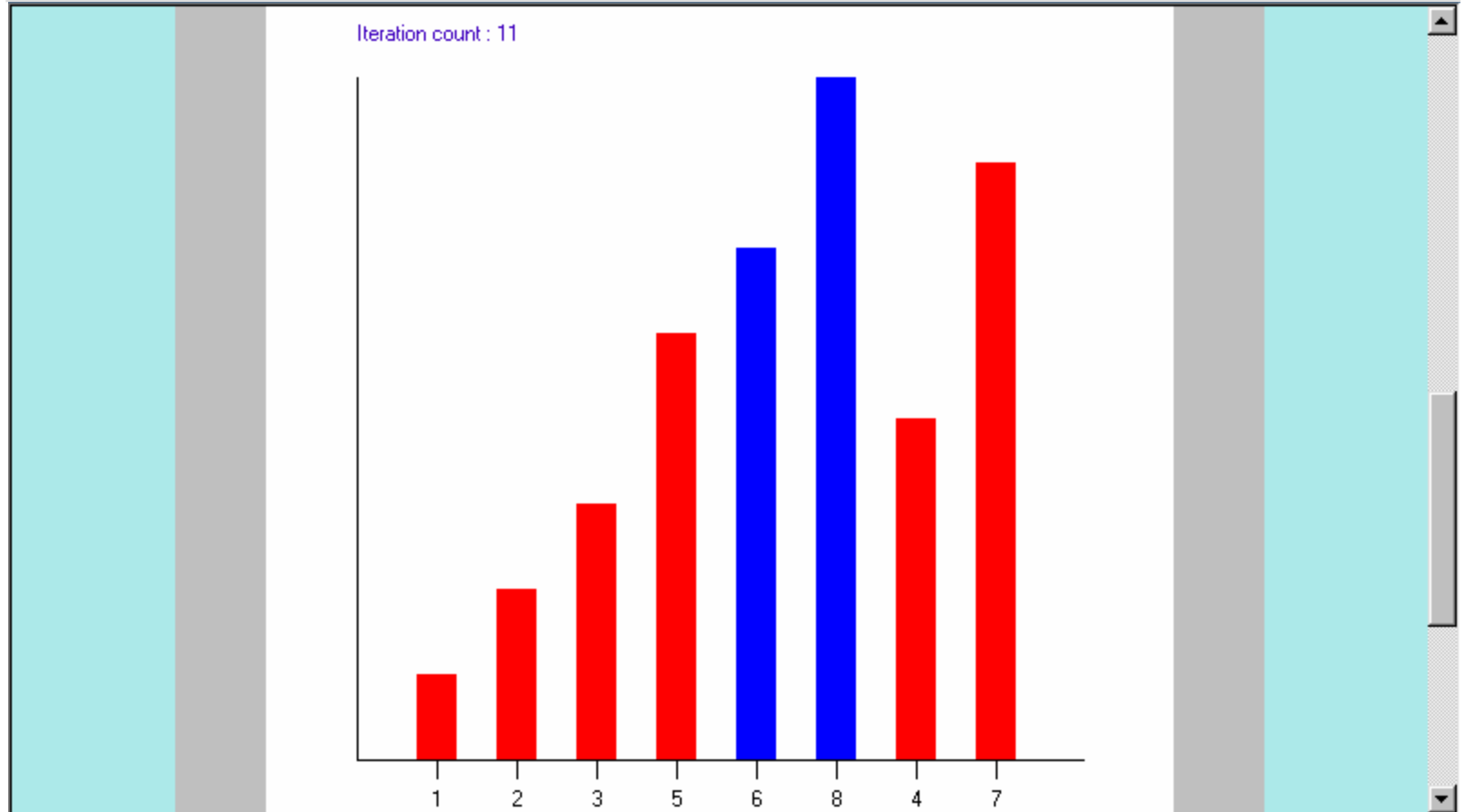
Applet started

<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

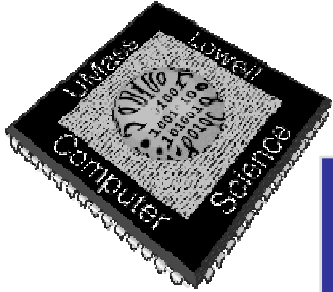


Insertion Sort Animation

Finding a place for item with value 6 in position 5:
Swap item in position 4 with item in position 5.

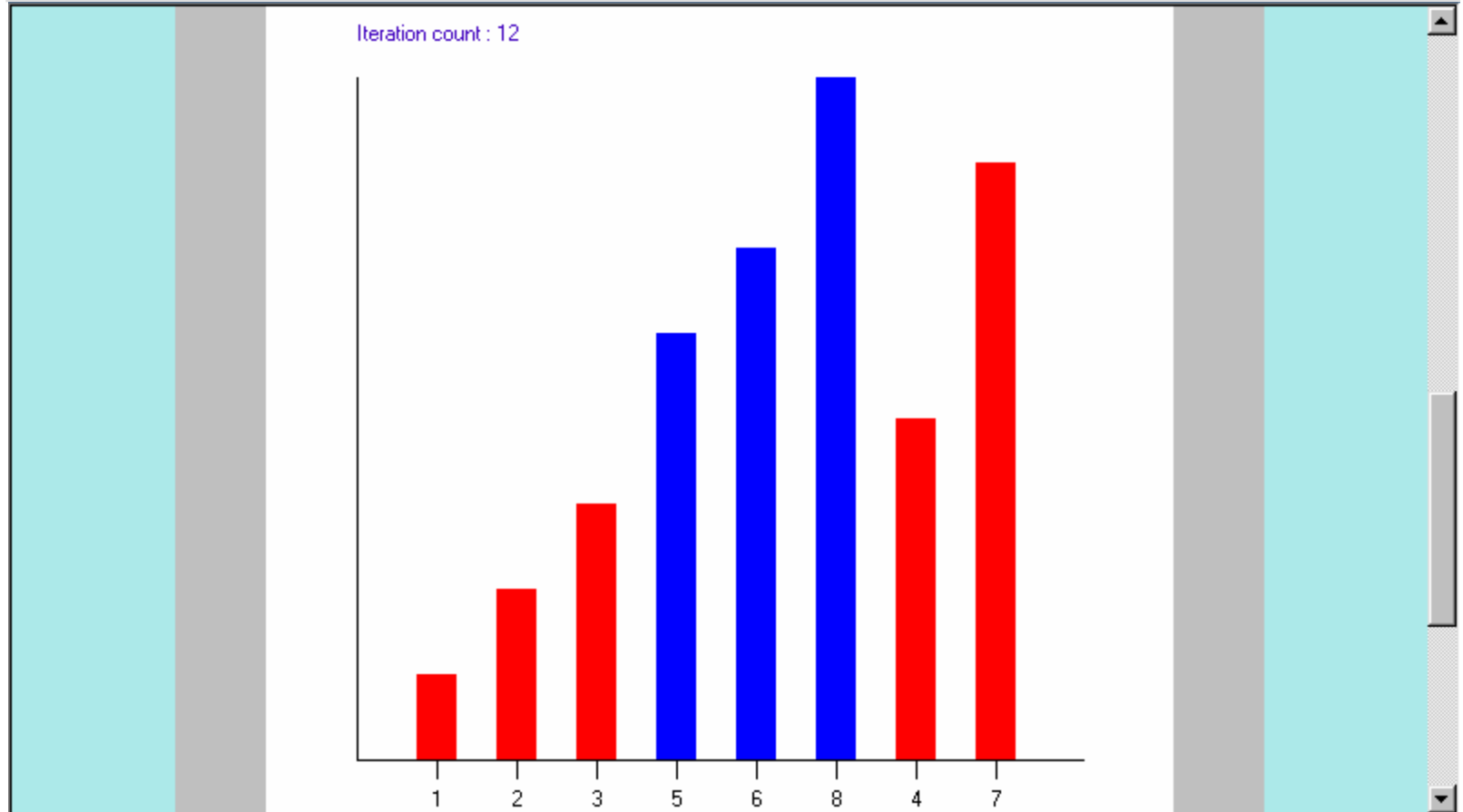


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

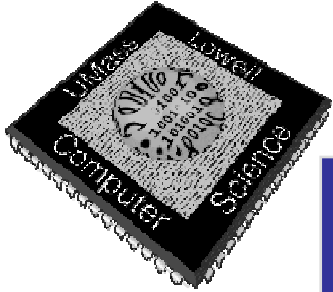


Insertion Sort Animation

Positions 0 through 5 are now in non-decreasing order.

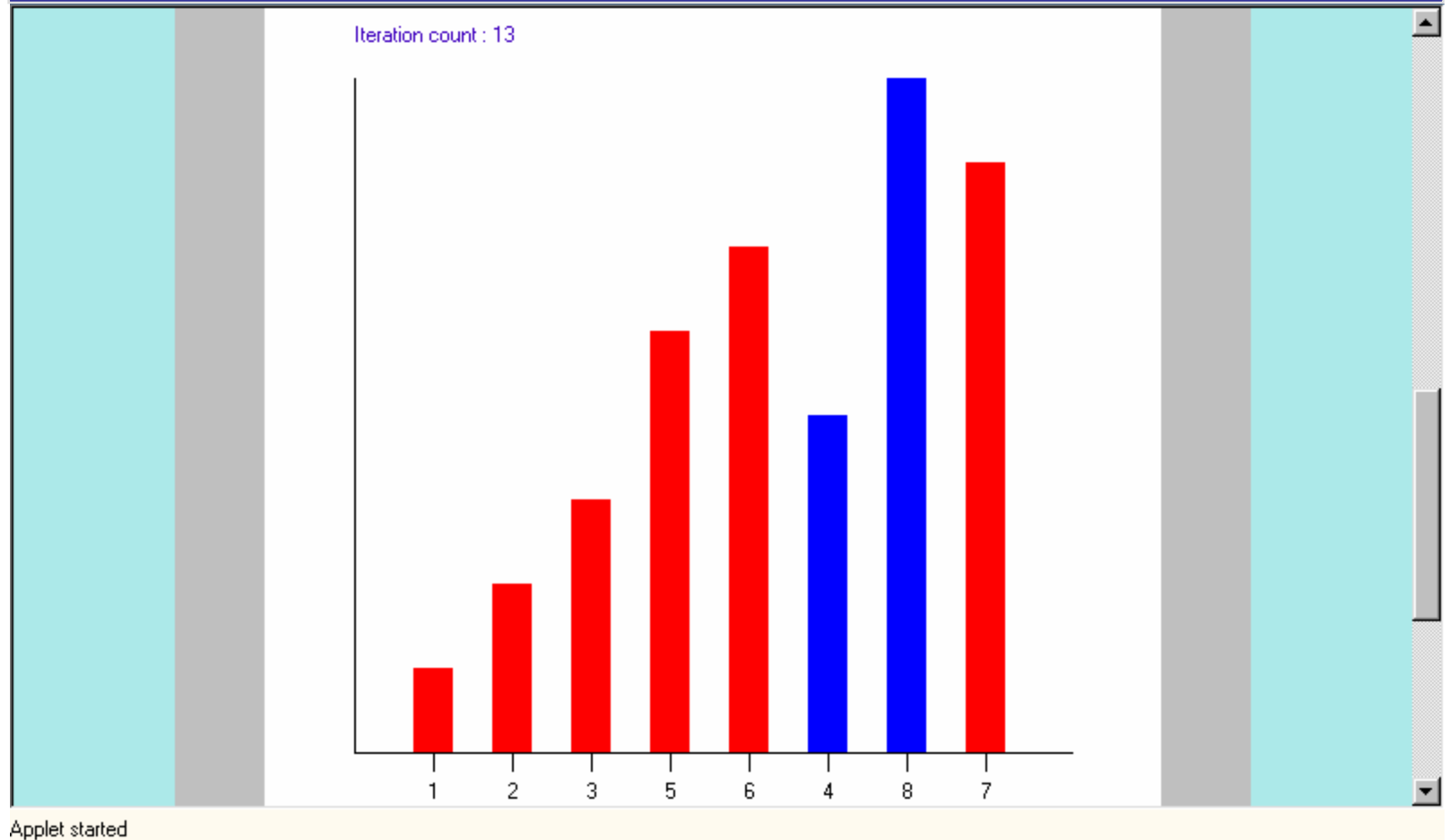


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

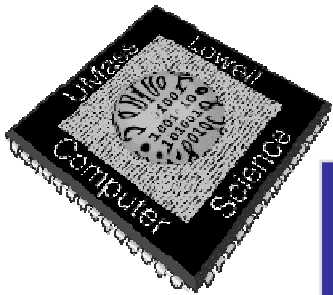


Insertion Sort Animation

Finding a place for item with value 4 in position 6:
Swap item in position 5 with item in position 6.

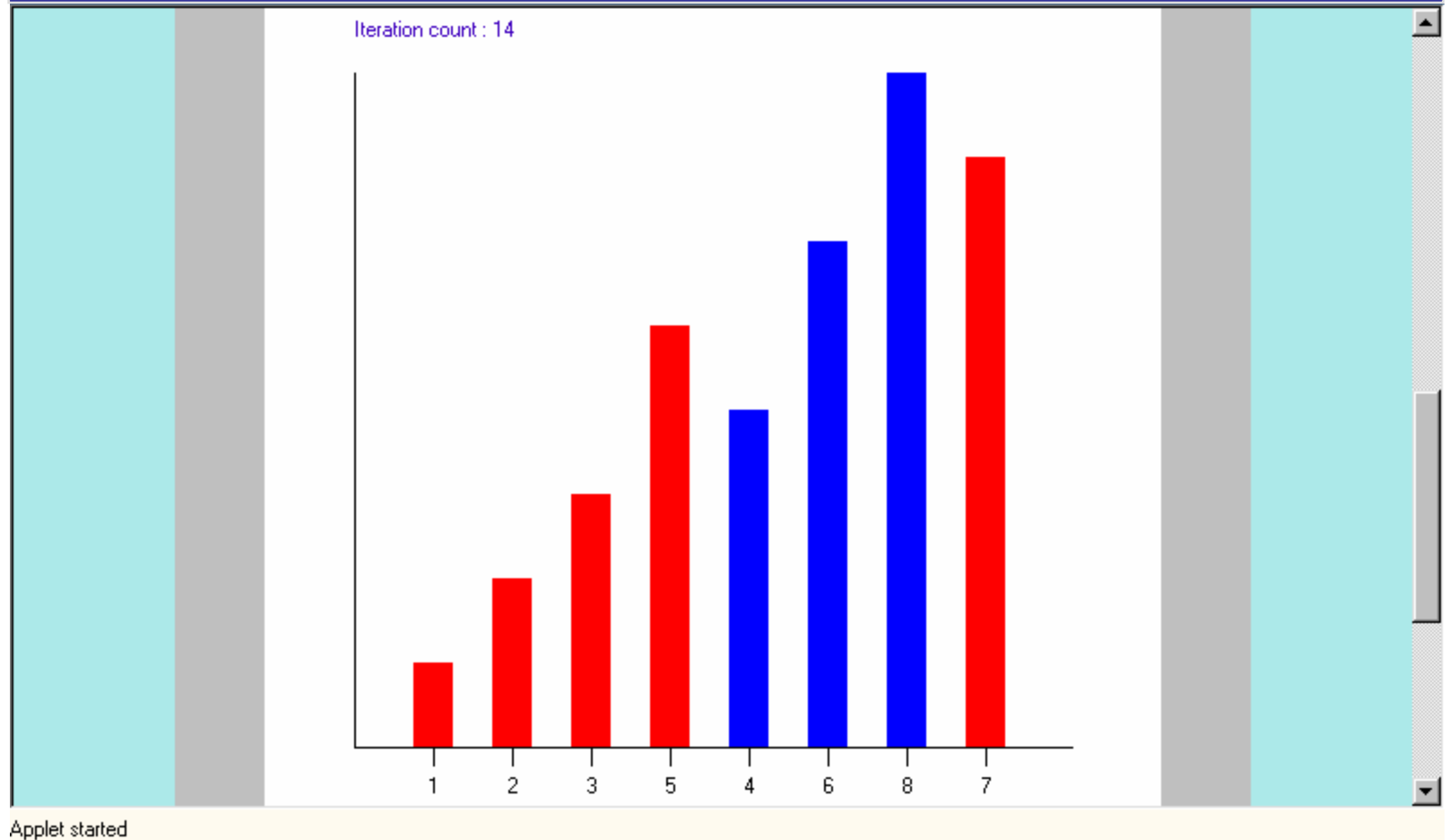


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

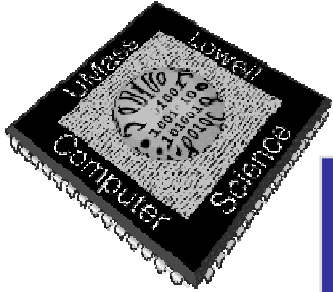


Insertion Sort Animation

Finding a place for item with value 4:
Swap item in position 4 with item in position 5.

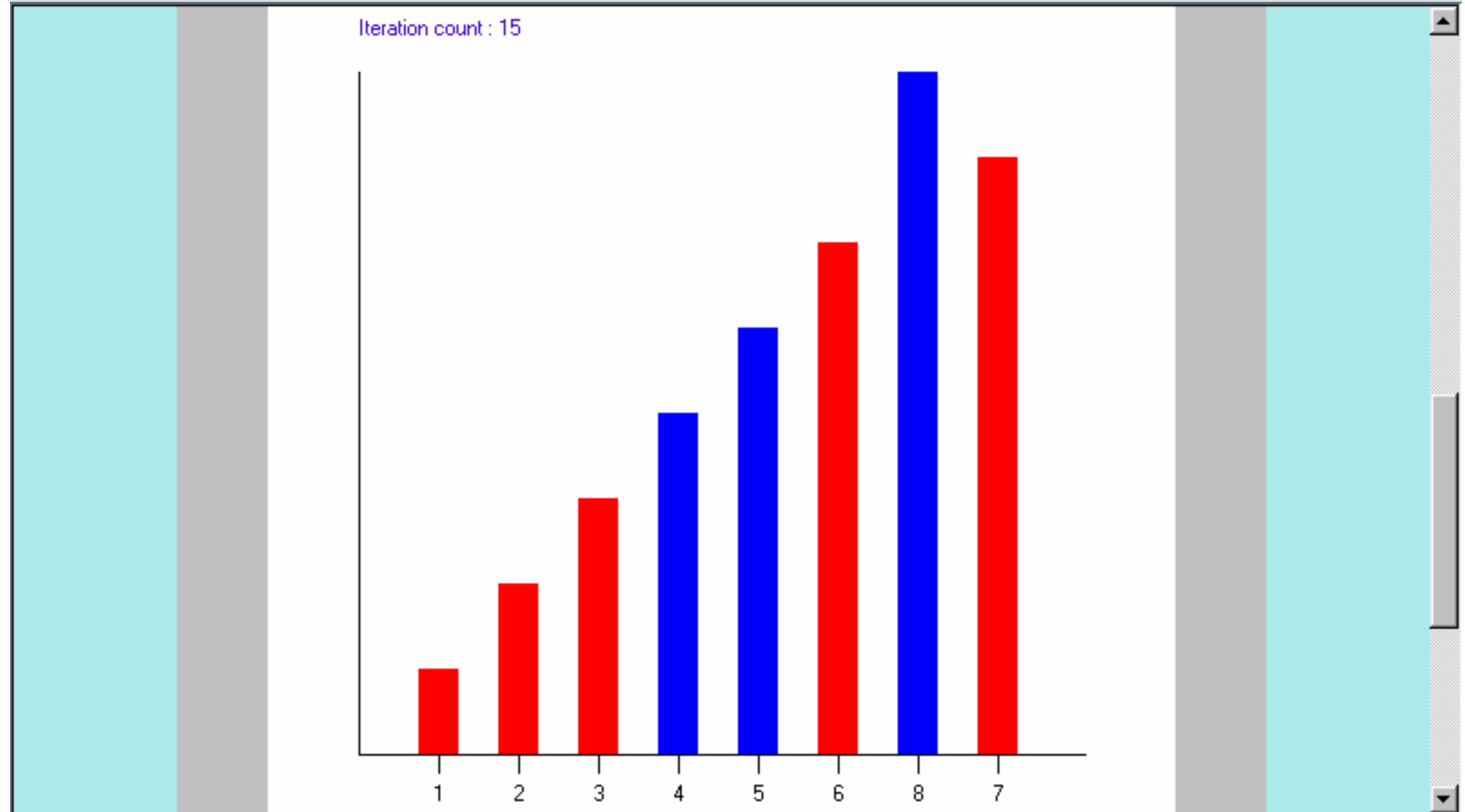


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



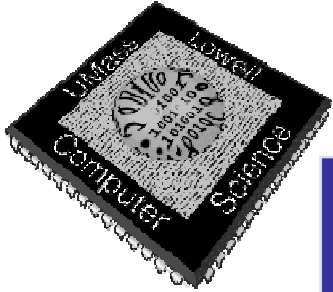
Insertion Sort Animation

Positions 0 through 6 are now in non-decreasing order.



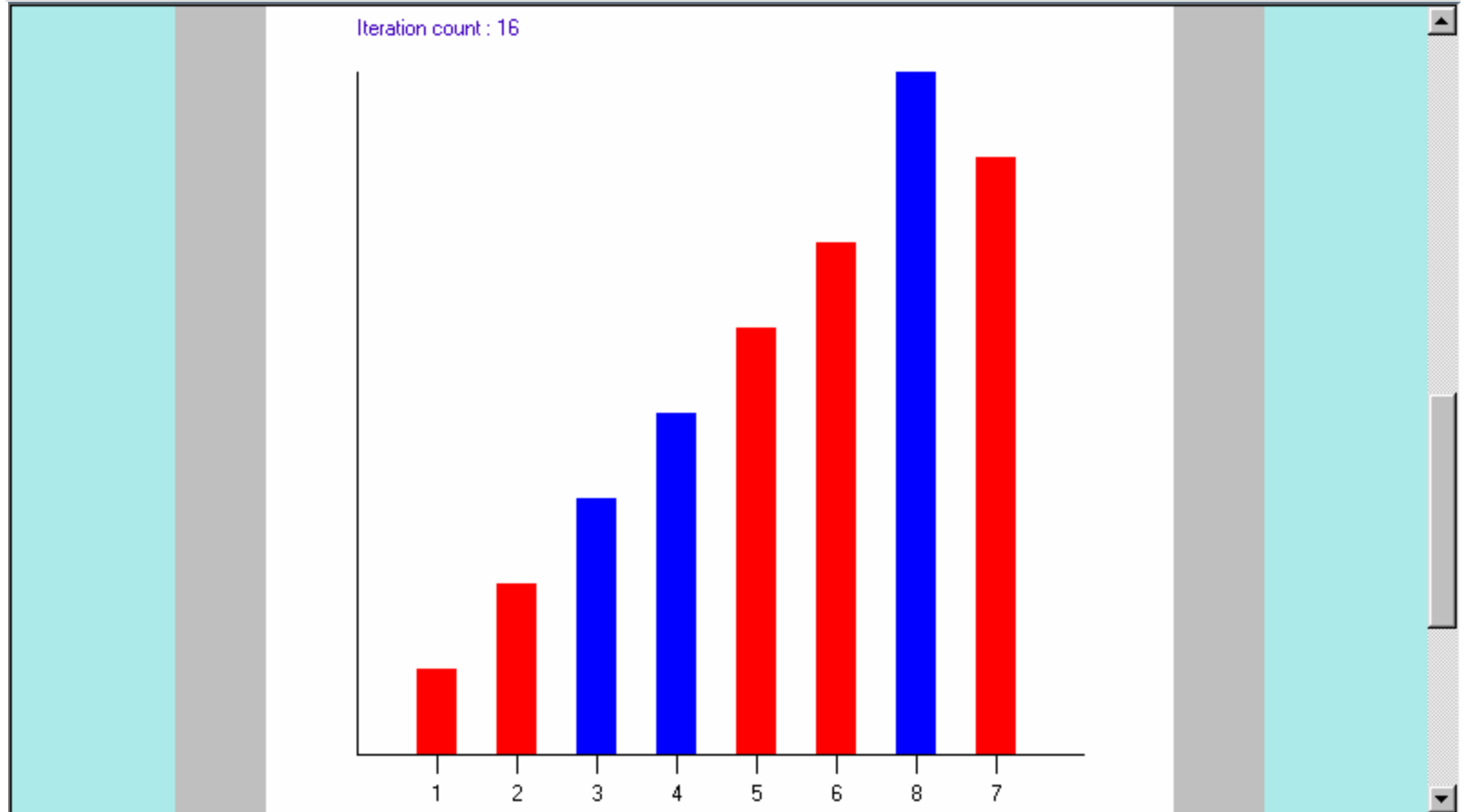
Applet started

<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

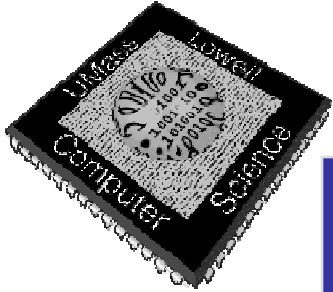


Insertion Sort Animation

Finding a place for item with value 7 in position 7:
Swap item in position 6 with item in position 7.

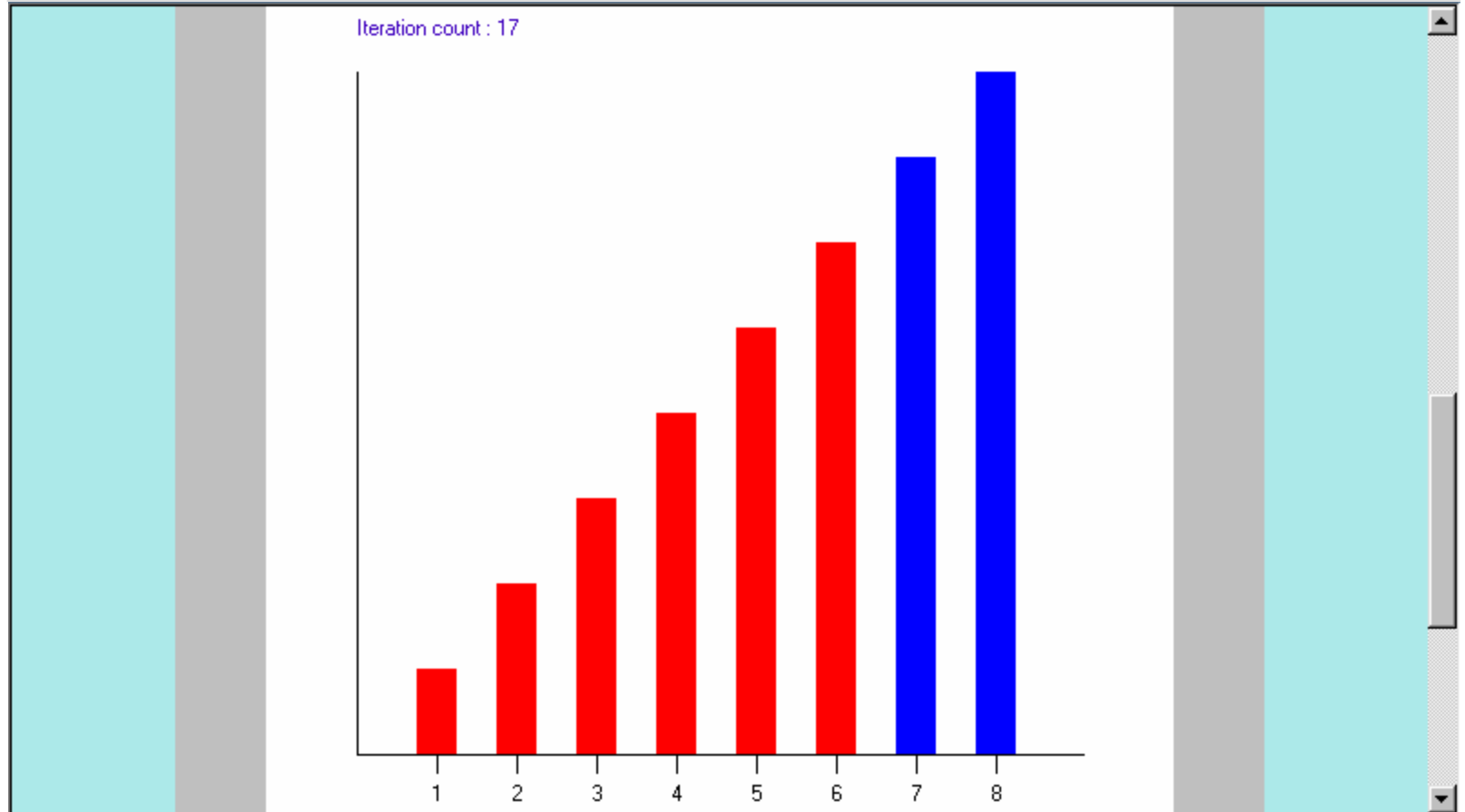


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

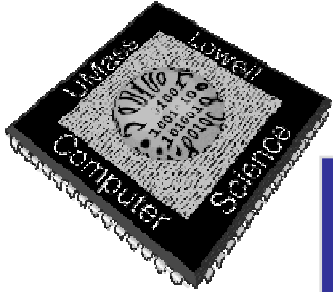


Insertion Sort Animation

Positions 0 through 7 are now in non-decreasing order.

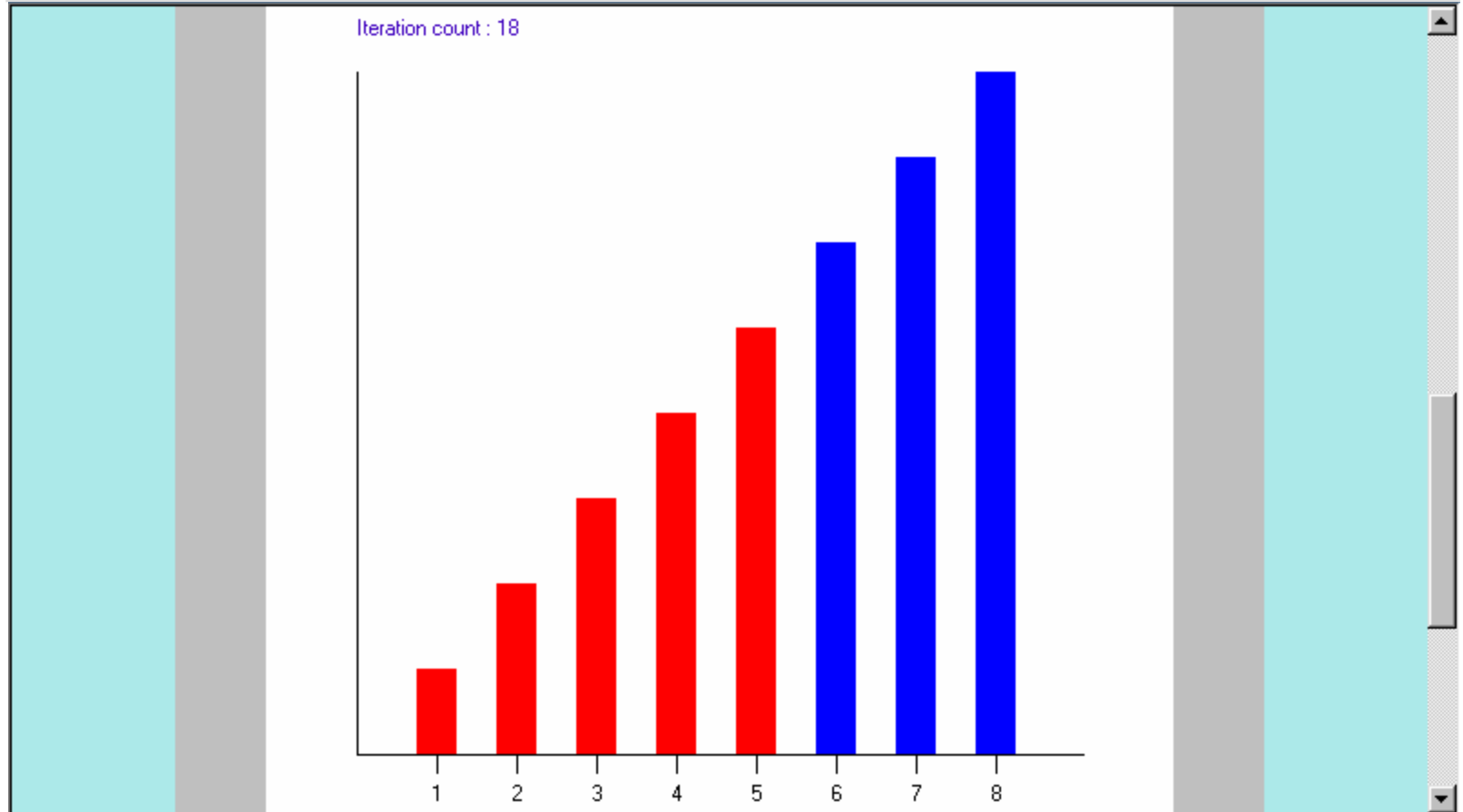


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>

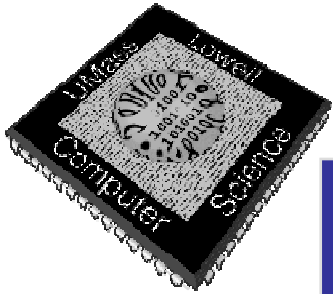


Insertion Sort Animation

Positions 0 through 7 are now in non-decreasing order.

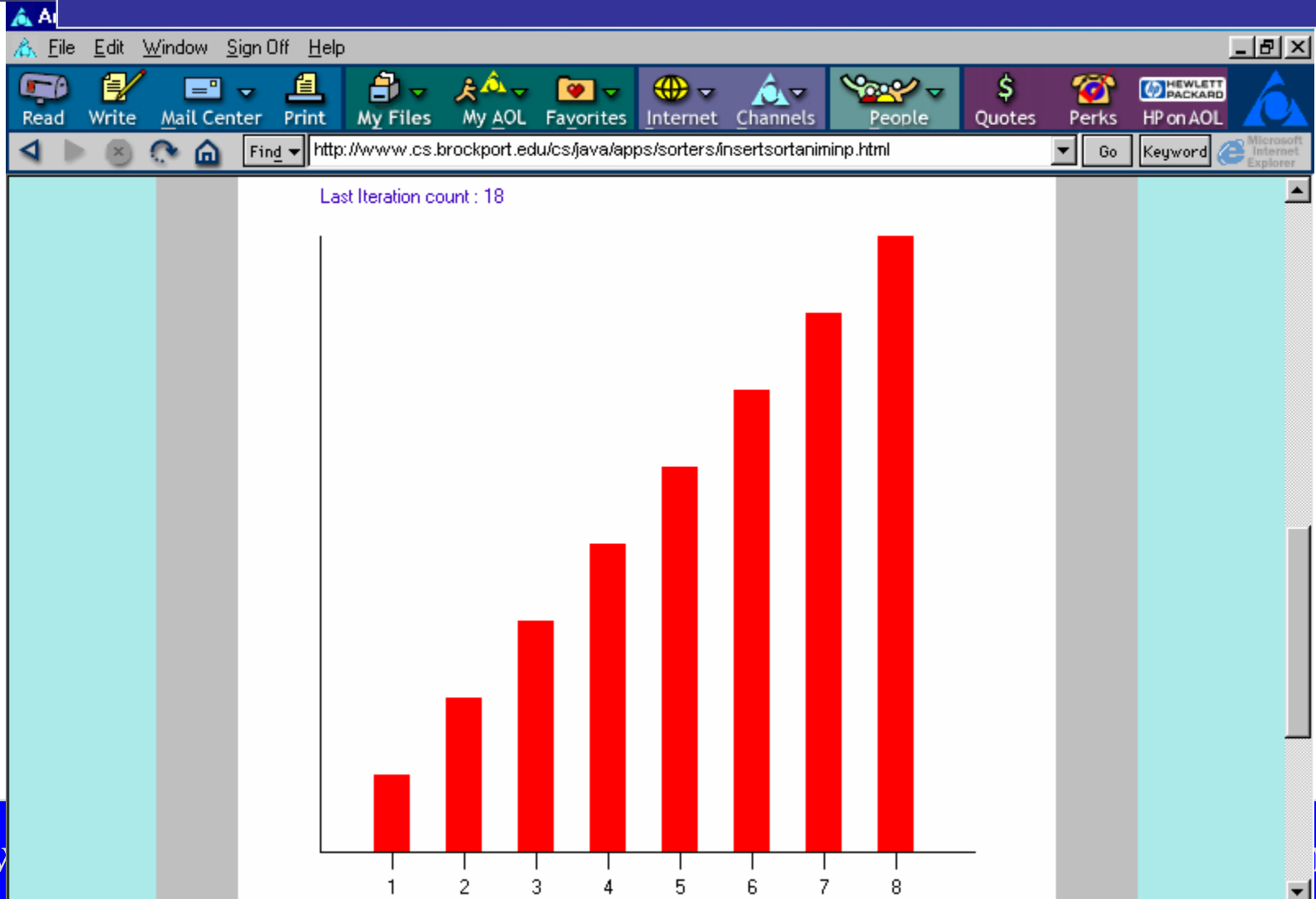


<http://www.cs.brockport.edu/cs/java/apps/sorters/insertsortaniminp.html>



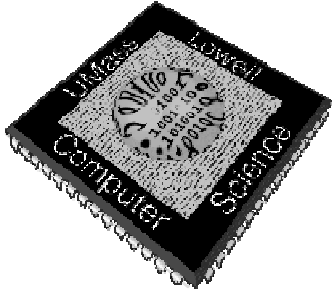
Insertion Sort Animation

Positions 0 through 7 are now in non-decreasing order.



http://ww

l



Insertion sort analysis

```
void insertionSort(int A[], int n)
```

```
{
```

```
    int i, j, tmp;
```

```
    for (i=1; i<n; i++) {
```

```
        tmp=A[i];
```

```
        j = i-1;
```

```
        while (j>=0 && tmp<A[j]) {
```

```
            A[j+1] = A[j];
```

```
            j--;
```

```
        }
```

```
        A[j+1] = tmp;
```

```
    }
```

```
}
```

cost

c_1

c_2

c_4

c_5

c_6

c_7

c_8

times

n

$n-1$

$n-1$

$\sum_{i=1}^{n-1} t_i$

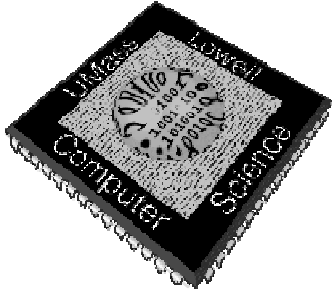
$\sum_{i=1}^{n-1} (t_i - 1)$

$\sum_{i=1}^{n-1} (t_i - 1)$

$n-1$

best case $t_i = 1$

worst case $t_i = i + 1$



Insert sort cost

$$T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=1}^{n-1} t_i + c_6 \sum_{i=1}^{n-1} (t_i - 1) + c_7 \sum_{i=1}^{n-1} (t_i - 1) + c_8(n-1)$$

best case $t_i = 1$

$$\begin{aligned} T(n) &= c_1n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=1}^{n-1} t_i + c_6 \sum_{i=1}^{n-1} (t_i - 1) + c_7 \sum_{i=1}^{n-1} (t_i - 1) + c_8(n-1) \\ &= c_1n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

worst case $t_i = i + 1$

$$\begin{aligned} T(n) &= c_1n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=1}^{n-1} t_i + c_6 \sum_{i=1}^{n-1} (t_i - 1) + c_7 \sum_{i=1}^{n-1} (t_i - 1) + c_8(n-1) \\ &= c_1n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=1}^{n-1} (i+1) + c_6 \sum_{i=1}^{n-1} i + c_7 \sum_{i=1}^{n-1} i + c_8(n-1) \\ &= \frac{c_5 + c_6 + c_7}{2} n^2 + (c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6 + c_7}{2} + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$