CSC 225

Algorithms and Data Structures I
Fall 2014
Rich Little

Sorting

Reading Assignment: Chapter 2

- **Sorting definition.** The process of ordering a sequence of objects according to some linear order.
- **Total versus partial order.** If any two elements in a set are comparable, then the set can be totally ordered otherwise partially ordered.
- Total order
  - 9 9 14 17 86
  - Coady Müller Stege Storey Thomo
- Partial order (Topological sort)
  - CSC 110<CSC 115<CSC 225 | SENG 321<SENG 371<SENG 426

Sorting

Reading Assignment: Chapter 2

- The basic unit for analyzing the time complexity of sorting algorithms is a comparison
  - $< \leq = \geq >$ int, char, string
  - $\leq = \geq$ should not be used for real or double
- Standard character sets
  - ASCII (256 characters)
  - Unicode (95,221 characters) http://www.i18nguy.com/unicode/char-count.html
  - ISO
  - EBCDIC

Sorting Reference Lists

Unsorted

Thagard, P., & Verbeurgt, K. (1998). *Coherence as constraint satisfaction.*

⋮

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability.*

⋮

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research.*

⋮

Gross, J. & Yellen, J. (1999). *Graph theory and its applications*.

Sorted

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research.*

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability.*

Gross, J. & Yellen, J. (1999). *Graph theory and its applications*.

⋮

Thagard, P., & Verbeurgt, K. (1998). *Coherence as constraint satisfaction.*

Sorting Address/Phone Books

Unsorted

J. Smith, 1420 Cook street, (250) 234 5627

⋮

D. S. Johnson, 130 Fort street, (250) 380 5627

⋮

M. Brent, 1110 Quadra street, (250) 380 9876

⋮

J. T. Vui, 230 Linden Ave., (250) 240 6517

⋮

R. Tristan, 1200 Dallas Road, (250) 340 3425

Sorted

M. Brent, 1110 Quadra street, (250) 380 9876

⋮

D. S. Johnson, 130 Fort street, (250) 380 5627

⋮

J. Smith, 1420 Cook street, (250) 234 5627

⋮

R. Tristan, 1200 Dallas Road, (250) 340 3425

⋮

J. T. Vui, 230 Linden Ave., (250) 240 6517

Index-Entries are Sorted

(after **Dictionary of Algorithms and Data Structures**
At http://www.nist.gov/dads/)

A

abstract data type

accepting state

Ackermann's function

acyclic graph

algorithm

amortized cost

approximation algorithm

array

B

backtracking

big-O notation

binary tree

M

matching

matrix

merge sort

Z

Zeller's congruence

Zipfian distribution

Zipf's law

Internal versus External Sorting

- Internal sorting takes place in main memory
  - Quicksort
  - Heapsort
  - Bubblesort
  - Mergesort (as long as all of the data fits into memory)
- External sorting is necessary if the number of objects is too large to fit into main memory
  - Mergesort
  - Merge streams of internally sorted data

Classes of Sorting Algorithms

Algorithm	**Complexity**	**Remarks**
Insertionsort	$O(n^2)$	Simple
Bubblesort	$O(n^2)$	
Selectionsort	$O(n^2)$	
Shellsort	$O(n^2)$	
Mergesort	O(n log n)	External sorting
Quicksort	$O(n^2)$ worst O(n log n) expected	Hoare 1962 Most practical
Heapsort	O(n log n)	
Smoothsort	O(n log n) O(n) best case	Dijkstra 1979
Lexicographic	O(n)	Restricted input
Bin or radixsort	O(n)	Restricted input

Computational Problem: **Sorting**

Input: A collection of *n* objects (stored in a list, vector, or sequence) and a comparator defining a total order on these objects

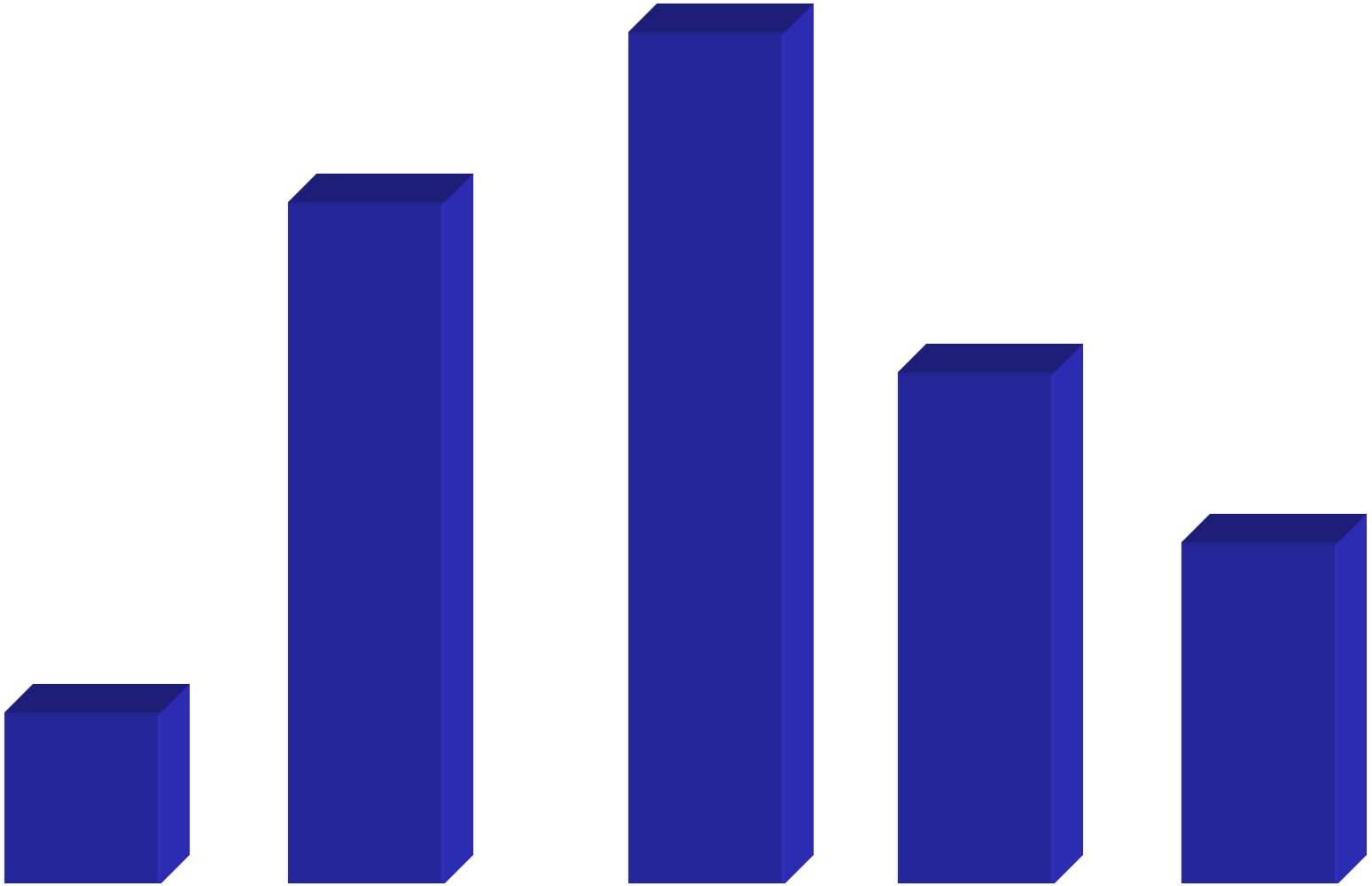
Output: Produce an ordered representation of these objects

Algorithm Design Technique
Brute Force

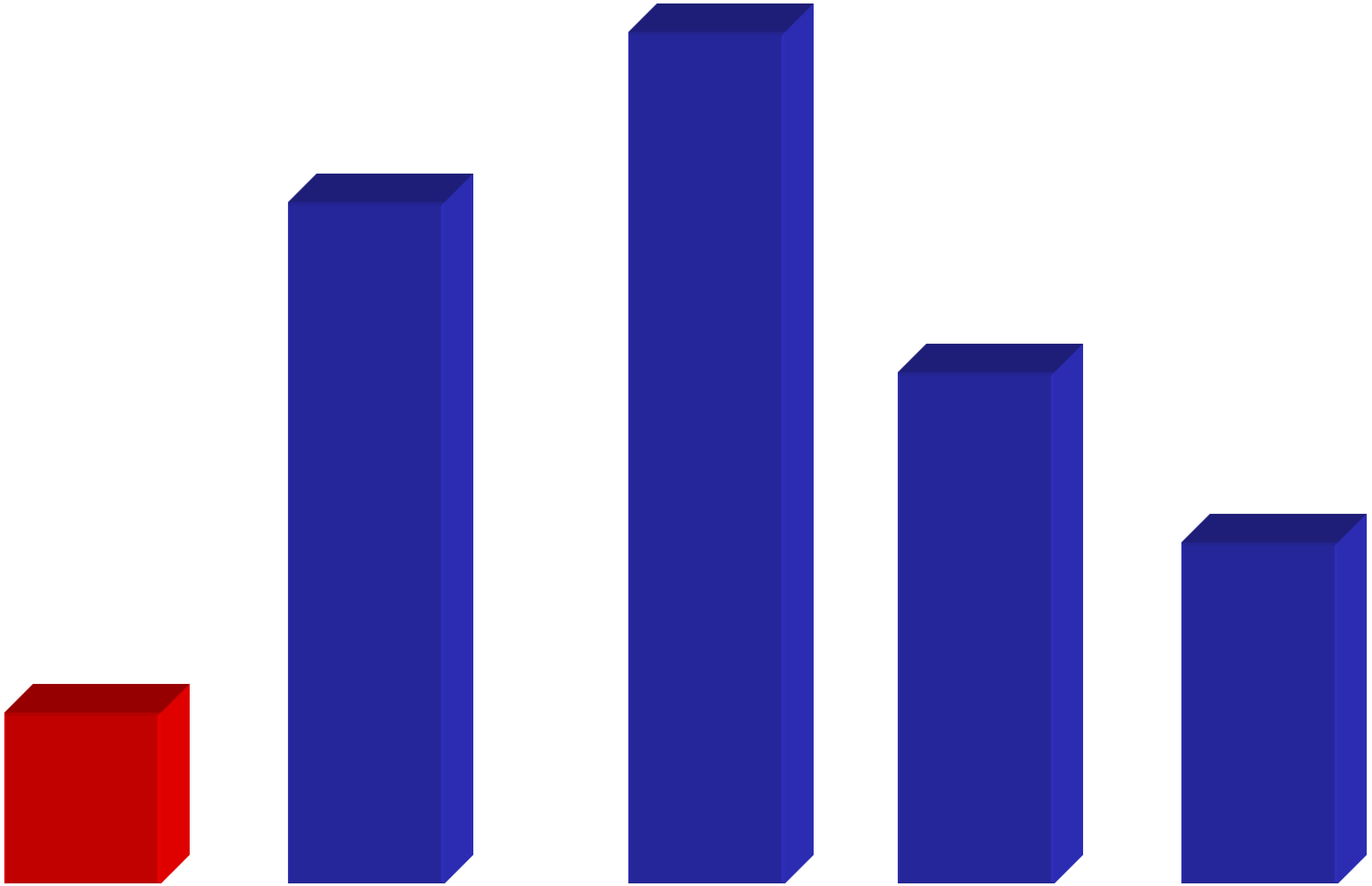
- **Brute force** is a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved.

- Simplest algorithm design technique
- Does not usually produce the most elegant or most efficient algorithm
- Examples of the Brute Force technique
  - Selection sort
  - Bubble sort
  - Sequential search
  - Exhaustive search

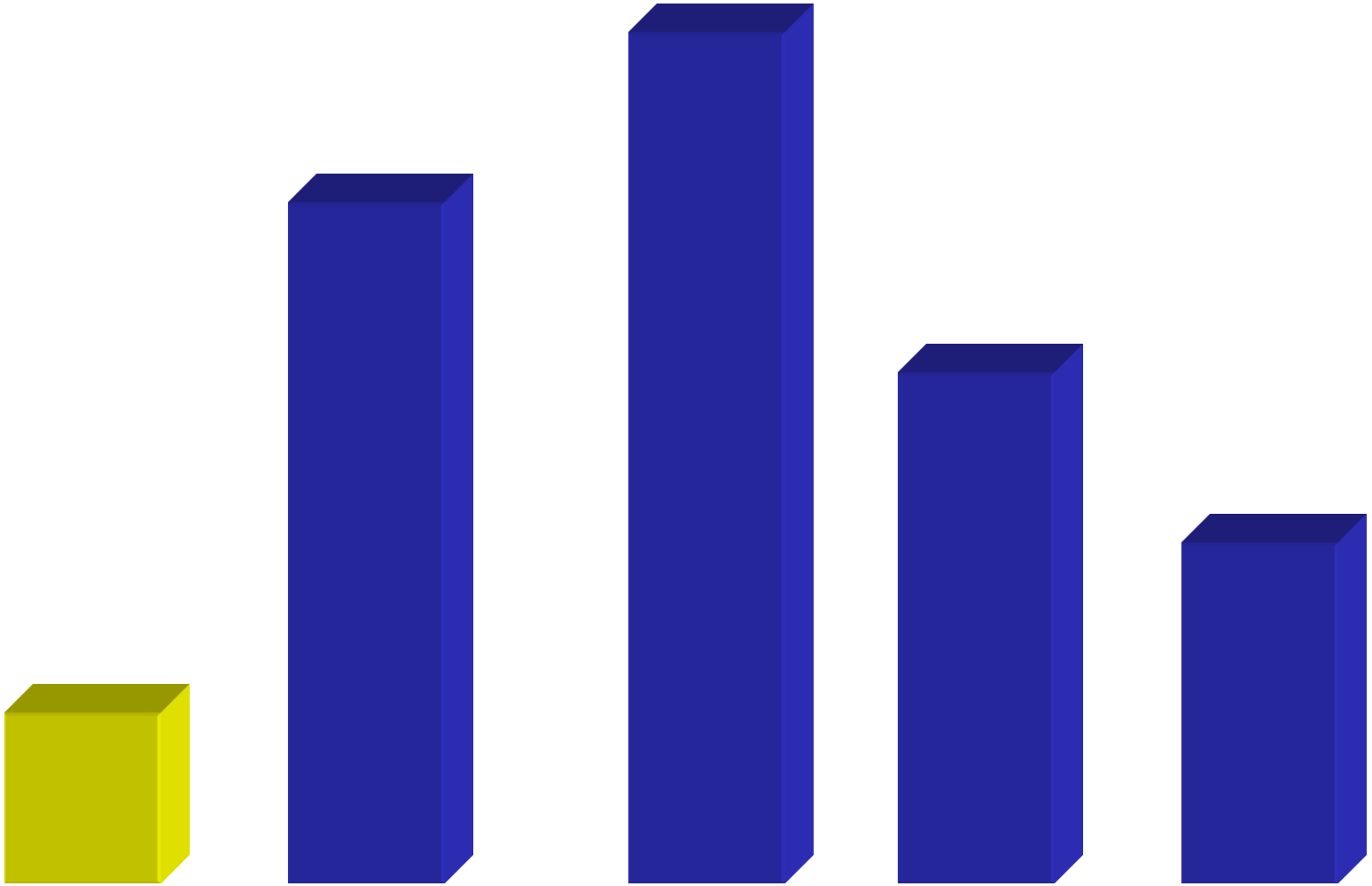
Selection Sort Animation



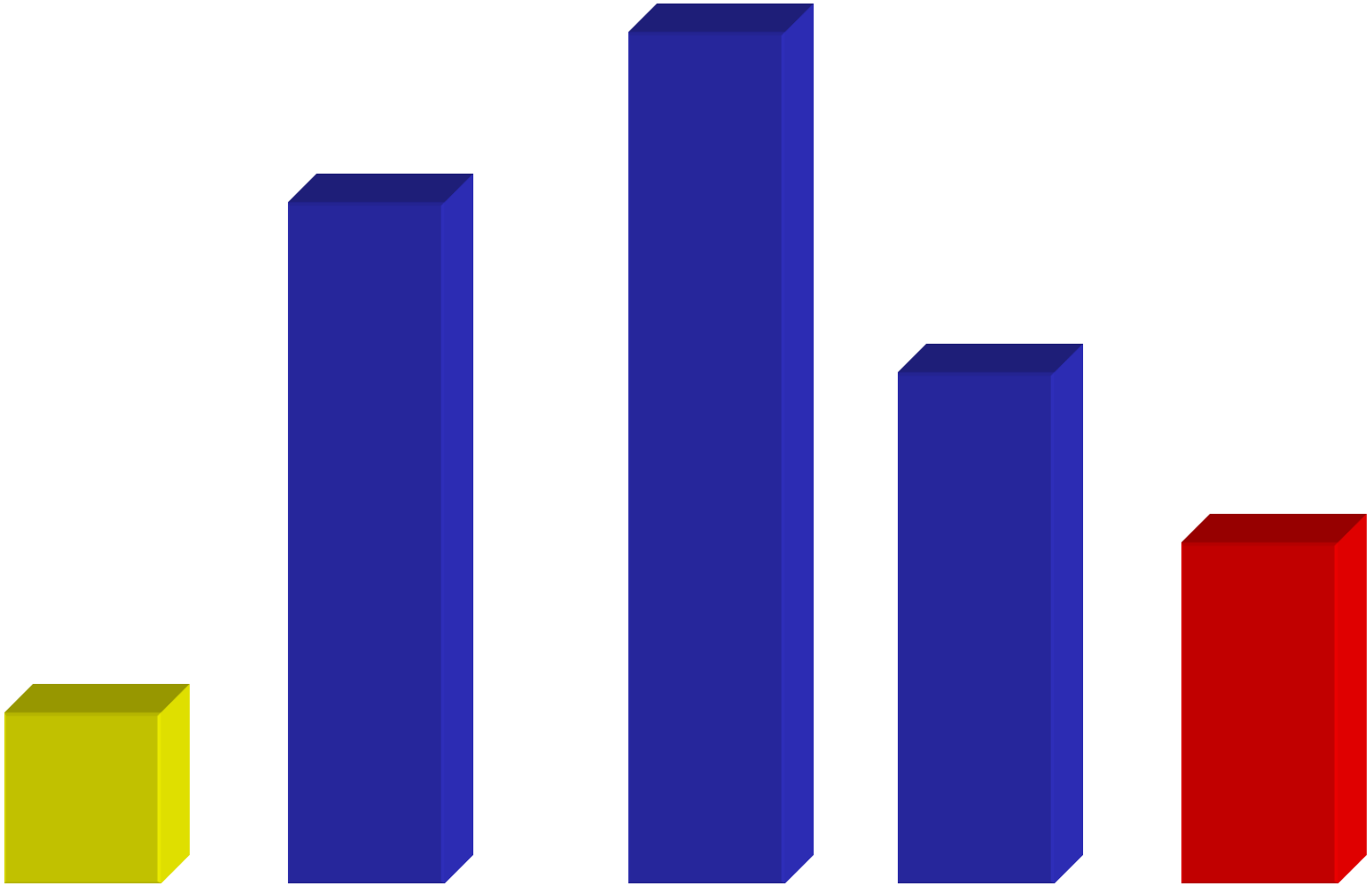
Selection Sort Animation



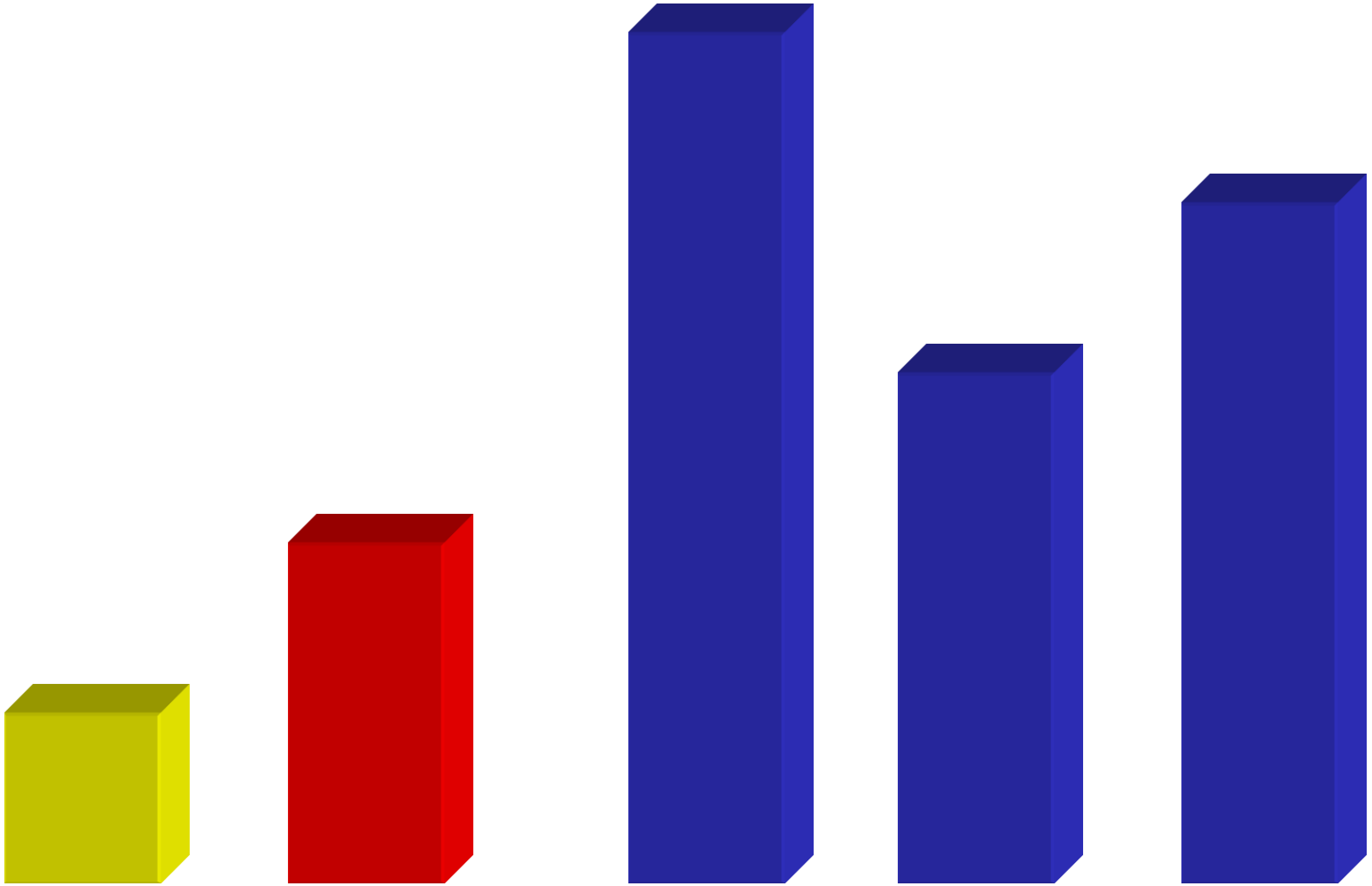
Selection Sort Animation



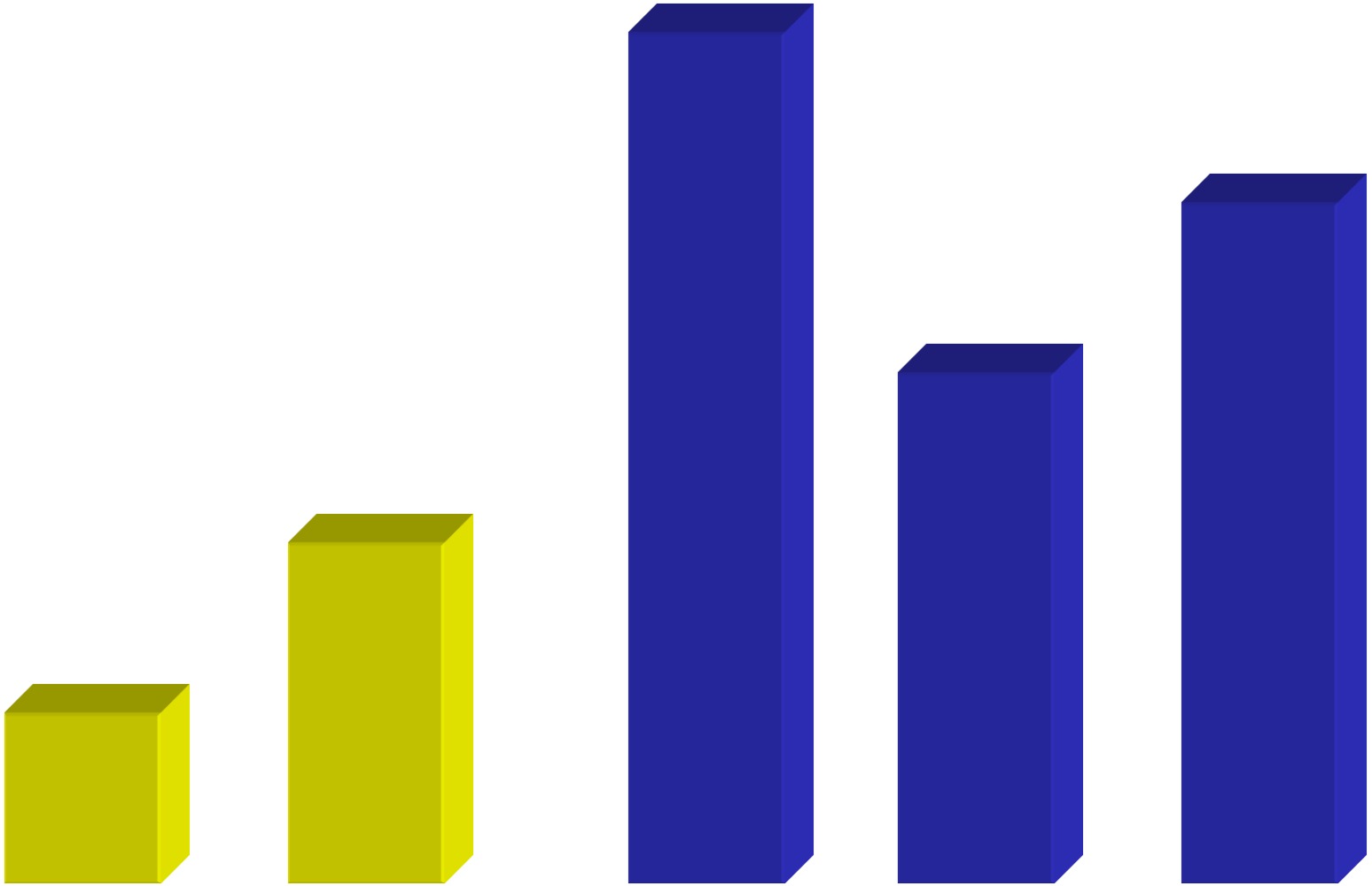
Selection Sort Animation



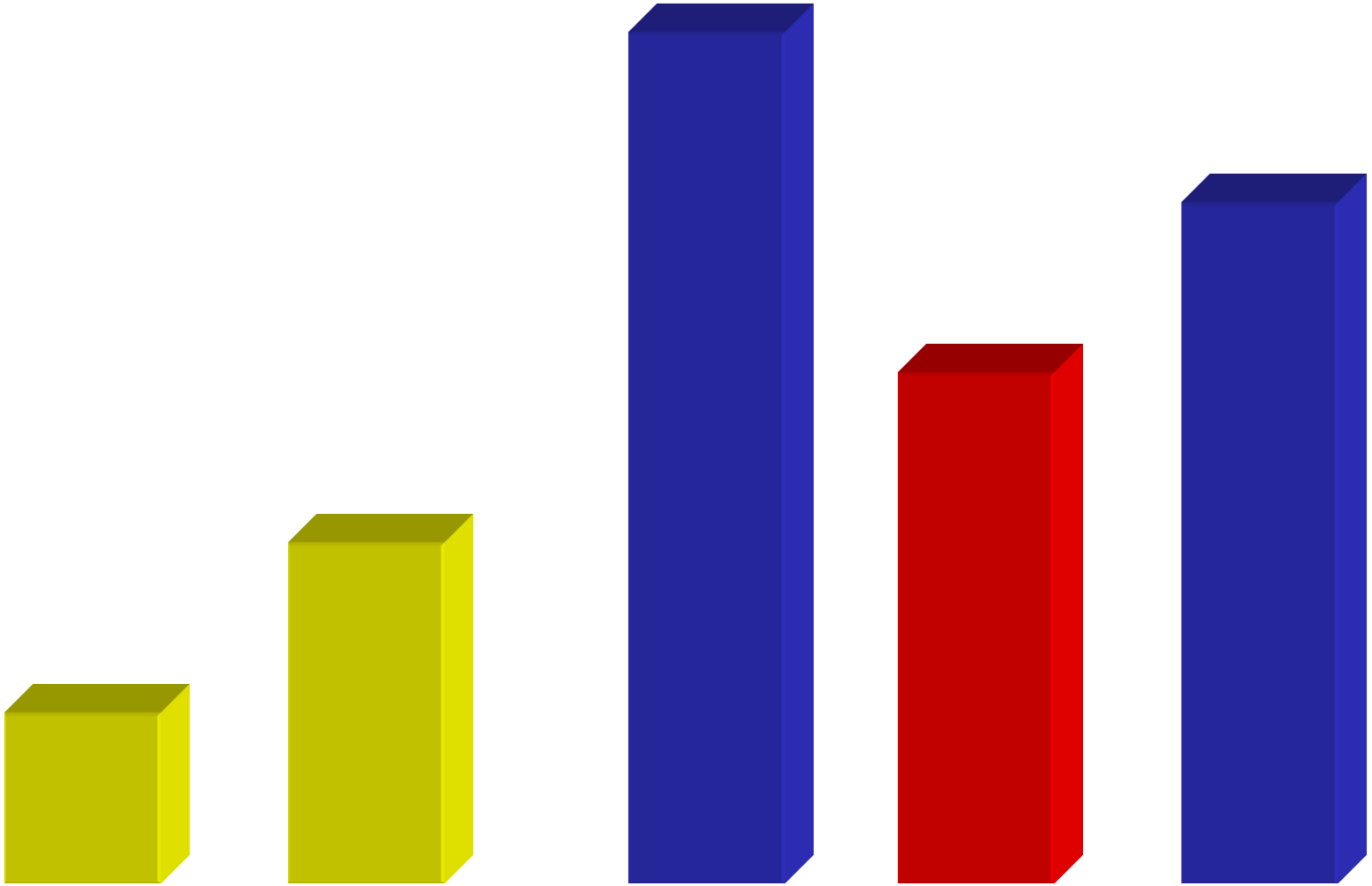
Selection Sort Animation



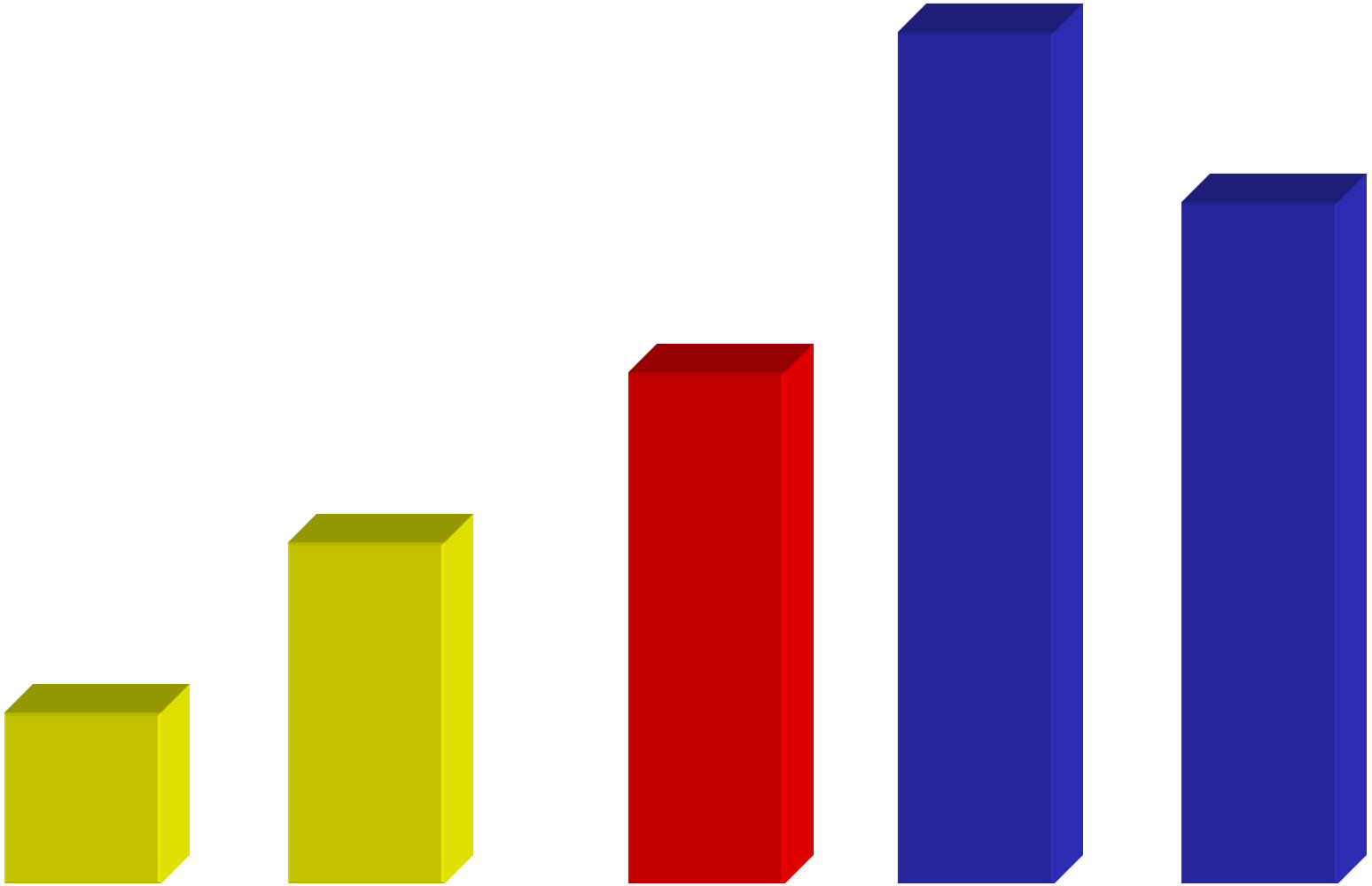
Selection Sort Animation



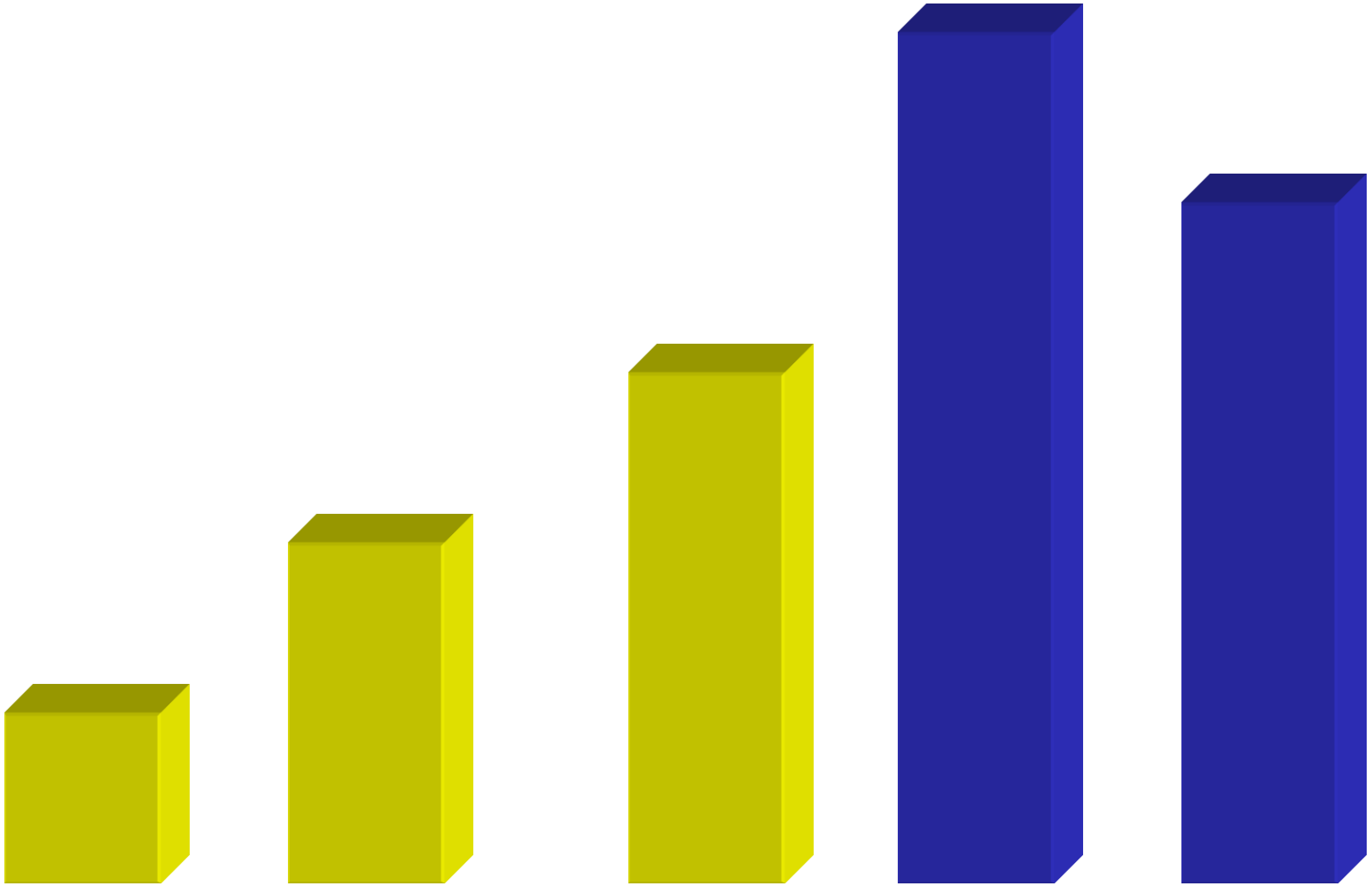
Selection Sort Animation



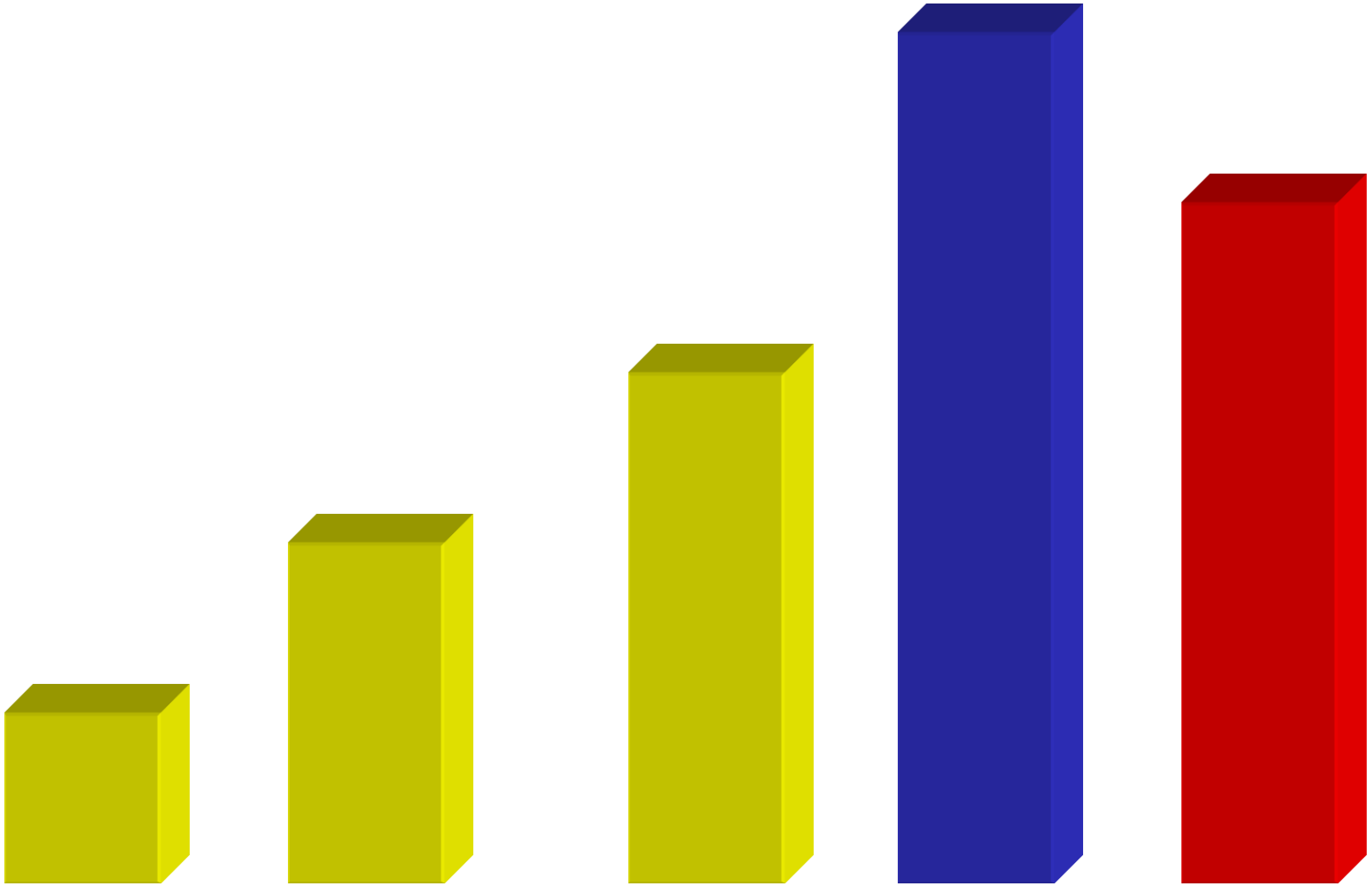
Selection Sort Animation



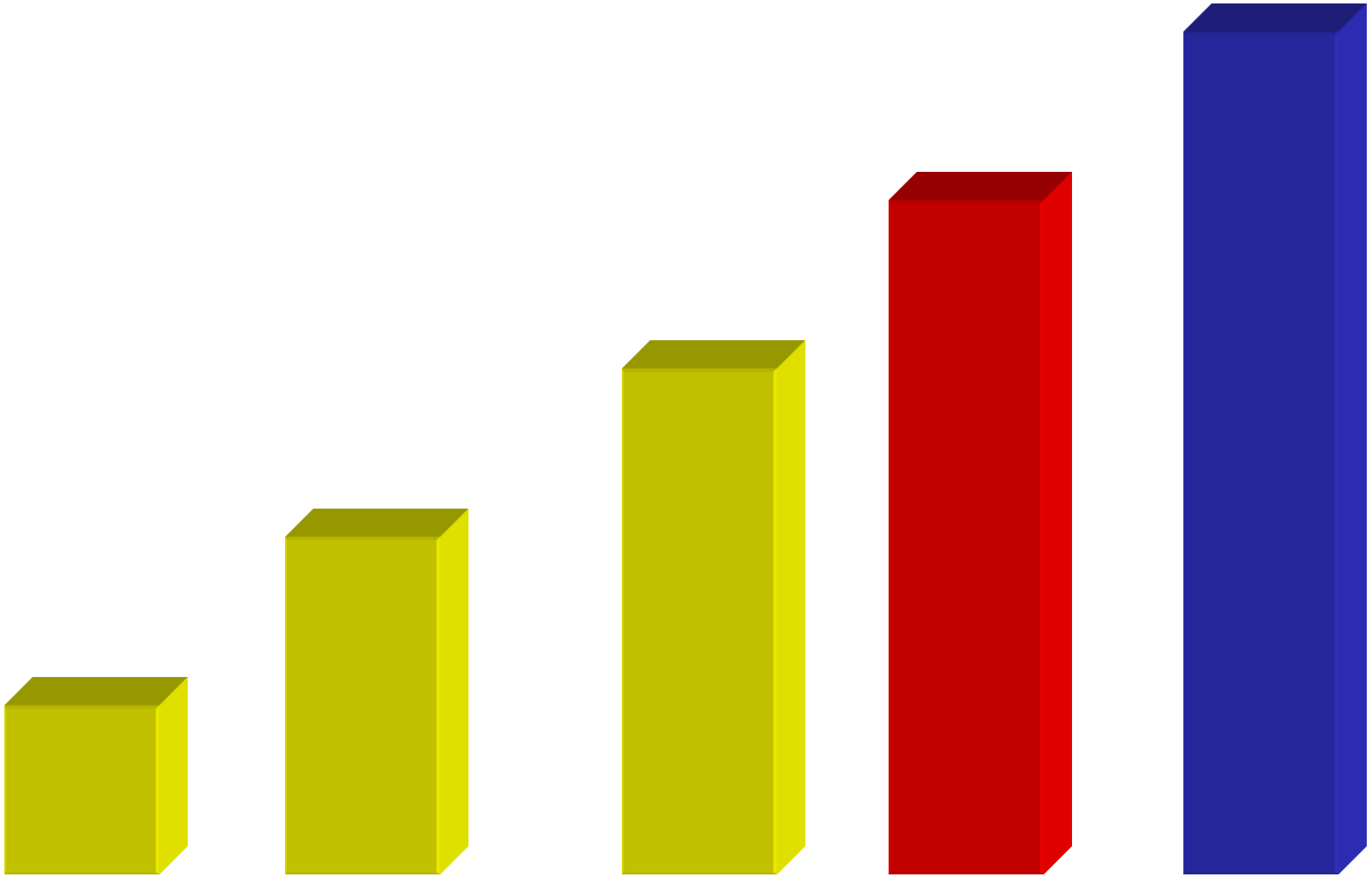
Selection Sort Animation



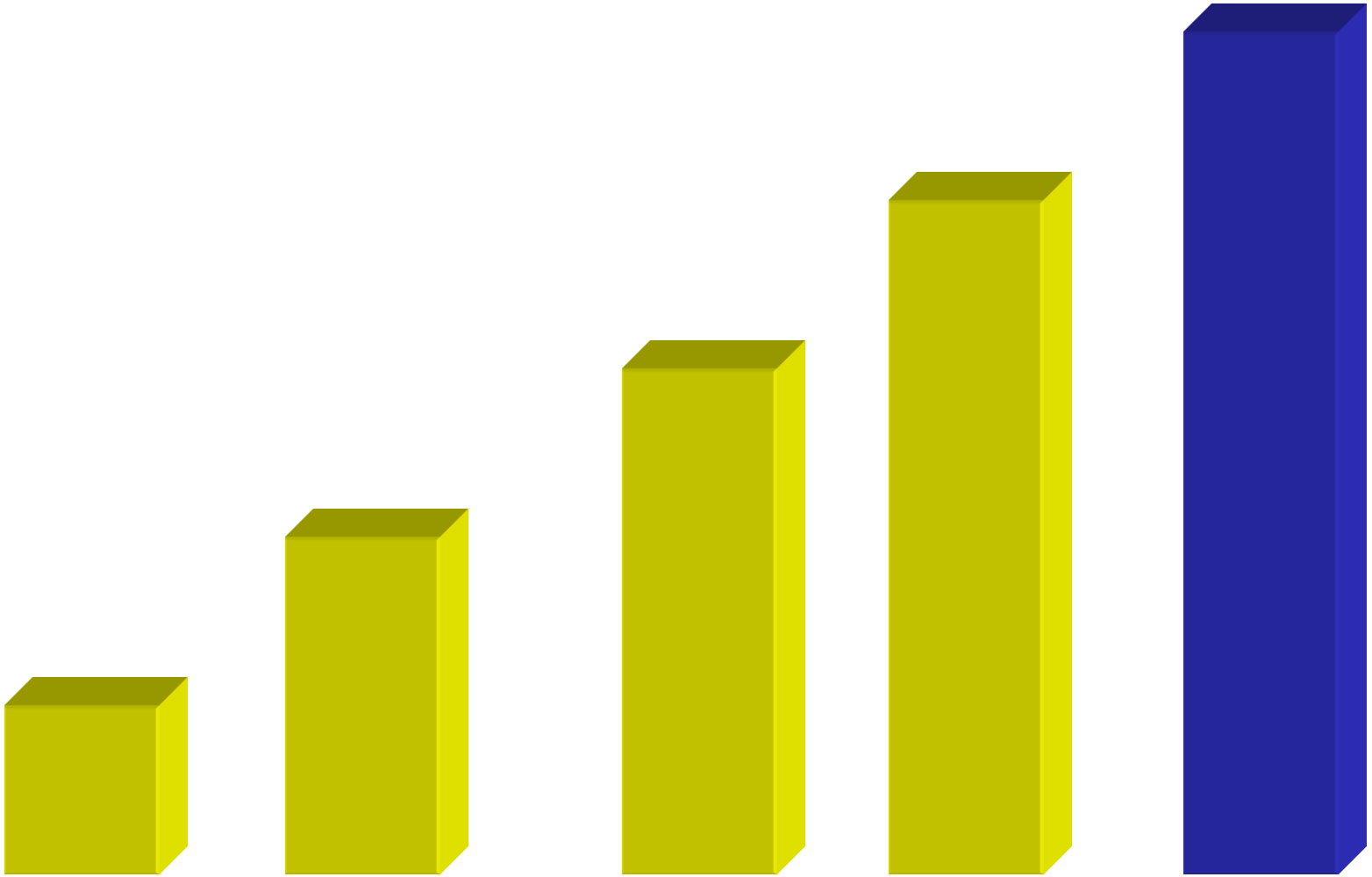
Selection Sort Animation



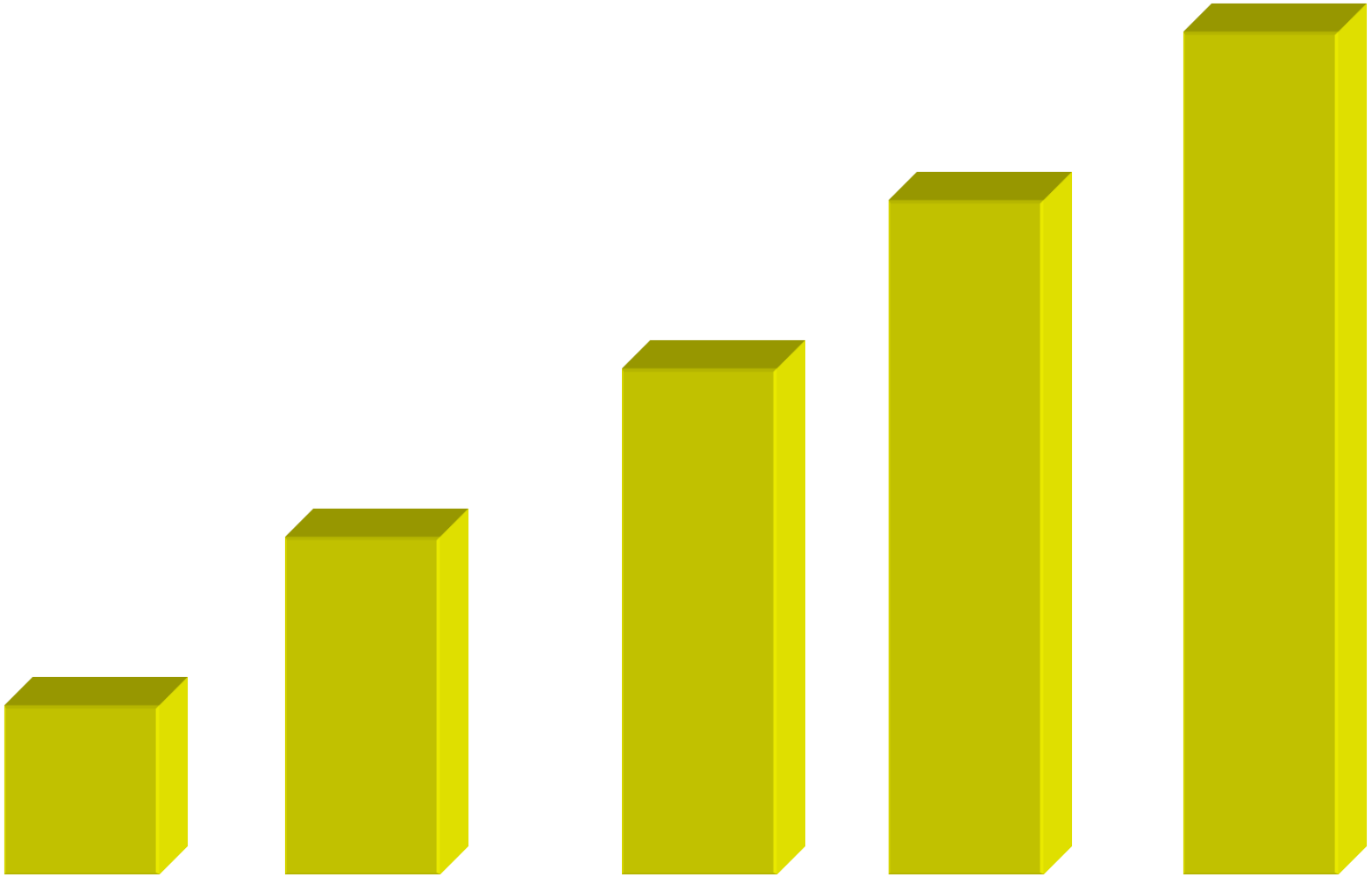
Selection Sort Animation



Selection Sort Animation



Selection Sort Animation



What is the Worst-case Running Time of Selection Sort?

for $k \leftarrow 0$ **to** $n-2$ **do**
    Find the smallest item $A[j]$ between $A[k]$ and $A[n-1]$
    swap $A[k]$ and $A[j]$
end

for $k \leftarrow 0$ **to** $n-2$ **do**
    min $\leftarrow k$
    for $j \leftarrow k+1$ **to** $n-1$ **do**
        if $A[j] < A[min]$ **then** $min \leftarrow j$ **end**
    end
    swap($A[k]$, $A[min]$)
end

What is the Worst-case Running Time of Selection Sort?

$$T(n) = \sum_{k=0}^{n-2} \sum_{j=k+1}^{n-1} \text{cmps} = \sum_{k=0}^{n-2} \sum_{j=k+1}^{n-1} 1 = \sum_{k=0}^{n-2} ((n-1)-(k+1)+1)$$

$$T(n) = \sum_{k=0}^{n-2} (n-1-k) = \sum_{k=0}^{n-2} n - \sum_{k=0}^{n-2} 1 - \sum_{k=0}^{n-2} k$$

$$T(n) = (n-1)n - (n-1) - \frac{(n-2)(n-1)}{2}$$

$$T(n) = \frac{2(n-1)n - 2(n-1) - (n-2)(n-1)}{2}$$

$$T(n) = \frac{(n-1)(2n-2-n+2)}{2} = \frac{(n-1)n}{2} \in O(n^2)$$

What is the Worst-case Running Time of Bubble Sort?

for $k \leftarrow 0$ **to** $n–2$ **do**

Bubble up the largest element to final position by swapping adjacent elements if they are out of place

end

for $k \leftarrow 0$ **to** $n–2$ **do**

min $\leftarrow k$

for $j \leftarrow 0$ **to** $n–2–k$ **do**

if $A[j+1] < A[j]$ **then** swap($A[j]$, $A[j+1]$) **end**

end

end

What is the Worst-case Running Time of Bubble Sort?

$$T(n) = \sum_{k=0}^{n-2} \sum_{j=0}^{n-2-k} \text{cmps} = \sum_{k=0}^{n-2} \sum_{j=0}^{n-2-k} 1 = \sum_{k=0}^{n-2} (n-2-k+1)$$

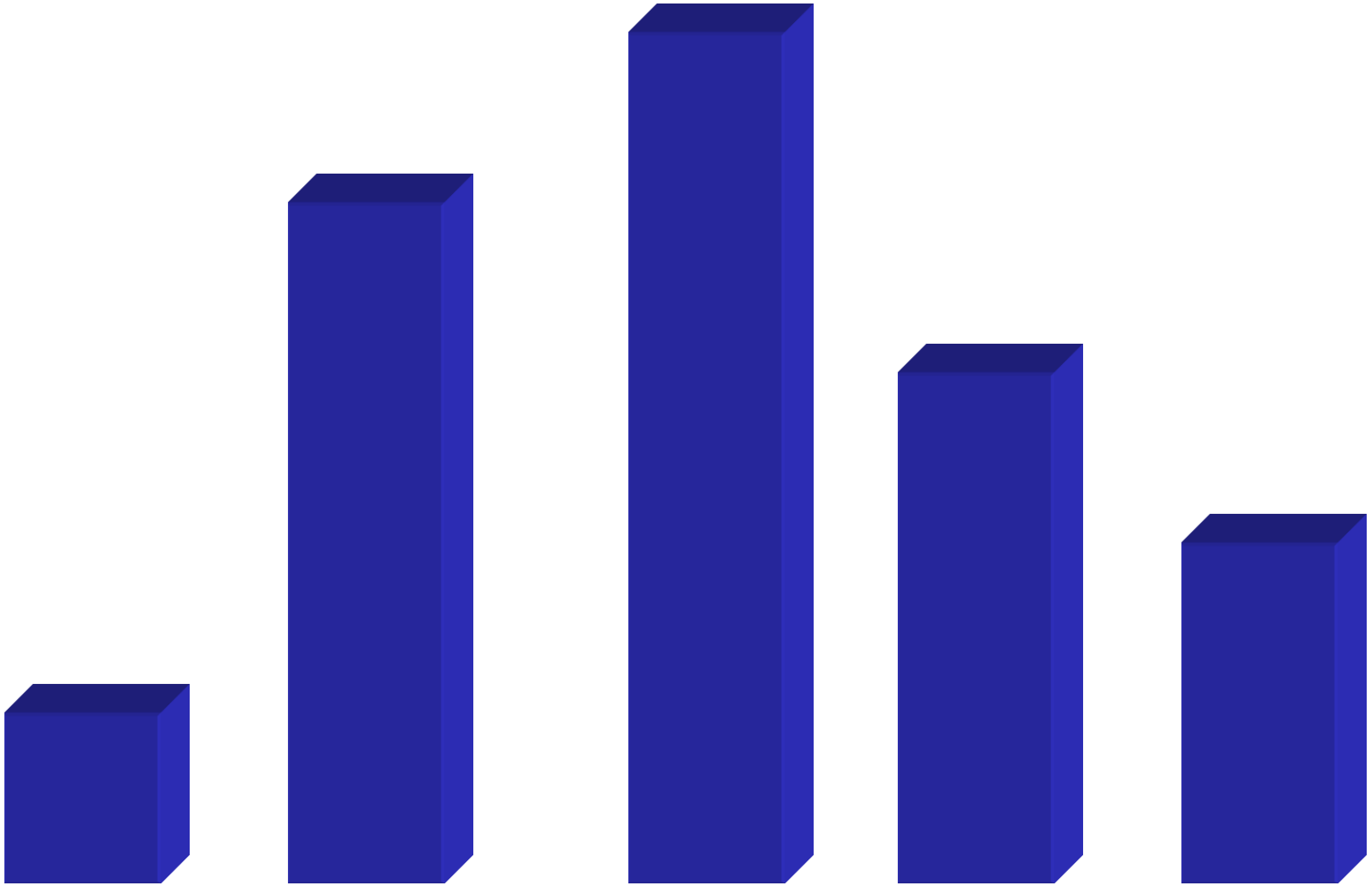
$$T(n) = \sum_{k=0}^{n-2} (n-1-k) = \sum_{k=0}^{n-2} n - \sum_{k=0}^{n-2} 1 - \sum_{k=0}^{n-2} k$$

$$T(n) = (n-1)n - (n-1) - \frac{(n-2)(n-1)}{2}$$

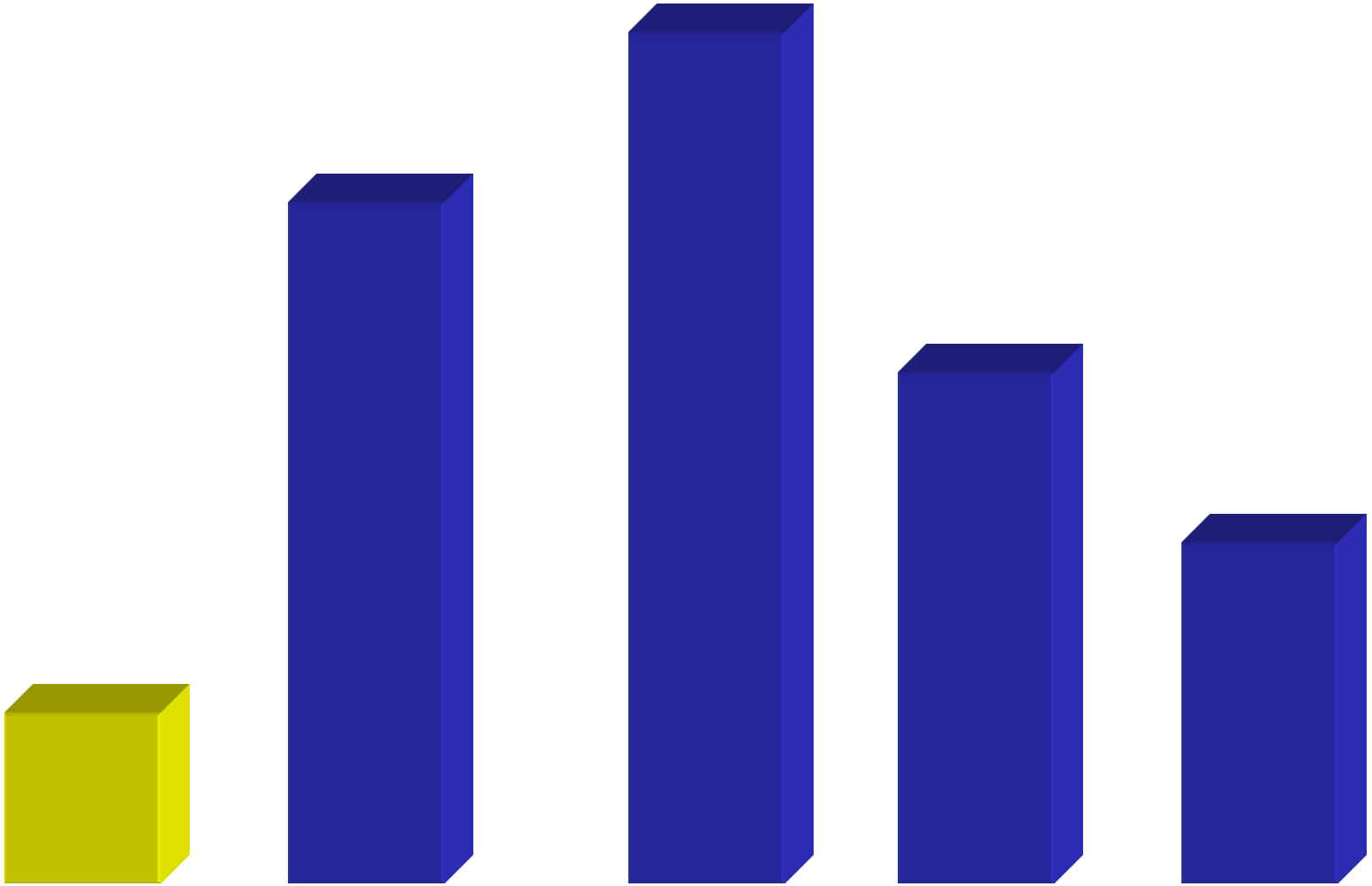
$$T(n) = \frac{2(n-1)n - 2(n-1) - (n-2)(n-1)}{2}$$

$$T(n) = \frac{(n-1)(2n-2-n+2)}{2} = \frac{(n-1)n}{2} \in O(n^2)$$

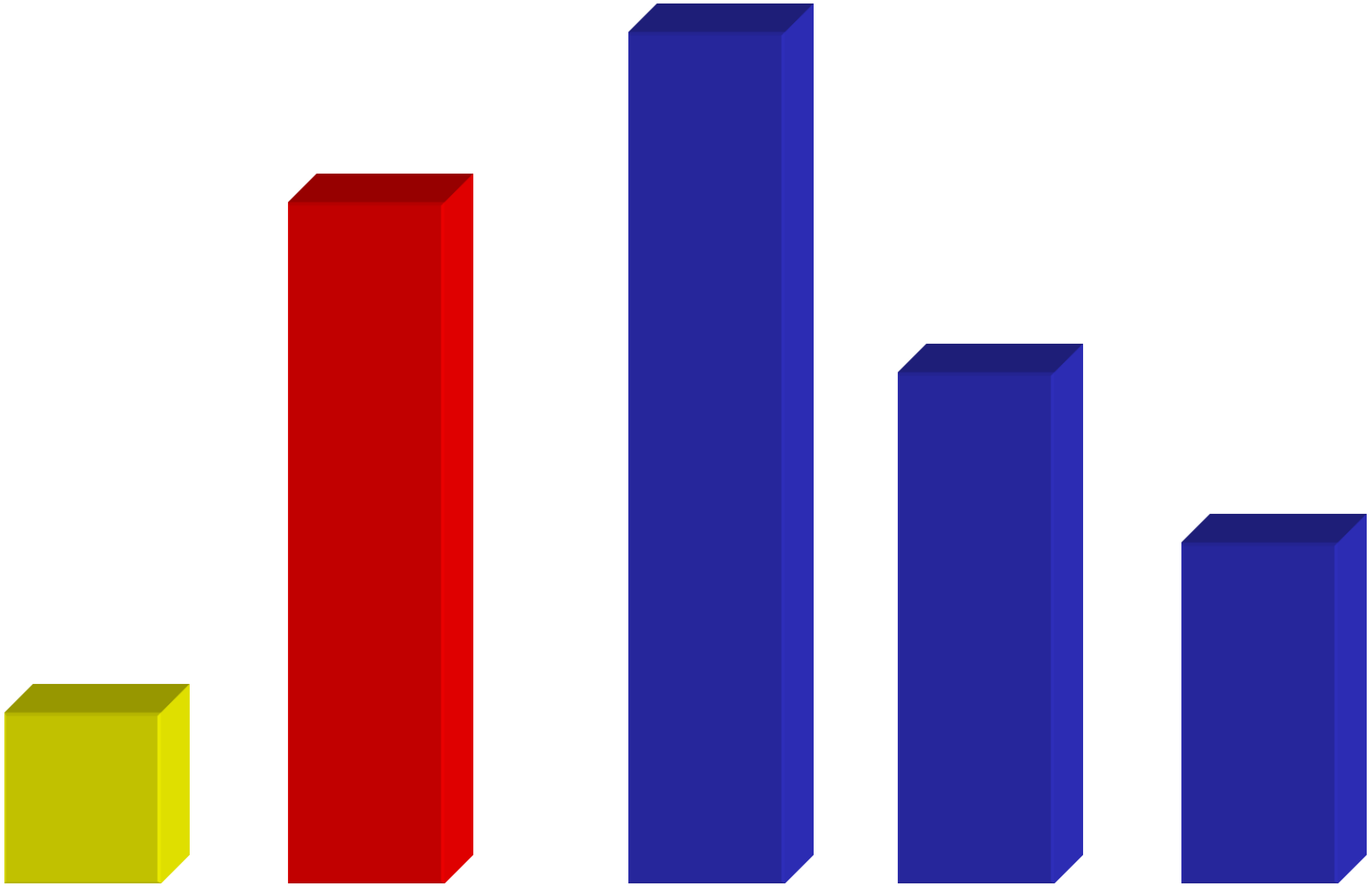
Insertion Sort



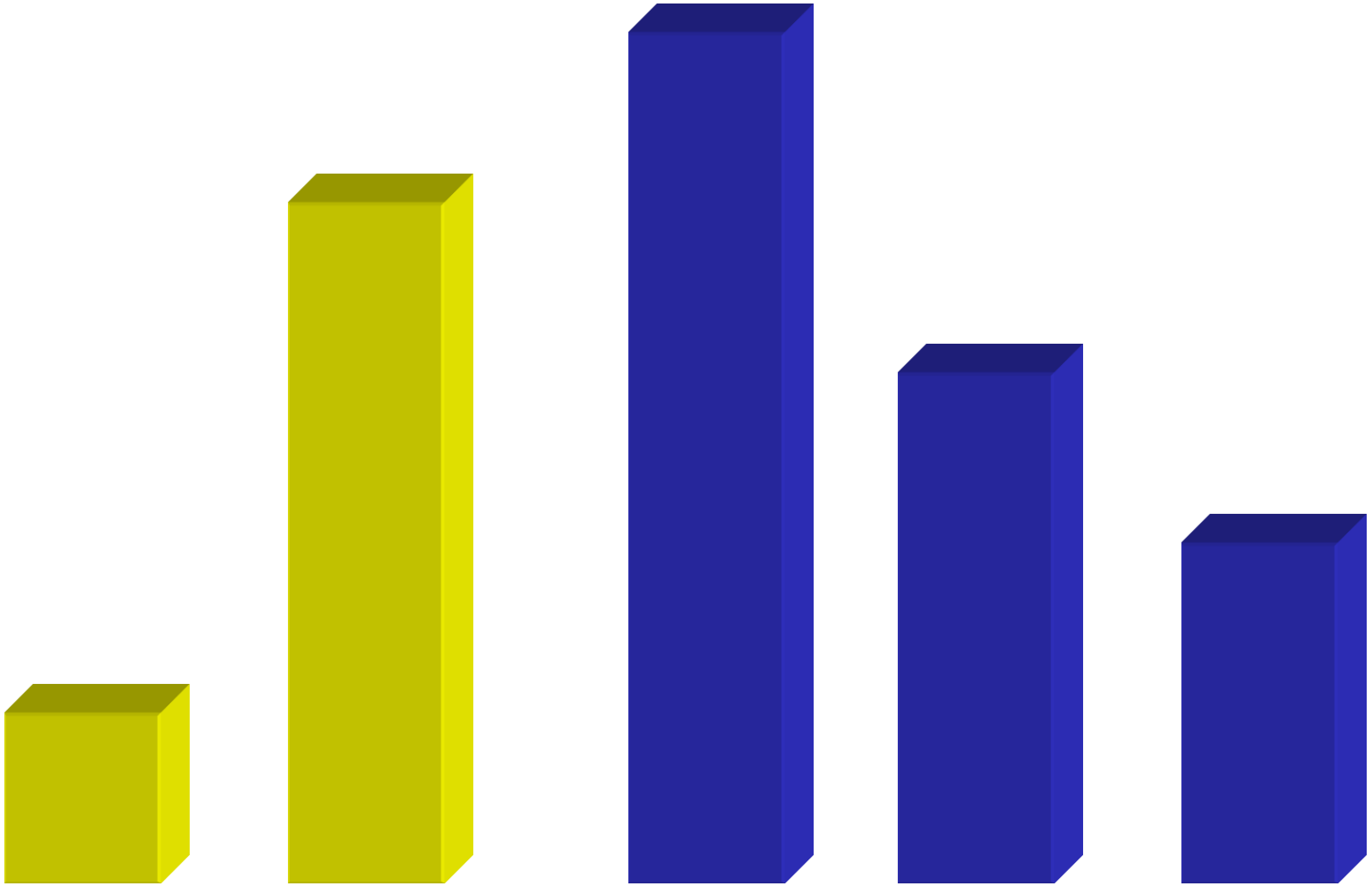
Insertion Sort



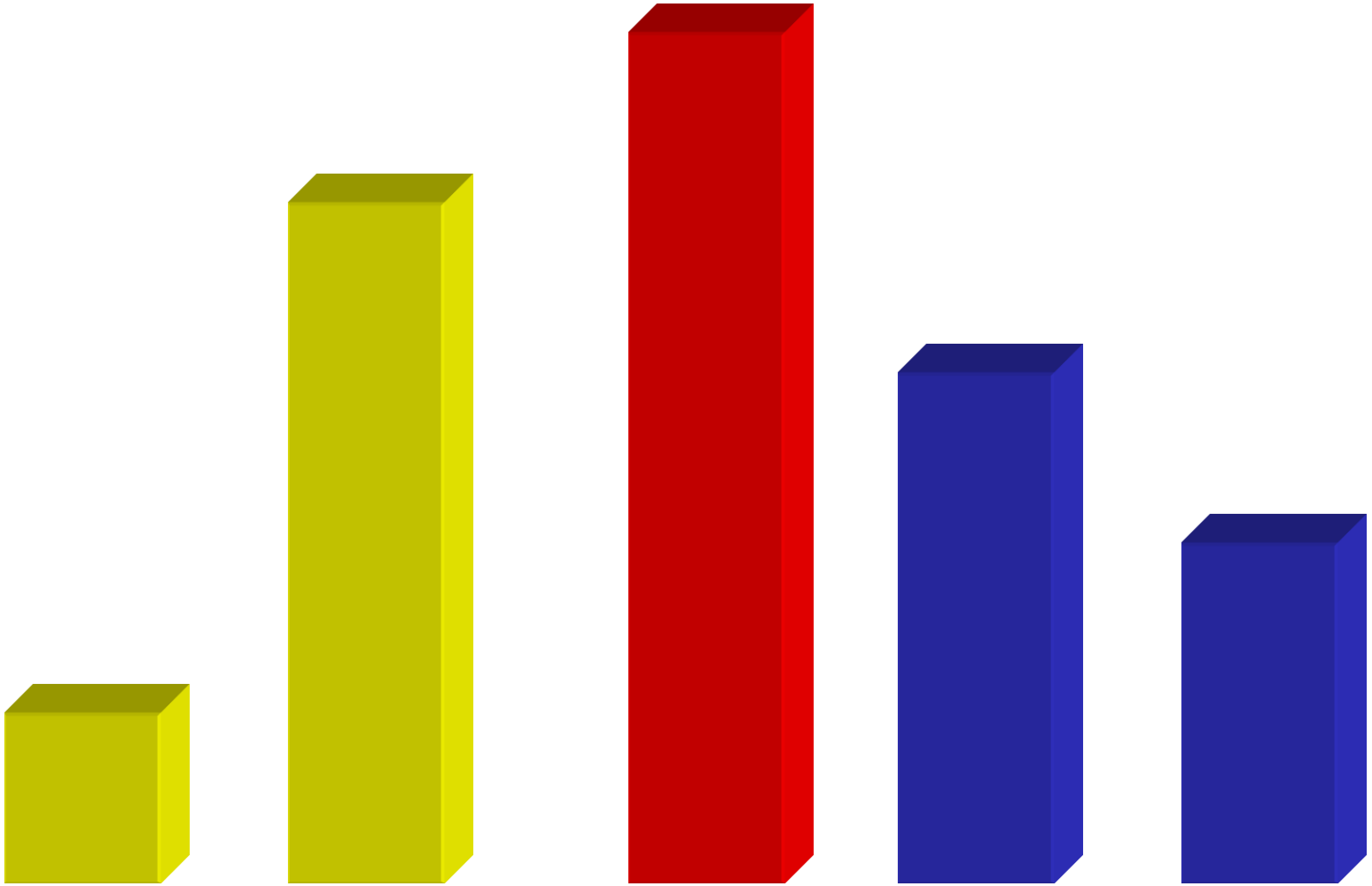
Insertion Sort



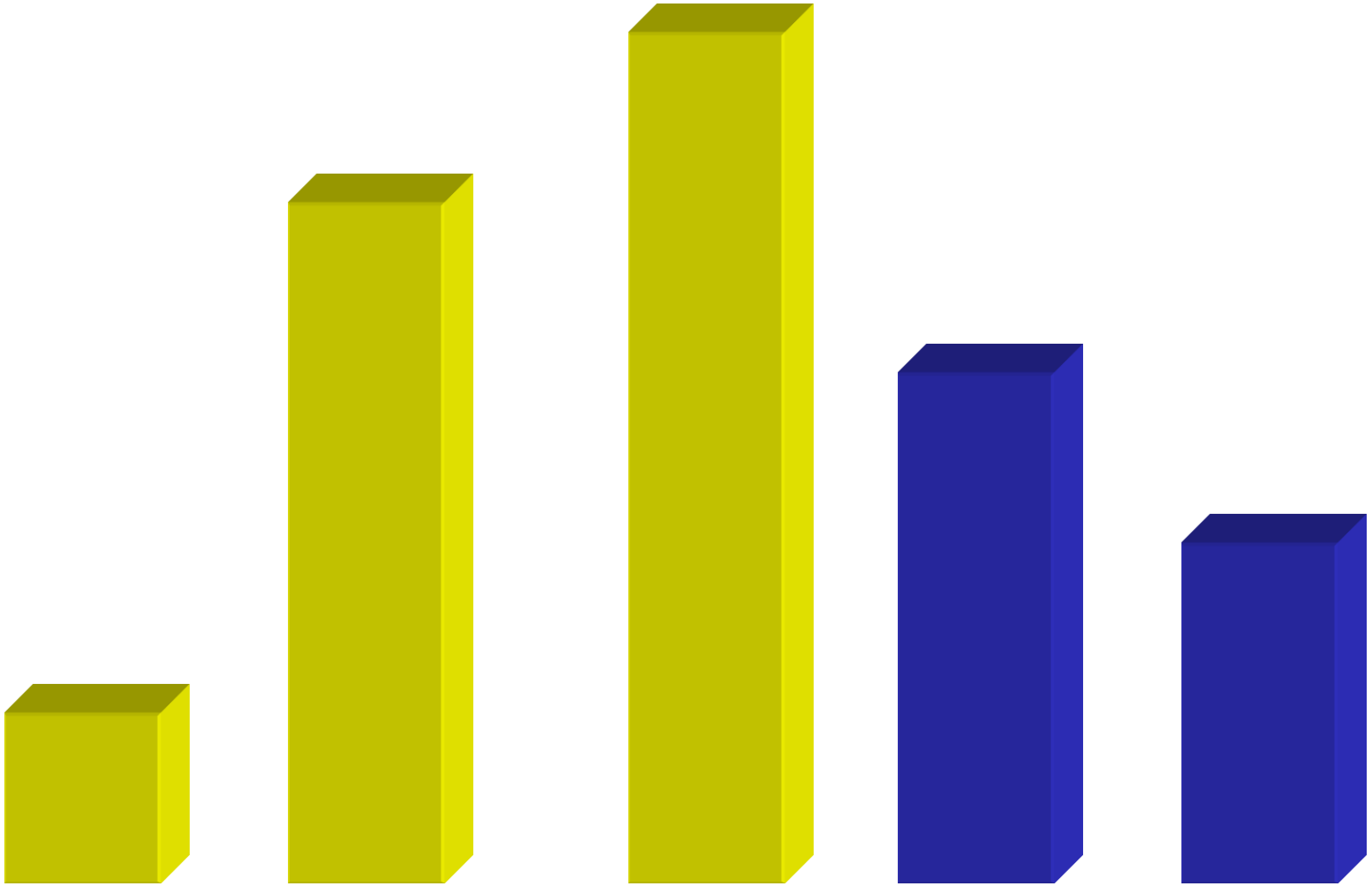
Insertion Sort



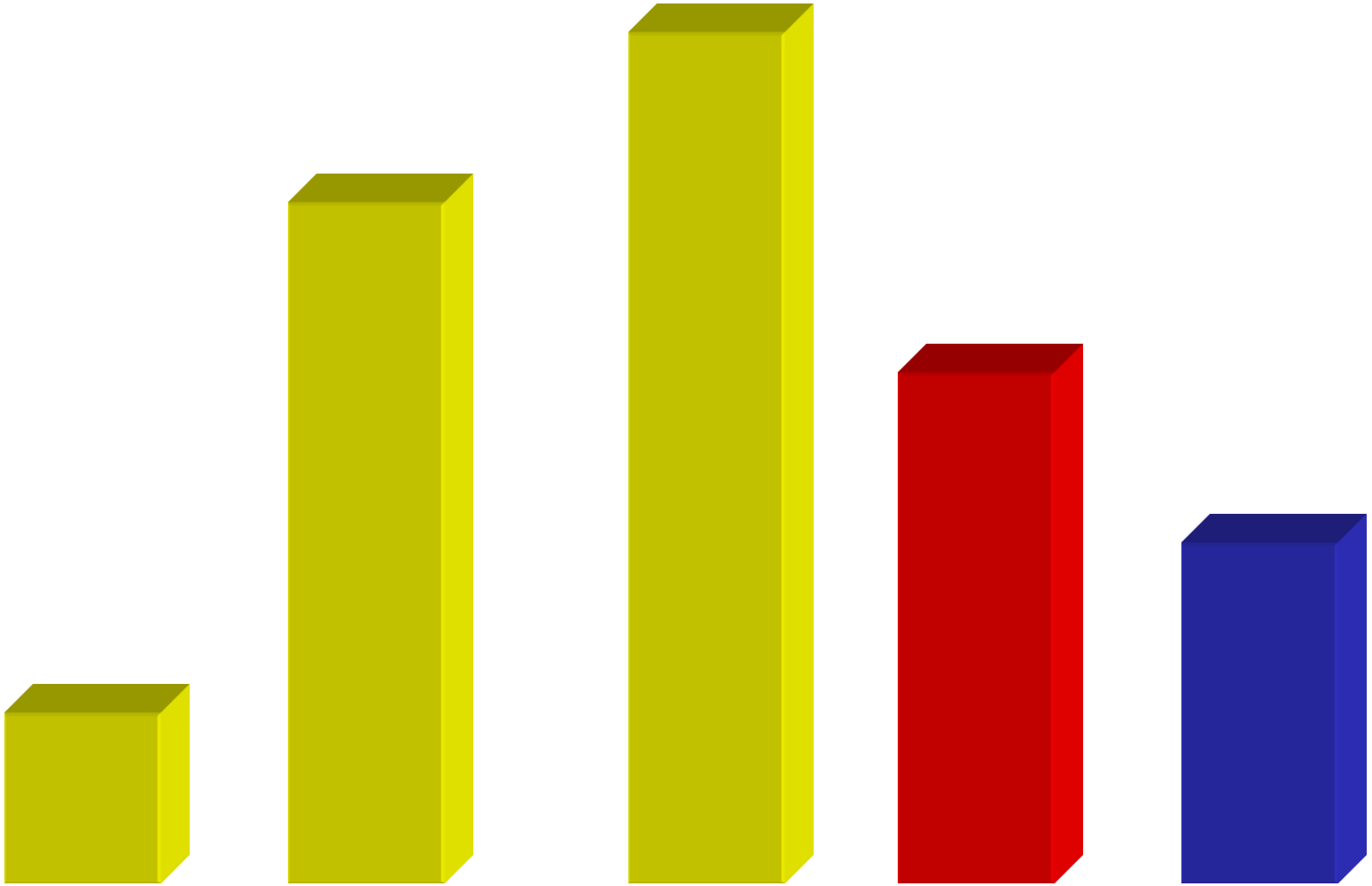
Insertion Sort



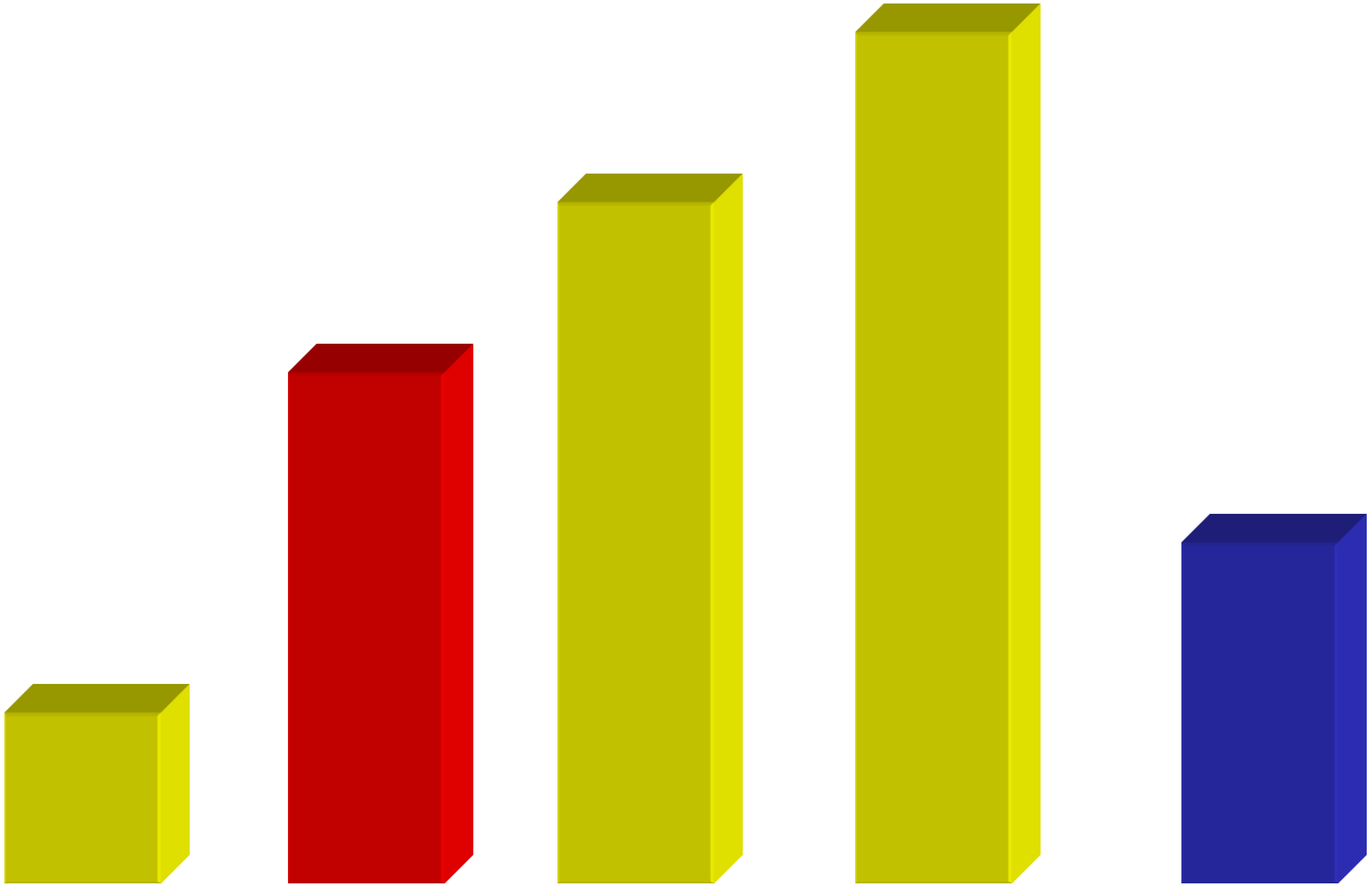
Insertion Sort



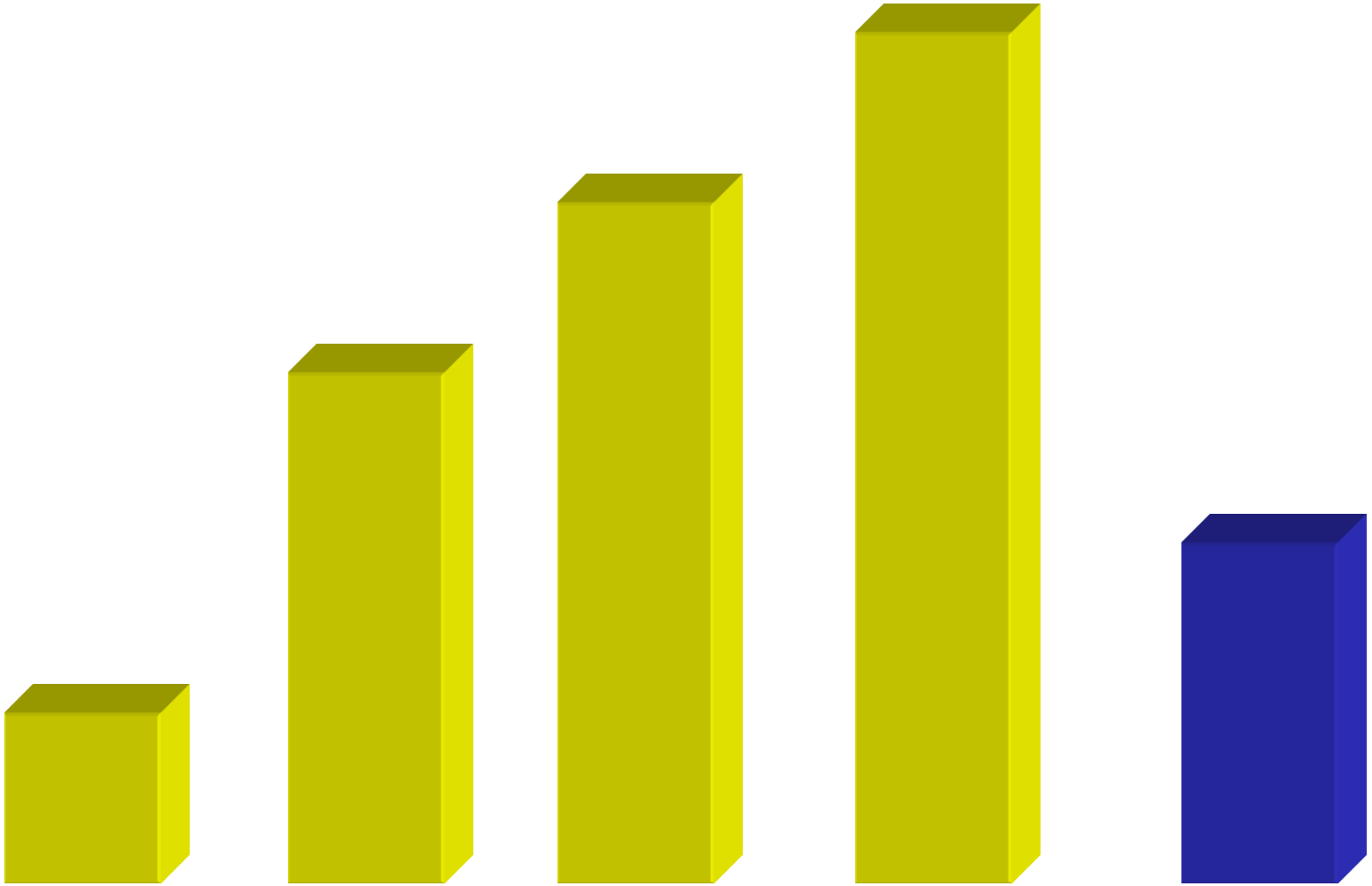
Insertion Sort



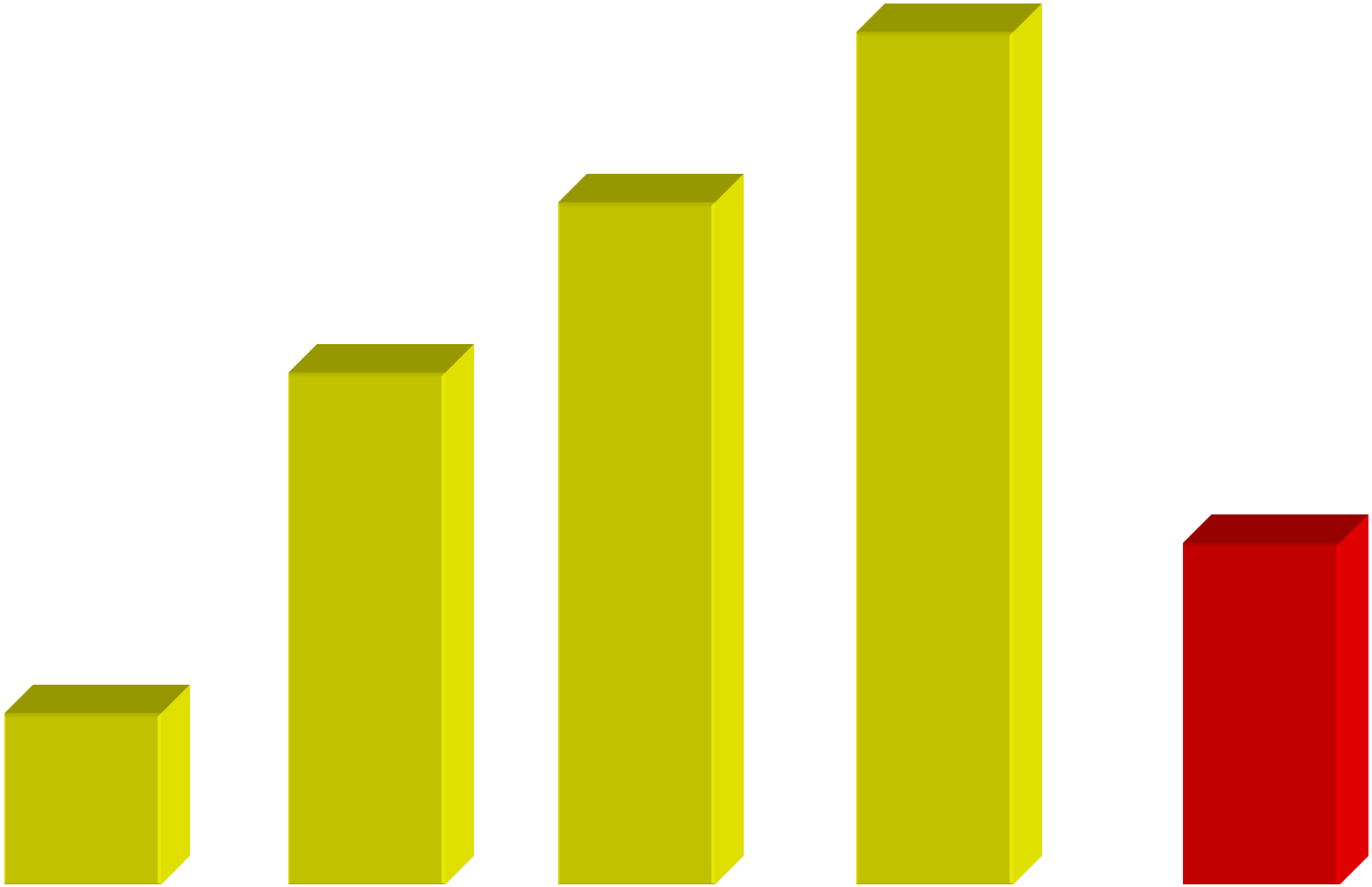
Insertion Sort



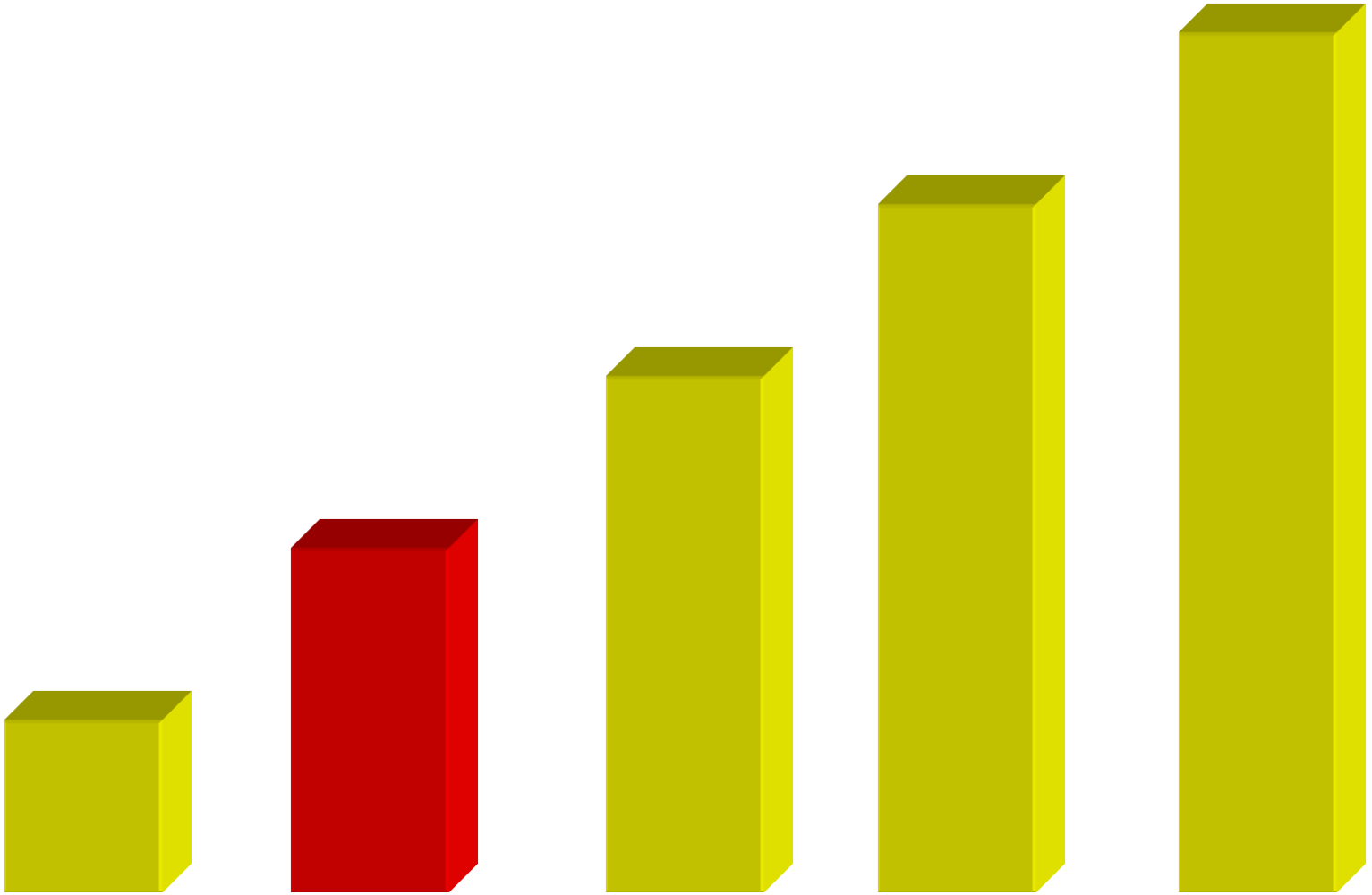
Insertion Sort



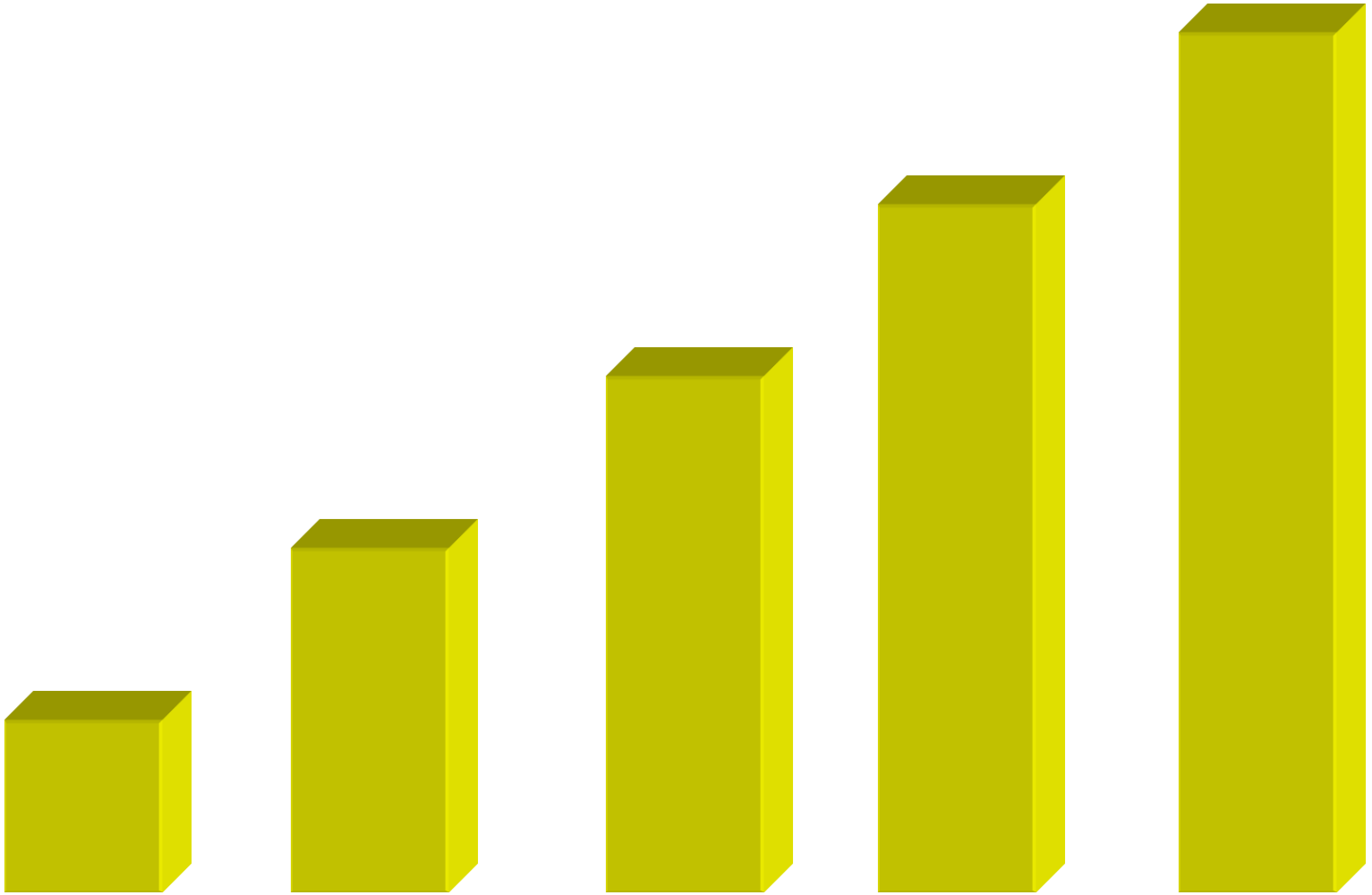
Insertion Sort



Insertion Sort



Insertion Sort



What is the Worst-case Running Time of Insertion Sort?

for $k \leftarrow 1$ **to** n-1 **do**
    Move $A[k]$ forward to position j ≤ k such that
    $A[k] < A[t]$ for $j \leq t < k$ and
    either $A[k] \geq A[j-1]$ or $j = 1$
end

for $k \leftarrow 1$ **to** n-1 **do**
    val $\leftarrow A[k]$
    $j \leftarrow k-1$
    while $j \geq 0$ **and** $A[j] >$ val **do**
        $A[j+1] \leftarrow A[j]$
        $j \leftarrow j - 1$
    end
    $A[j+1]$ = val
end

What is the Worst-case Running Time of Insertion Sort?

$$T(n) = \sum_{k=1}^{n-1} \sum_{j=0}^{k-1} cmps = \sum_{k=1}^{n-1} \sum_{j=0}^{k-1} 1 =$$

$$T(n) = \sum_{k=1}^{n-1} k = \frac{(n-1)n}{2} \in O(n^2)$$