CSC 225

Algorithms and Data Structures I

Rich Little
rlittle@uvic.ca
ECS 516

Sorting

- **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

Partial Order

Partial Order

A relation, $\leq$, on a set A is called a partial order if $\leq$ is

i. **Reflexive:** For all $k \in A, k \leq k$.

ii. **Antisymmetric:** For all $k_1, k_2 \in A$, if $k_1 \leq k_2$ and $k_2 \leq k_1$, then $k_1 = k_2$.

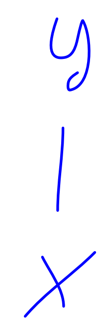
iii. **Transitive:** For all $k_1, k_2, k_3 \in A$, if $k_1 \leq k_2$ and $k_2 \leq k_3$, then $k_1 \leq k_3$.

Partially Ordered Sets (Posets)

Partially Ordered Set

Let A be a set and $\leq$ a relation on A. The pair $(A, \leq)$ is called a <u>partially ordered set</u> (or <u>poset</u>) if $\leq$ on A is a partial order.

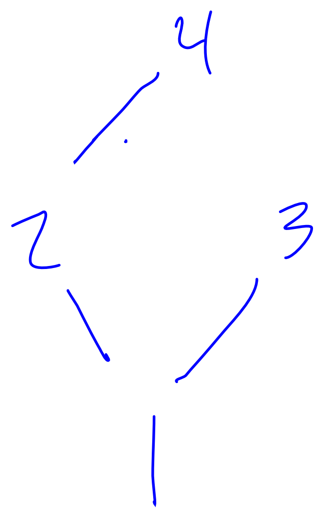
Hasse Diagram



If $\leq$ is a partial order on A, we construct a <u>Hasse diagram</u> for $\leq$ on A by connecting x "up" to y if and only if $x \leq y$ and there are no other $z \in A$ such that $x \leq z$ and $z \leq y$.

Example 1

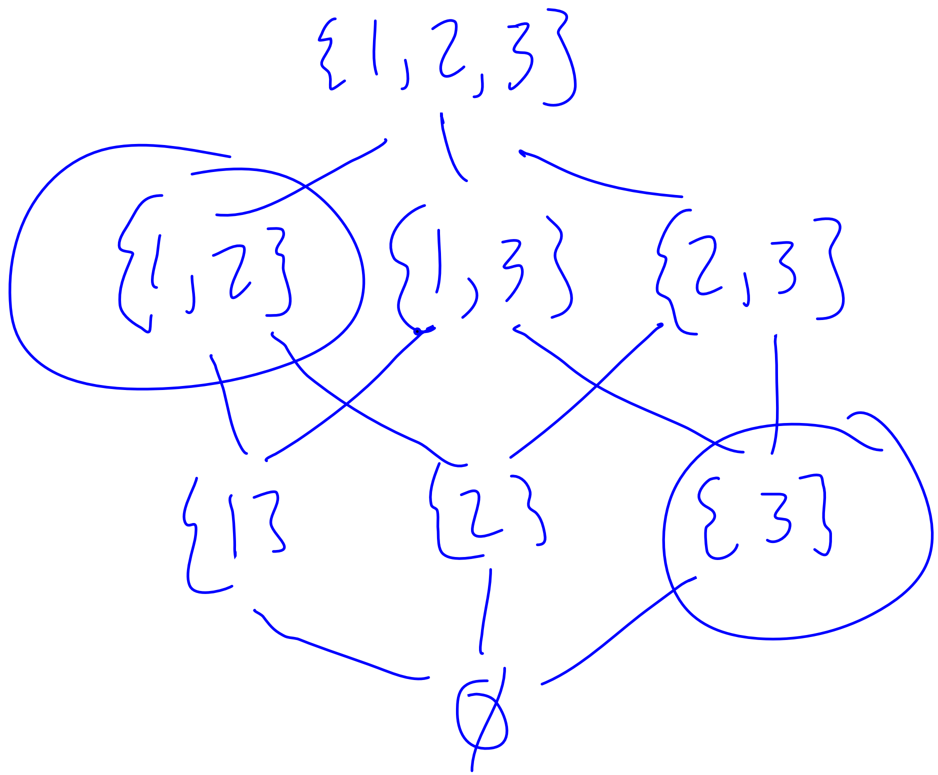
Let $A = \{1,2,3,4\}$ and define $\leq$ on A by $x \leq y$ if $x, y \in A$ and $x|y$. Draw a Hasse diagram for $\leq$.



Example 2

Consider the power set, $P(A)$, where $A = \{1,2,3\}$. Draw the Hasse diagram to illustrate the subset relation. $\preceq = \subseteq$

$P(A)$



Total Order

Total Order

If $(A, \leq)$ is a poset, it is a total order if for all $k_1, k_2 \in A$, either $k_1 \leq k_2$ or $k_2 \leq k_1$.

Examples

- Letters of the alphabet via lexicographic order.
- The set of real numbers by $\leq$ or $\geq$ relations.

Computational Problem: **Sorting**

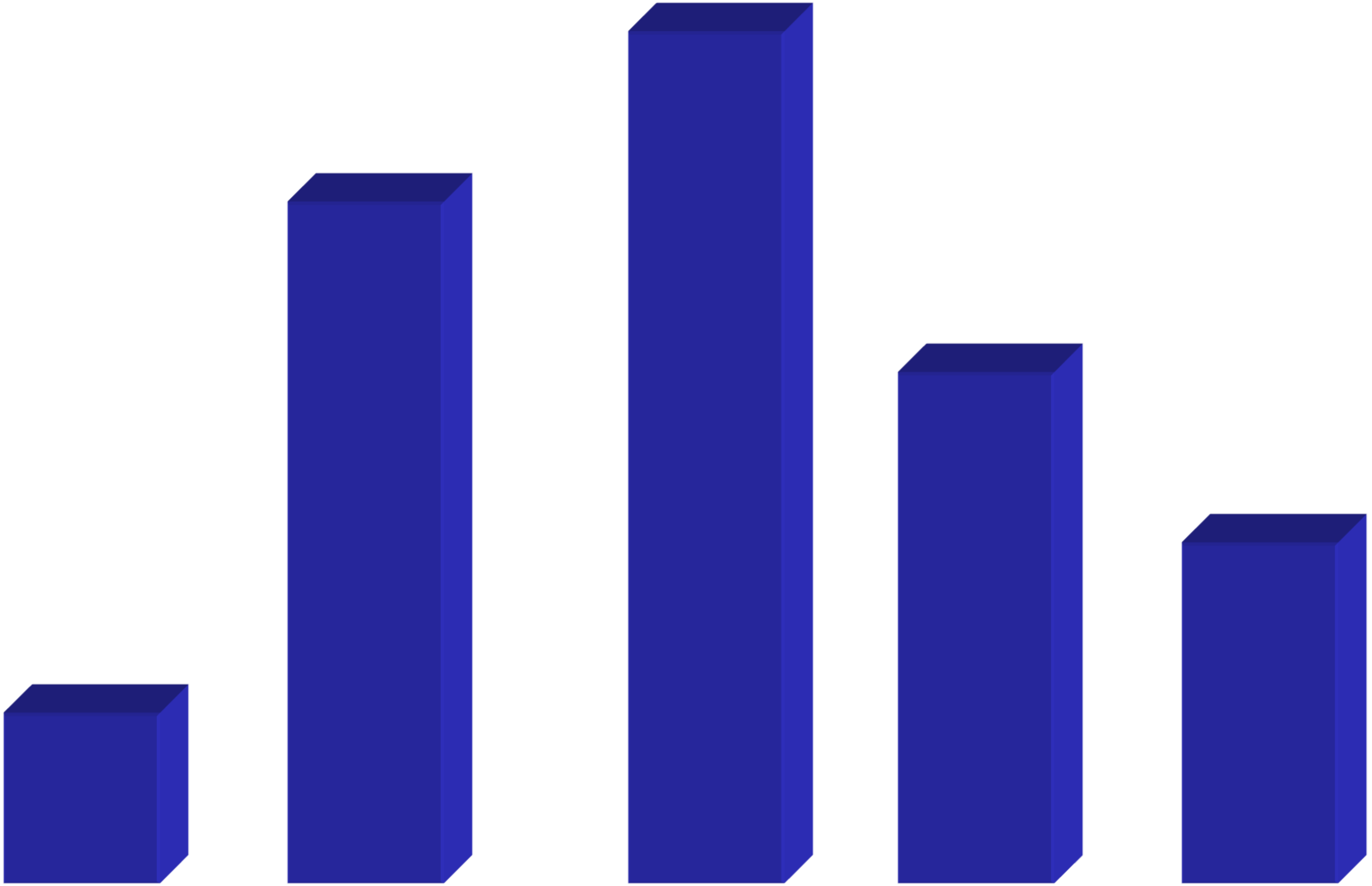
Input: A collection of *n* objects (stored in a data structure) and a comparator defining a total order on these objects

Output: Produce a linear ordered representation (ascending or descending) of these objects

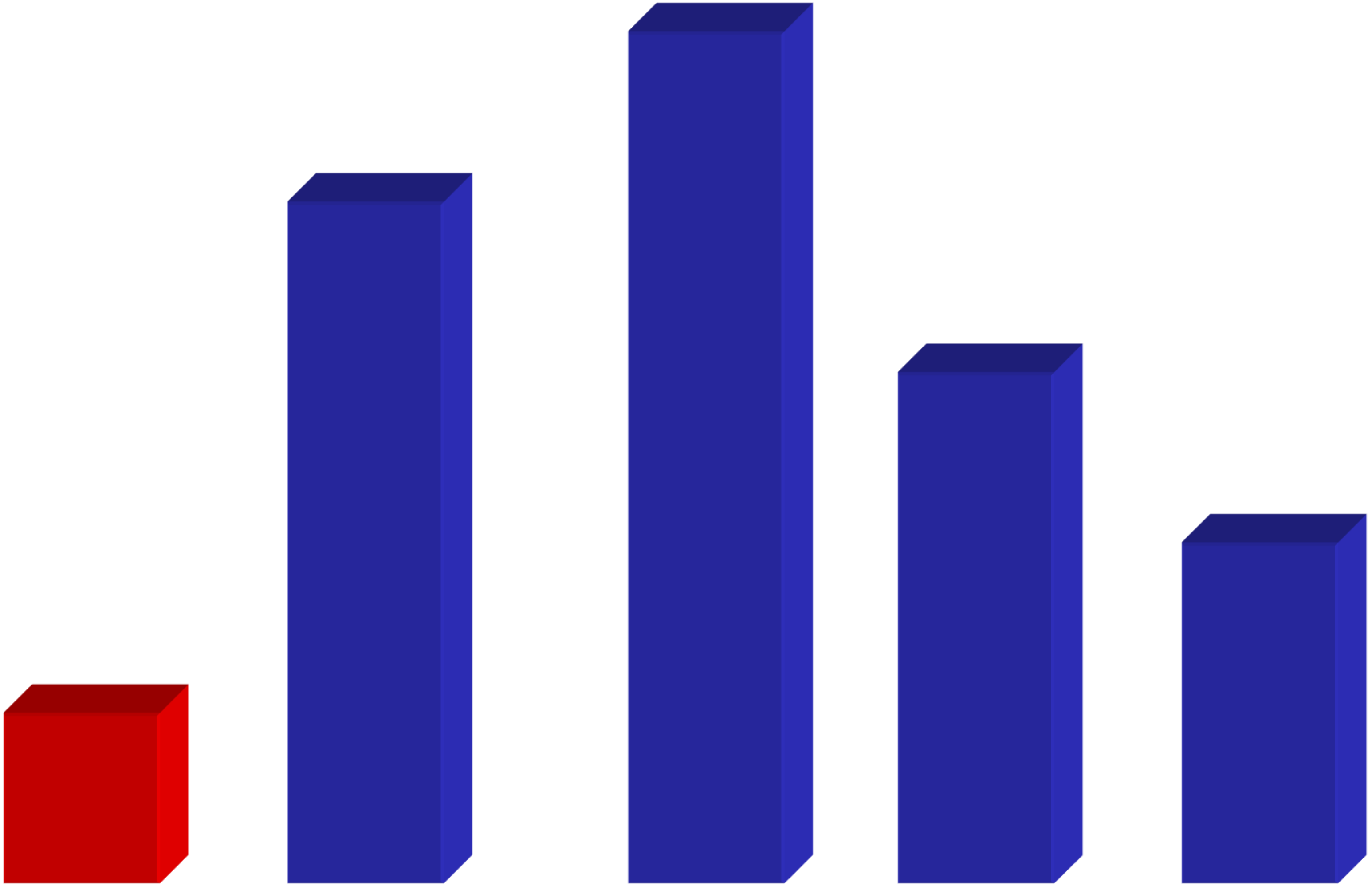
Algorithm Design Technique
Brute Force

- **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 algorithms
- Examples of the Brute Force technique
  - Selection sort
  - Bubble sort
  - Insertion sort

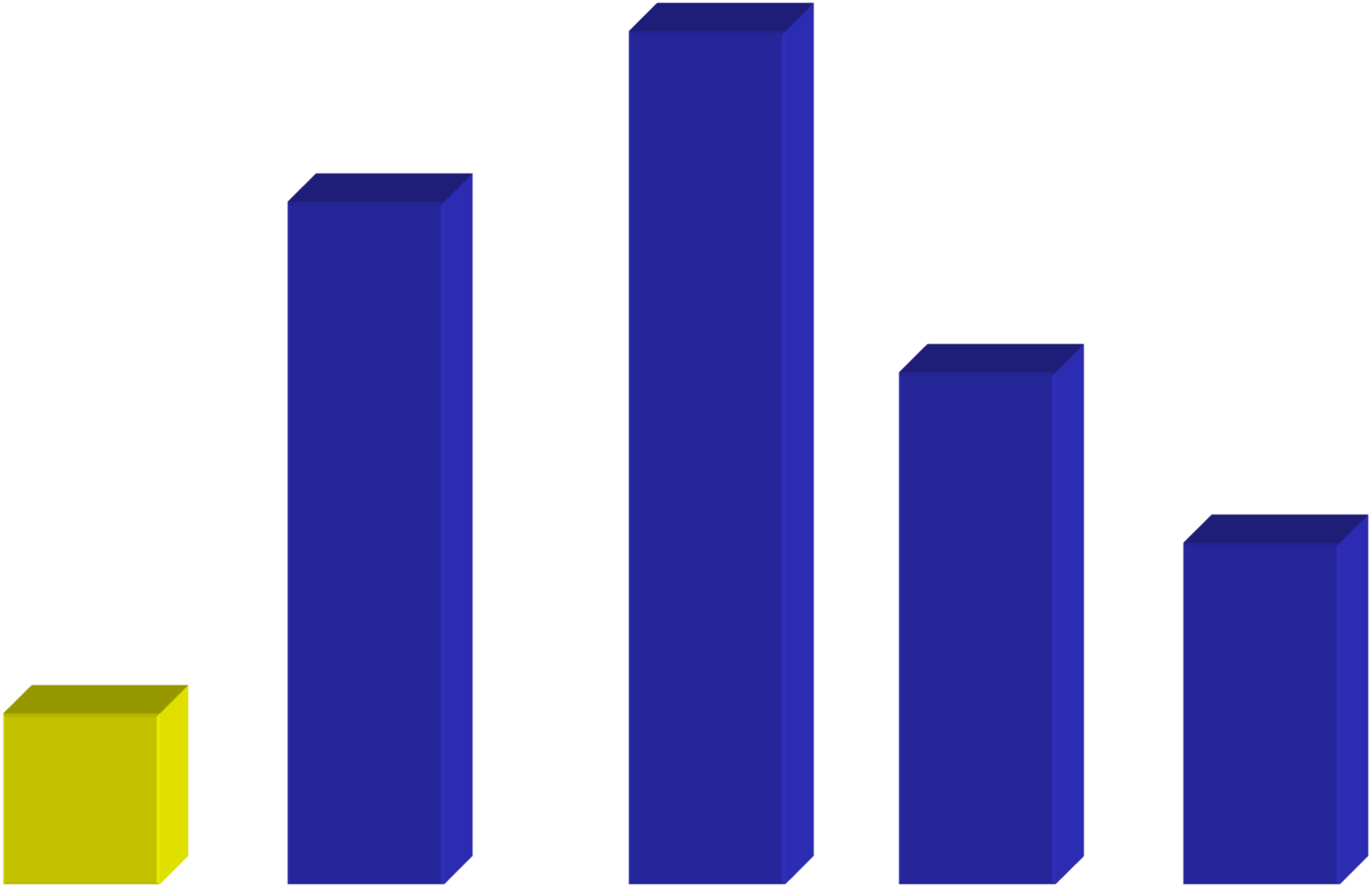
Selection Sort Animation



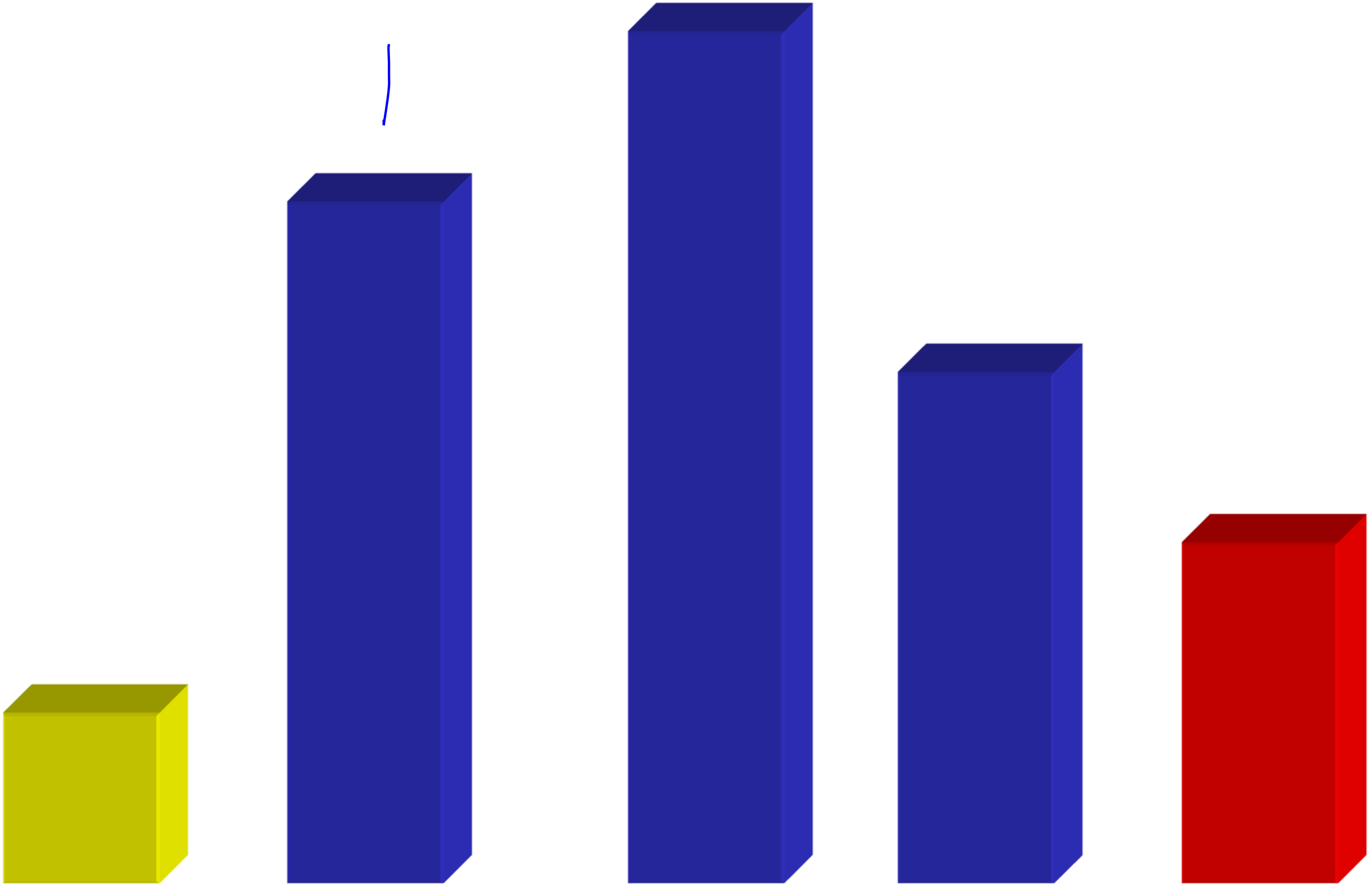
Selection Sort Animation



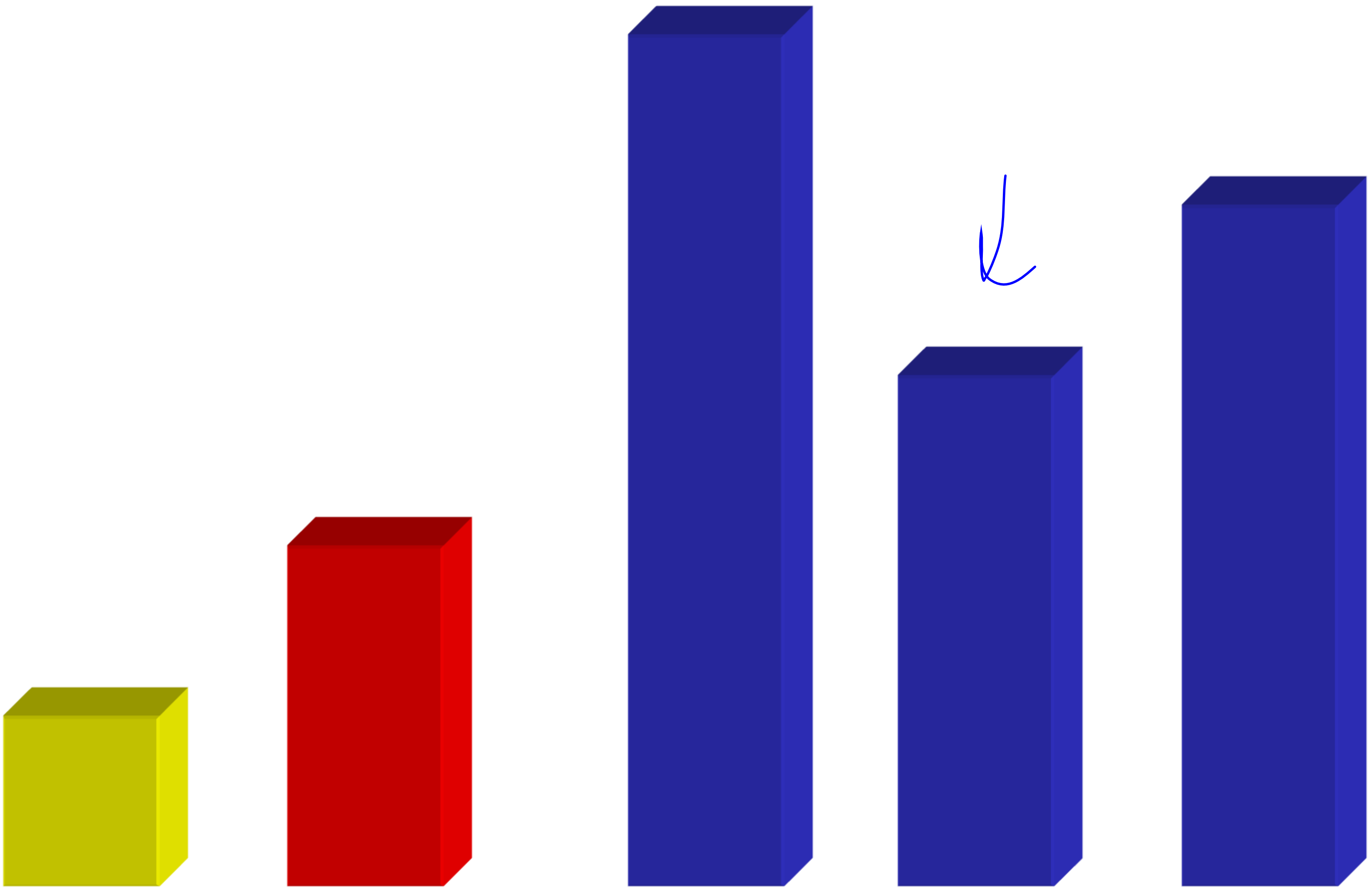
Selection Sort Animation



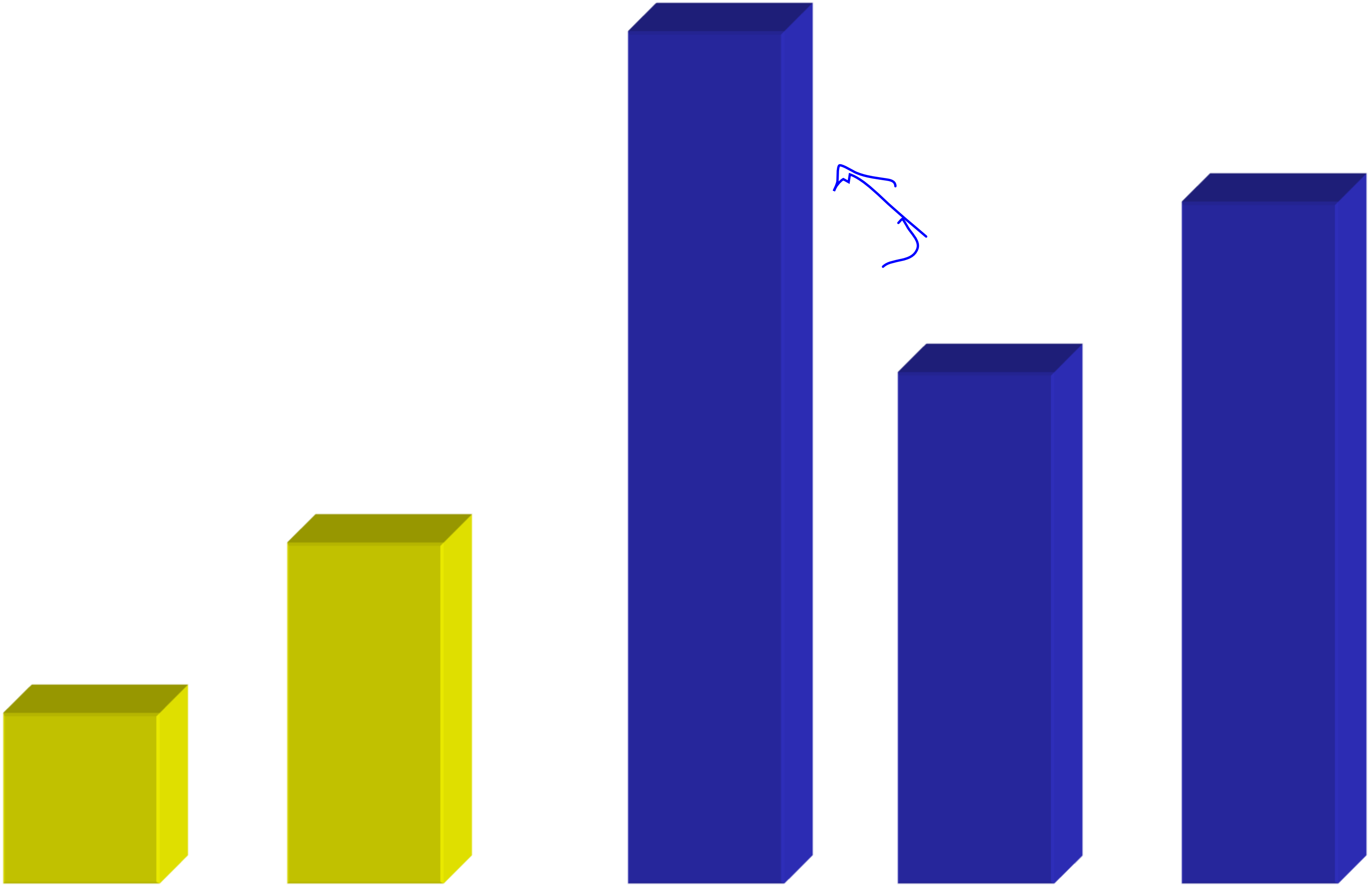
Selection Sort Animation



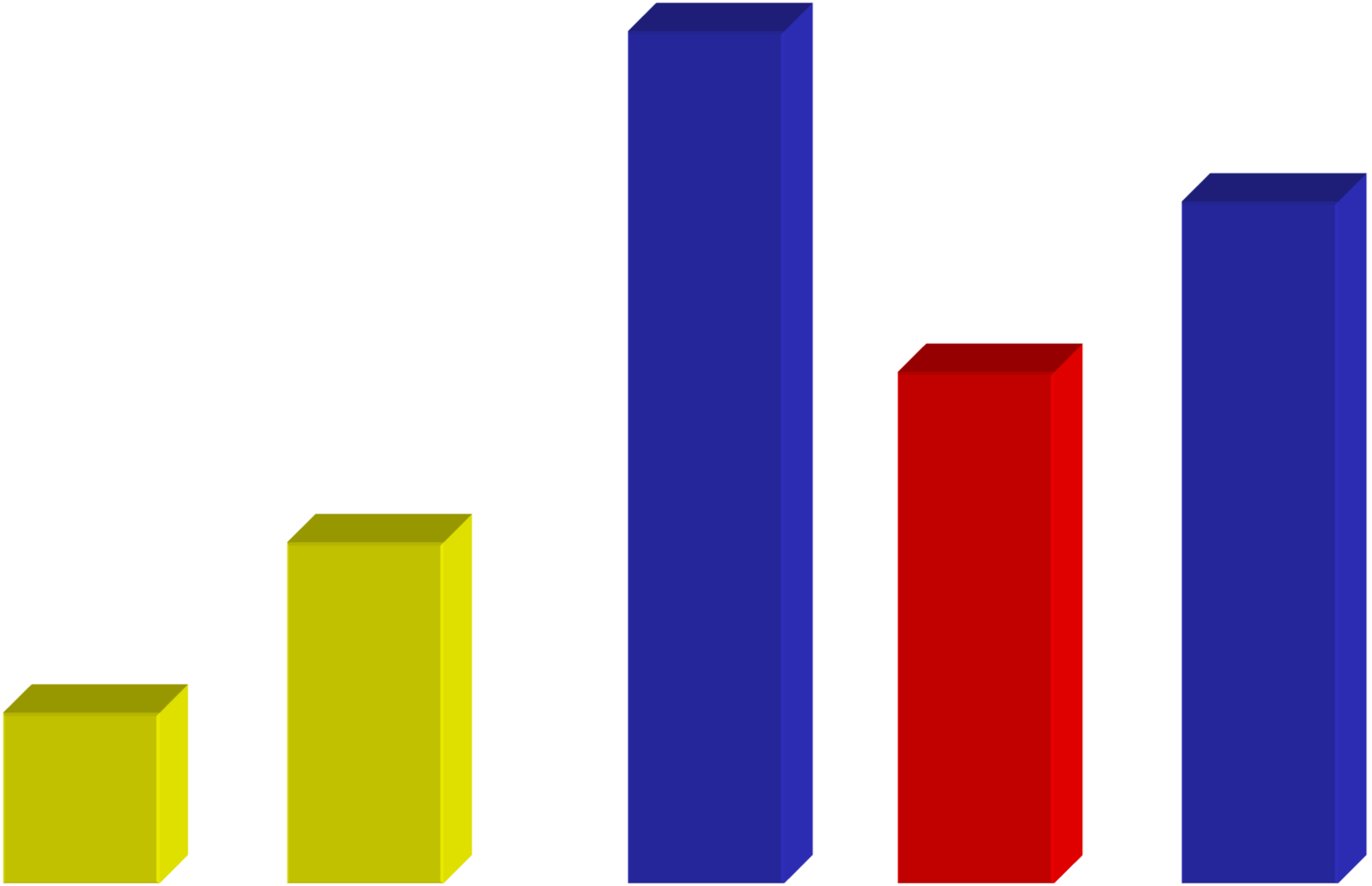
Selection Sort Animation



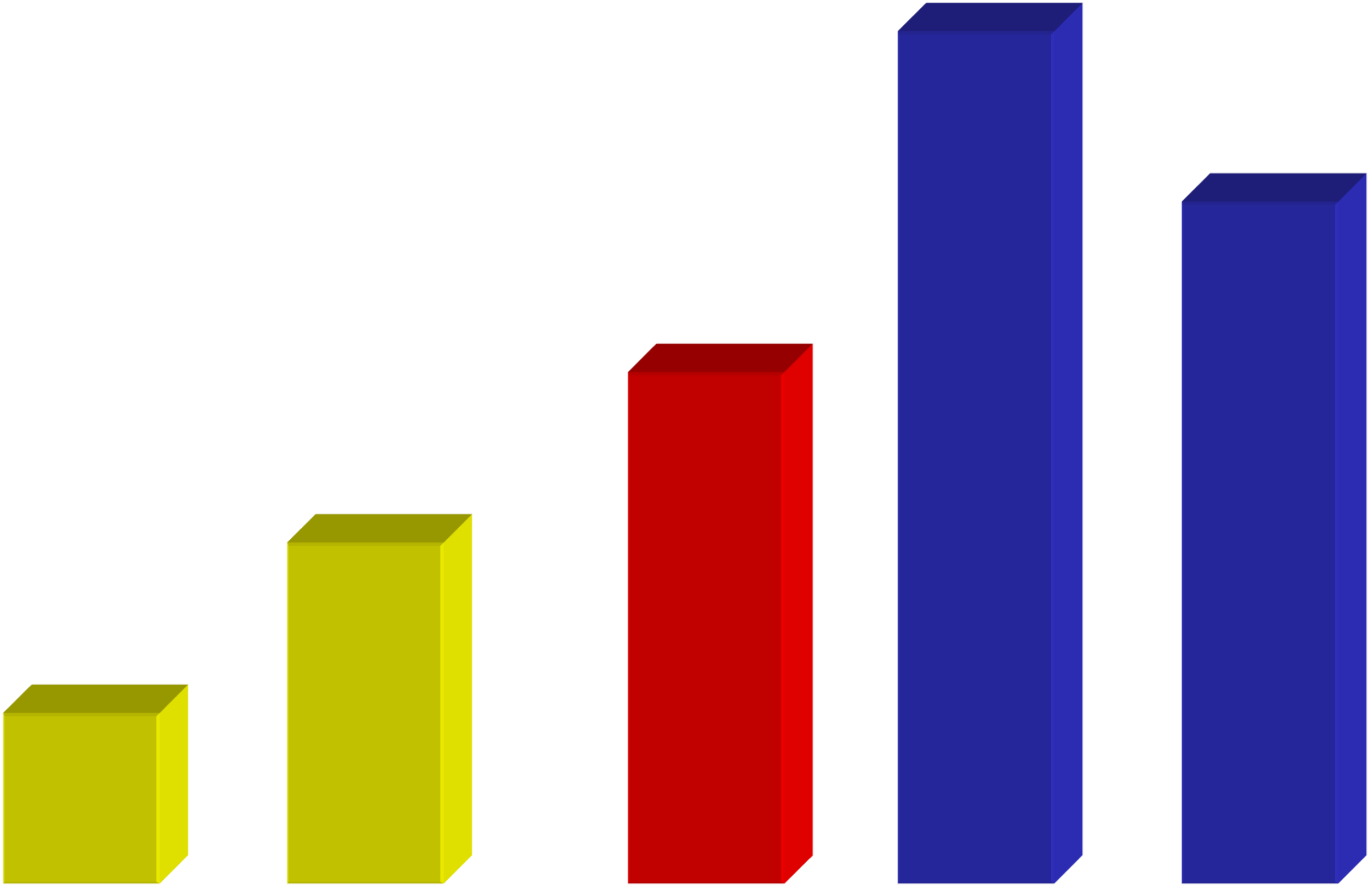
Selection Sort Animation



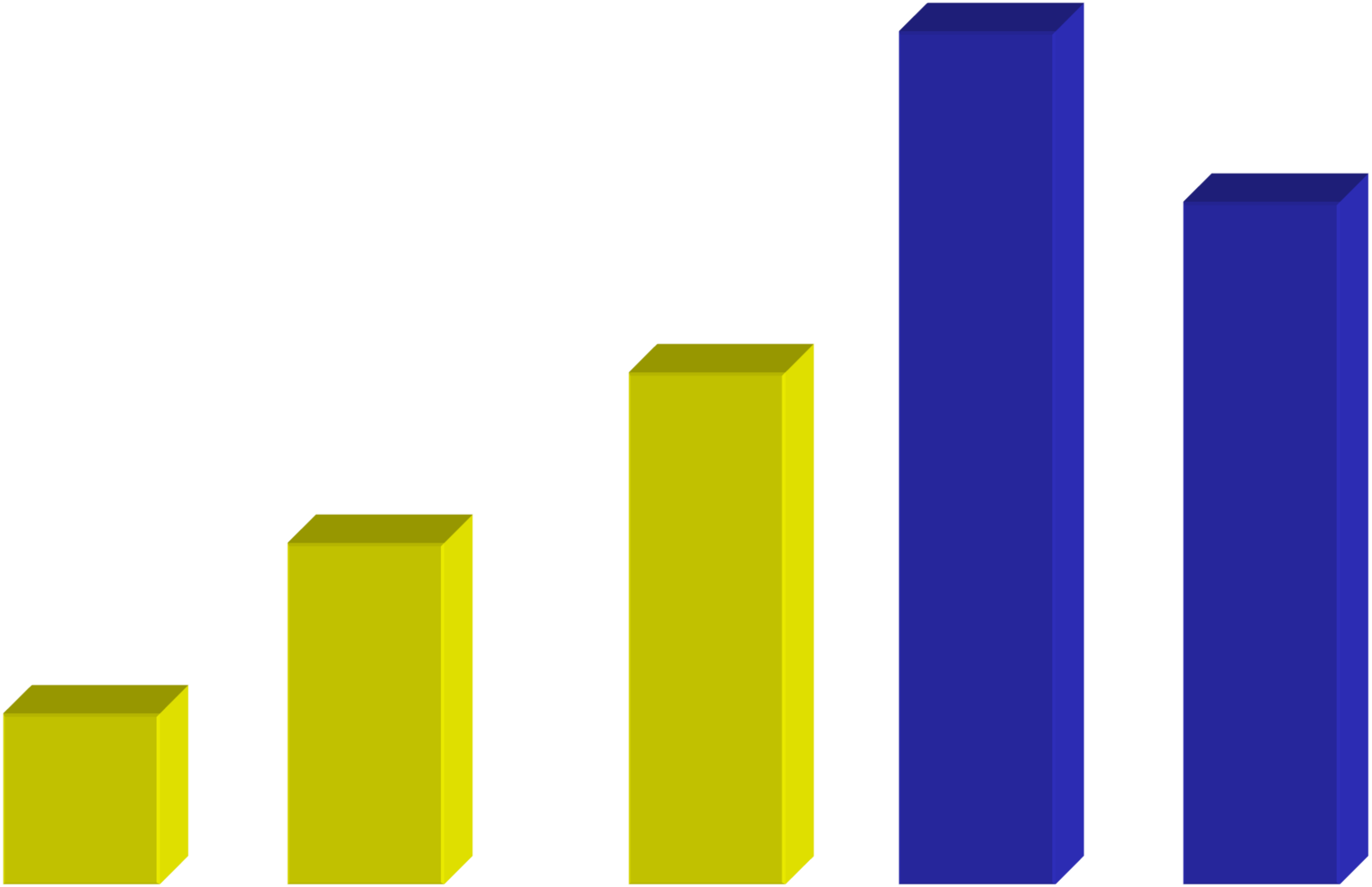
Selection Sort Animation



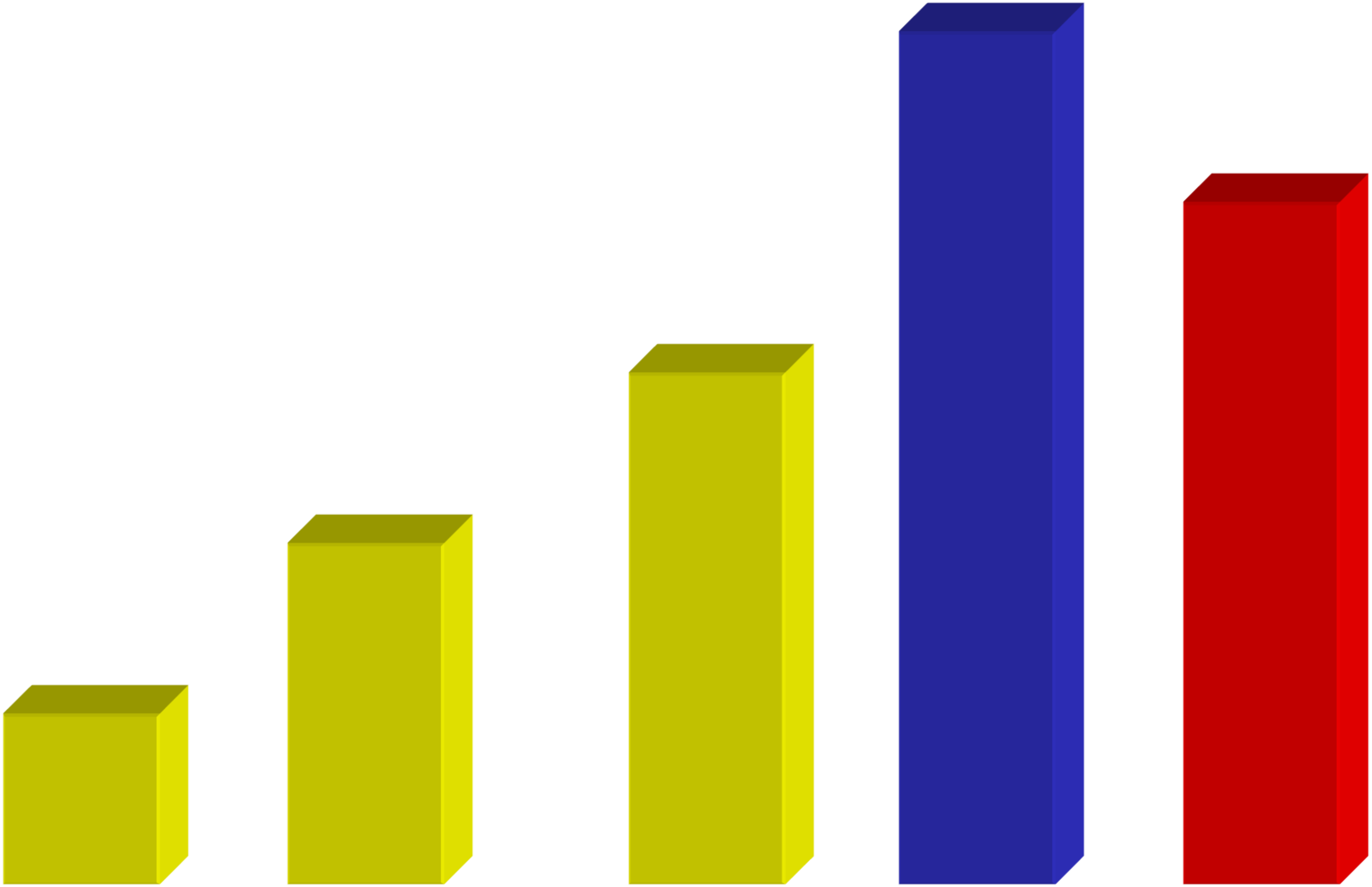
Selection Sort Animation



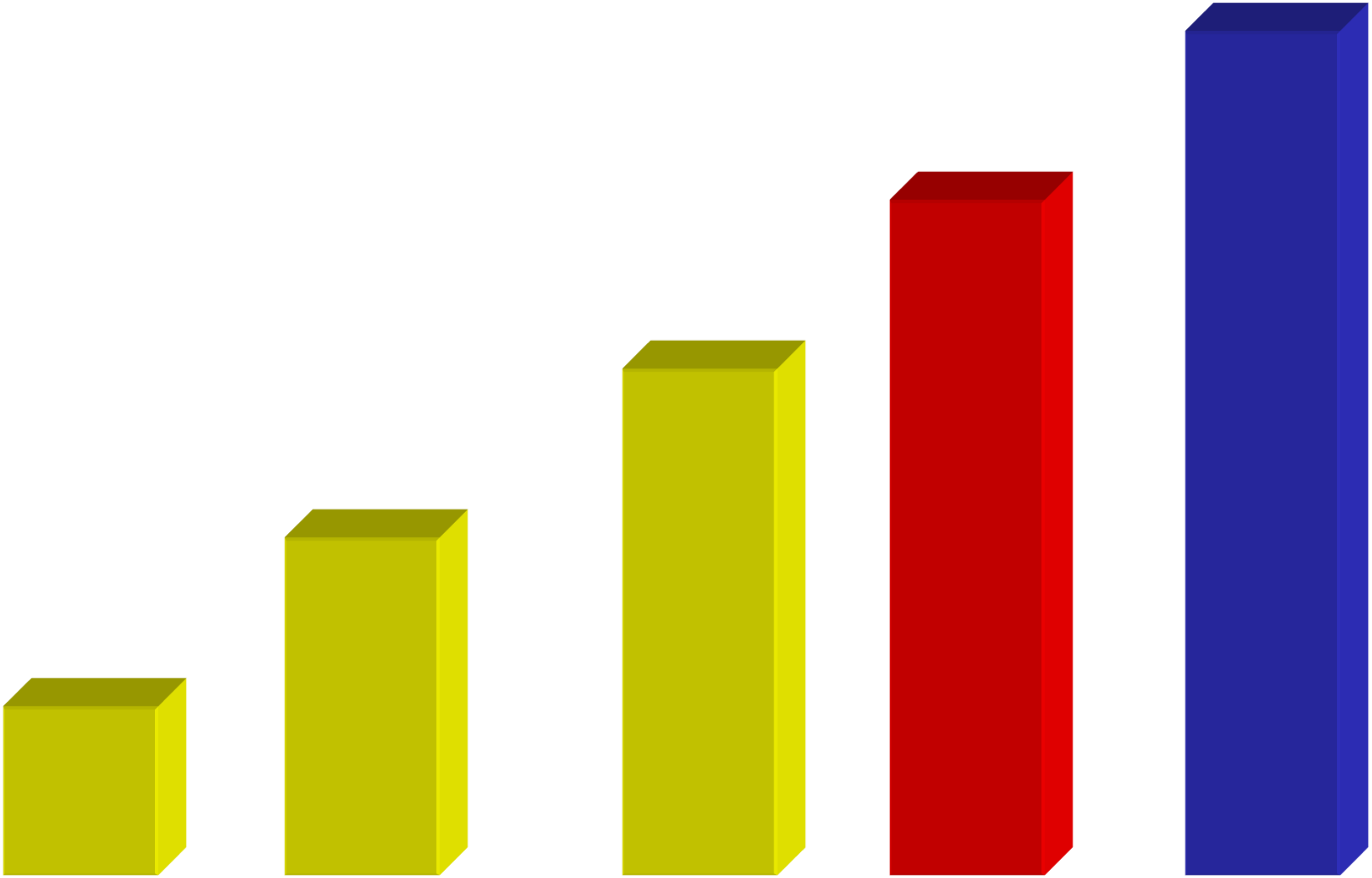
Selection Sort Animation



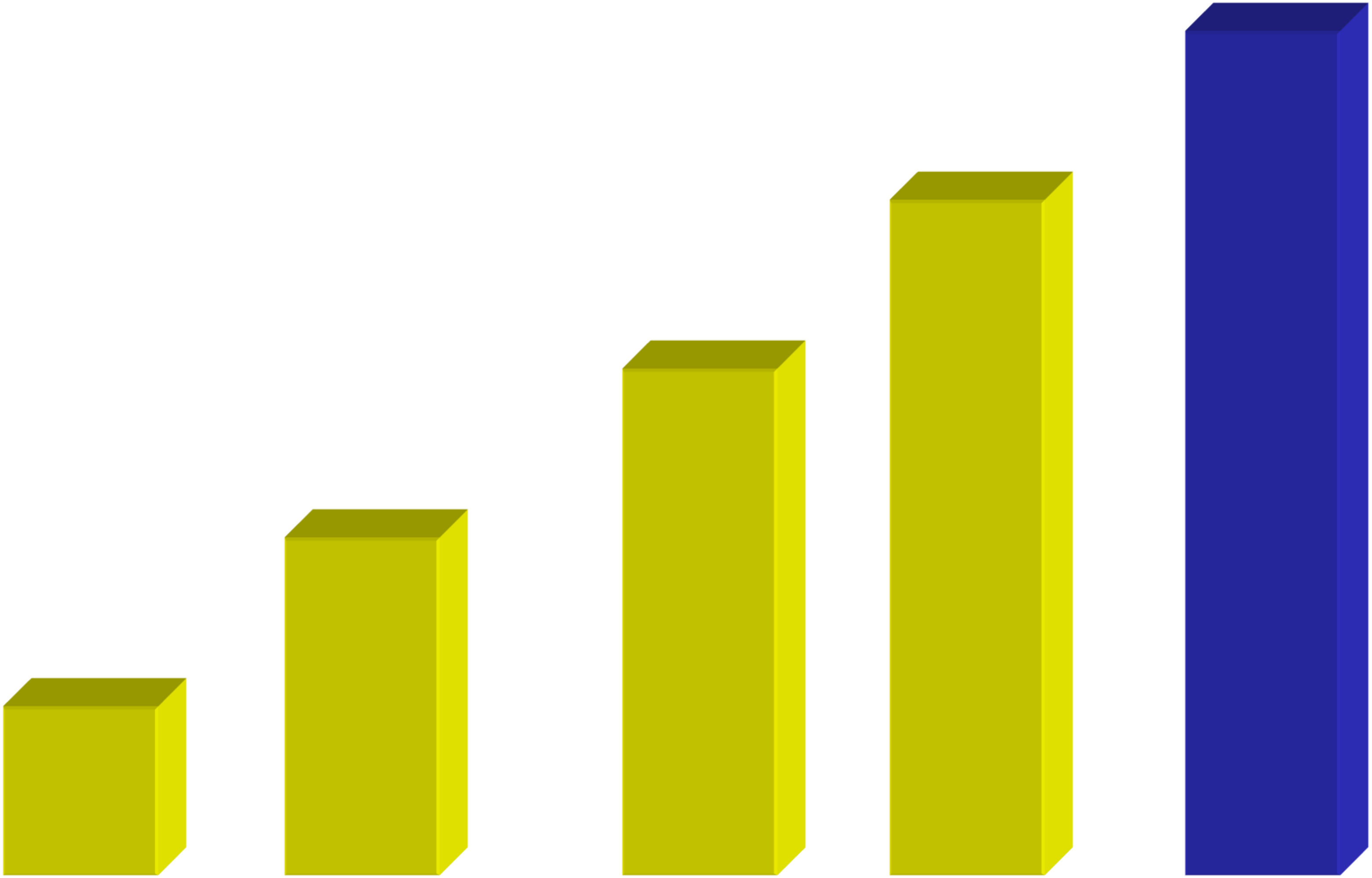
Selection Sort Animation



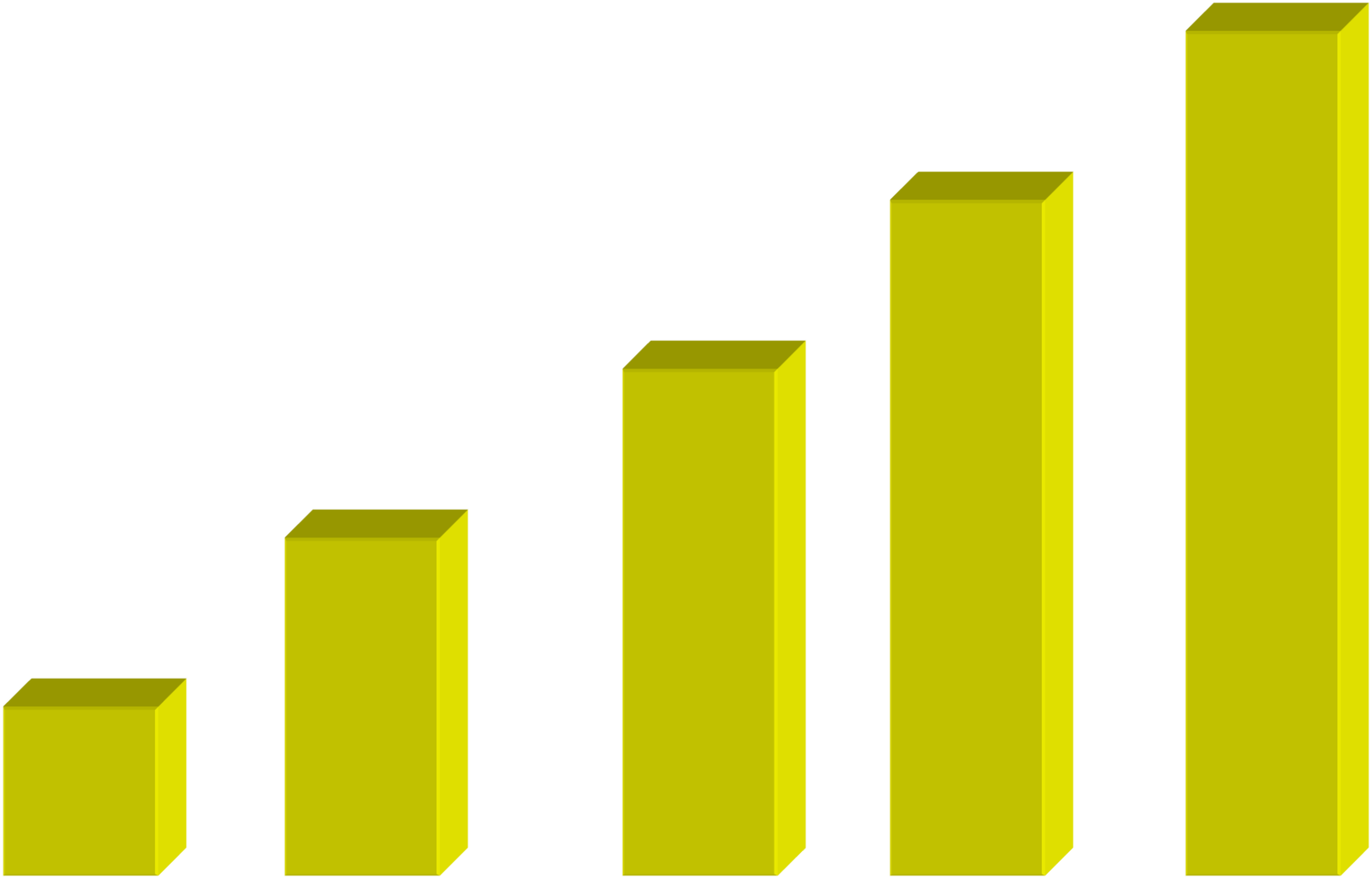
Selection Sort Animation



Selection Sort Animation



Selection Sort Animation



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

```
Algorithm selectionSort(A,n):
  Input: Array A of size n
  Output: Array A sorted

  for k ← 0 to n−2 do
     min ← k
     for j ← k+1 to n−1 do
          if A[j] < A[min] then
               min ← j
          end
     end
     swap(A[k], A[min])
  end
end
```

$$\sum_{k=0}^{n-2} \sum_{j=k+1}^{n-1} 1$$

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

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

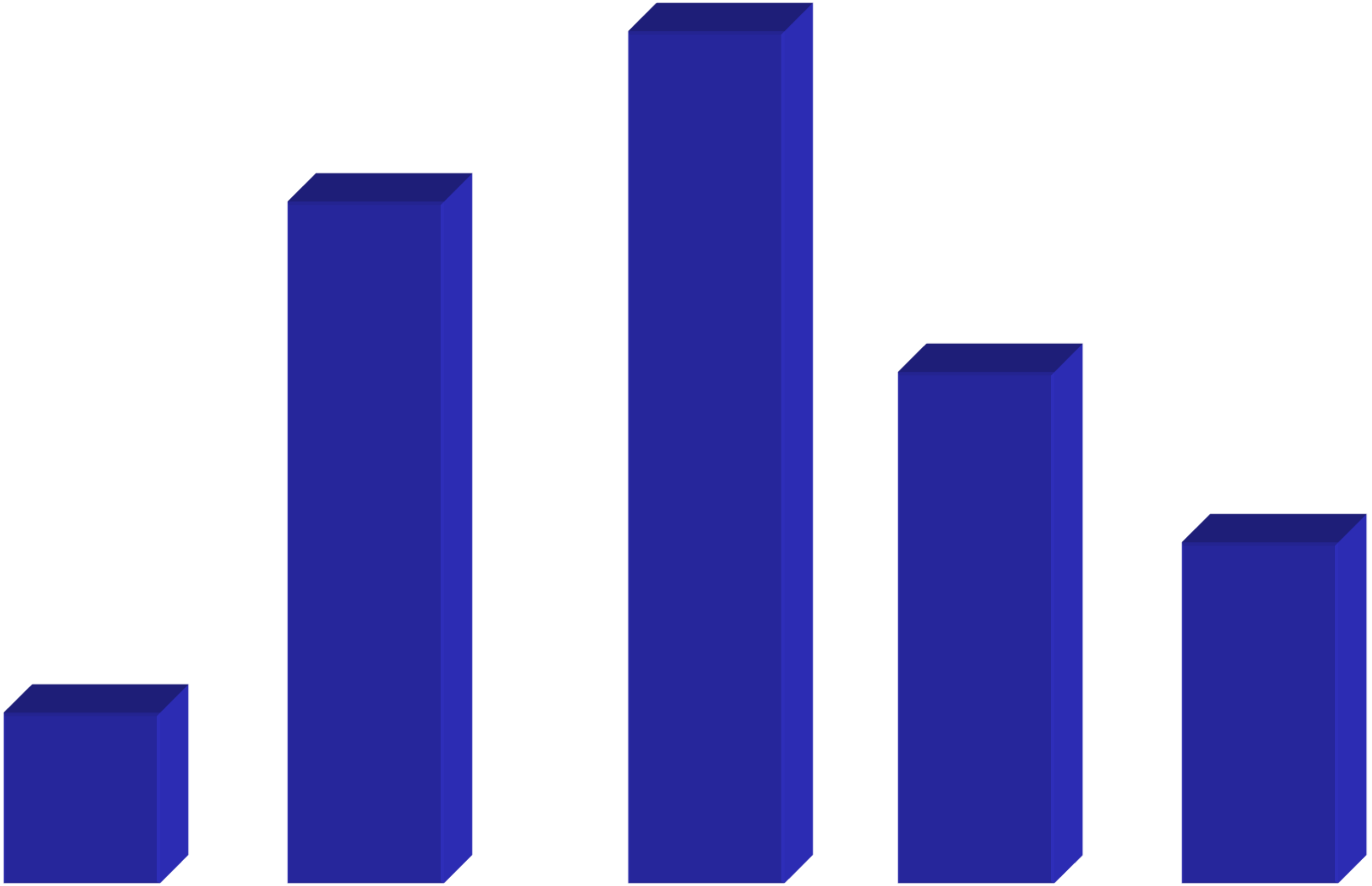
$$= \sum_{k=0}^{n-2}\left(n-k-1\right)$$

$$= n\sum_{k=0}^{n-2}1 - \sum_{k=0}^{n-2}k - \sum_{k=0}^{n-2}1 \qquad \in O(n^2)$$

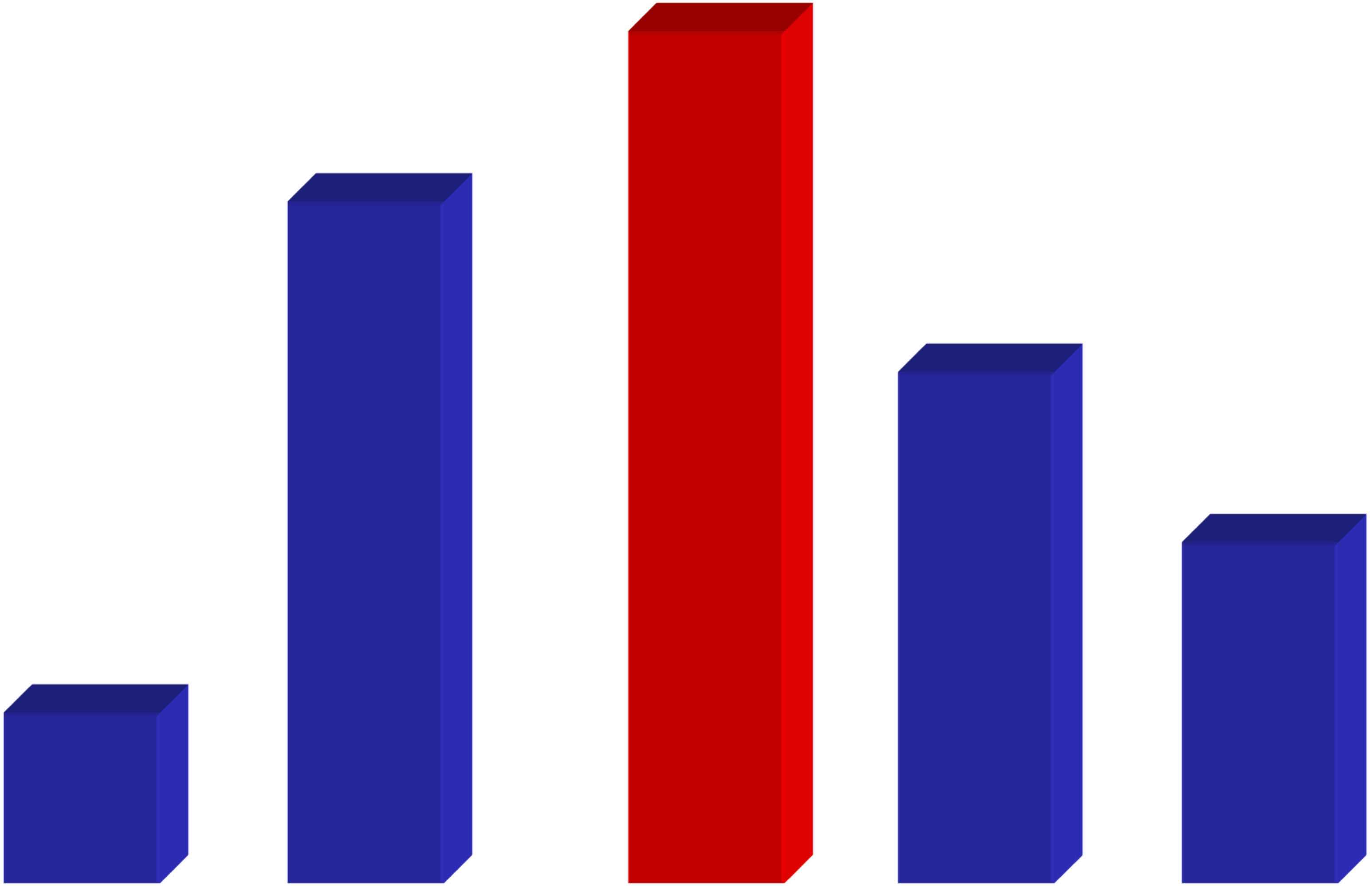
$$= n\big((n-2)+1\big) - \frac{(n-2)(n-1)}{2} - \big((n-2)+1\big)$$

$$= n^2 - n - \frac{n^2-3n+2}{2} - n + 1 = \frac{n^2}{2} - \frac{n}{2}$$

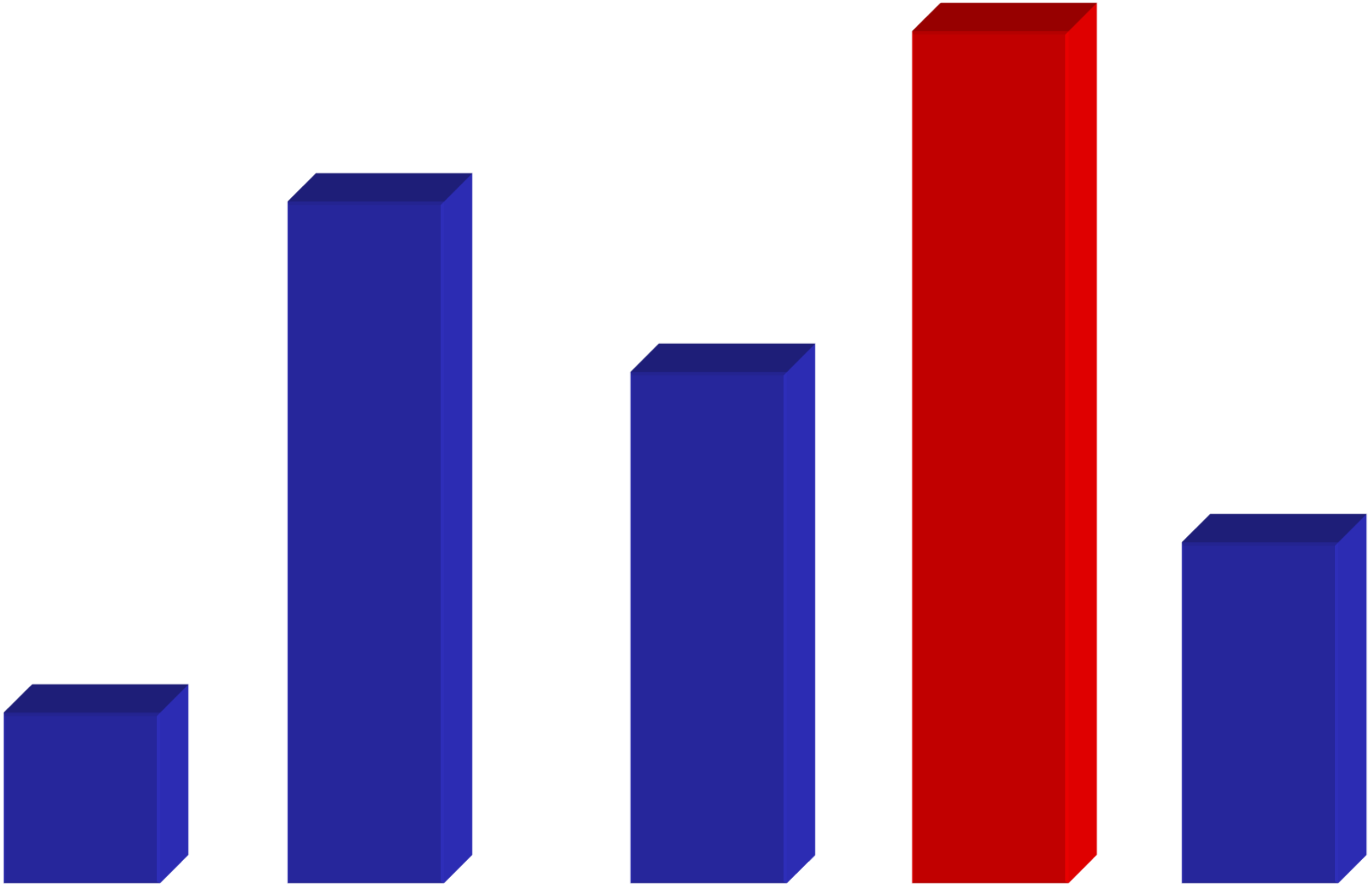
Bubble Sort



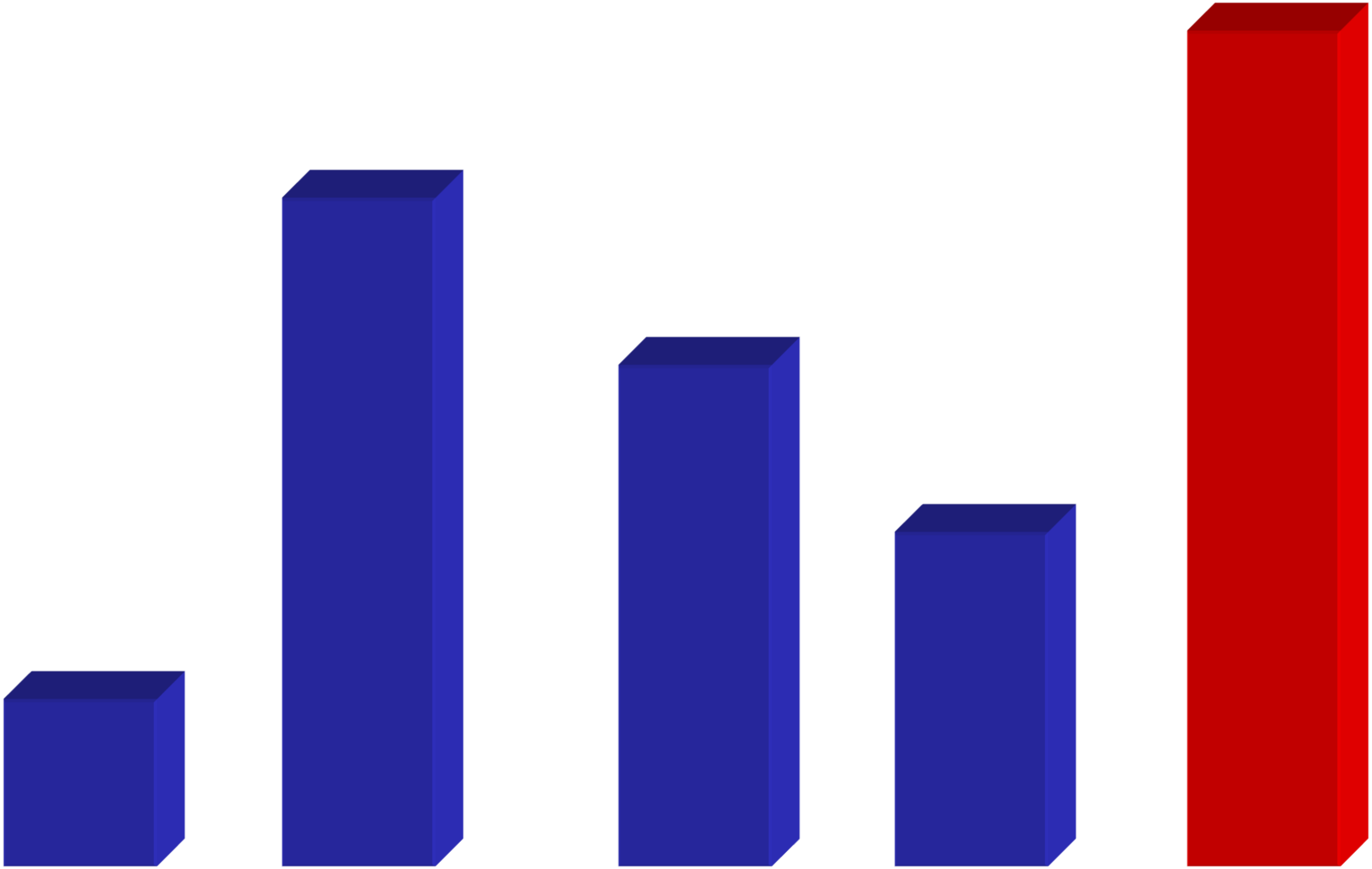
Bubble Sort



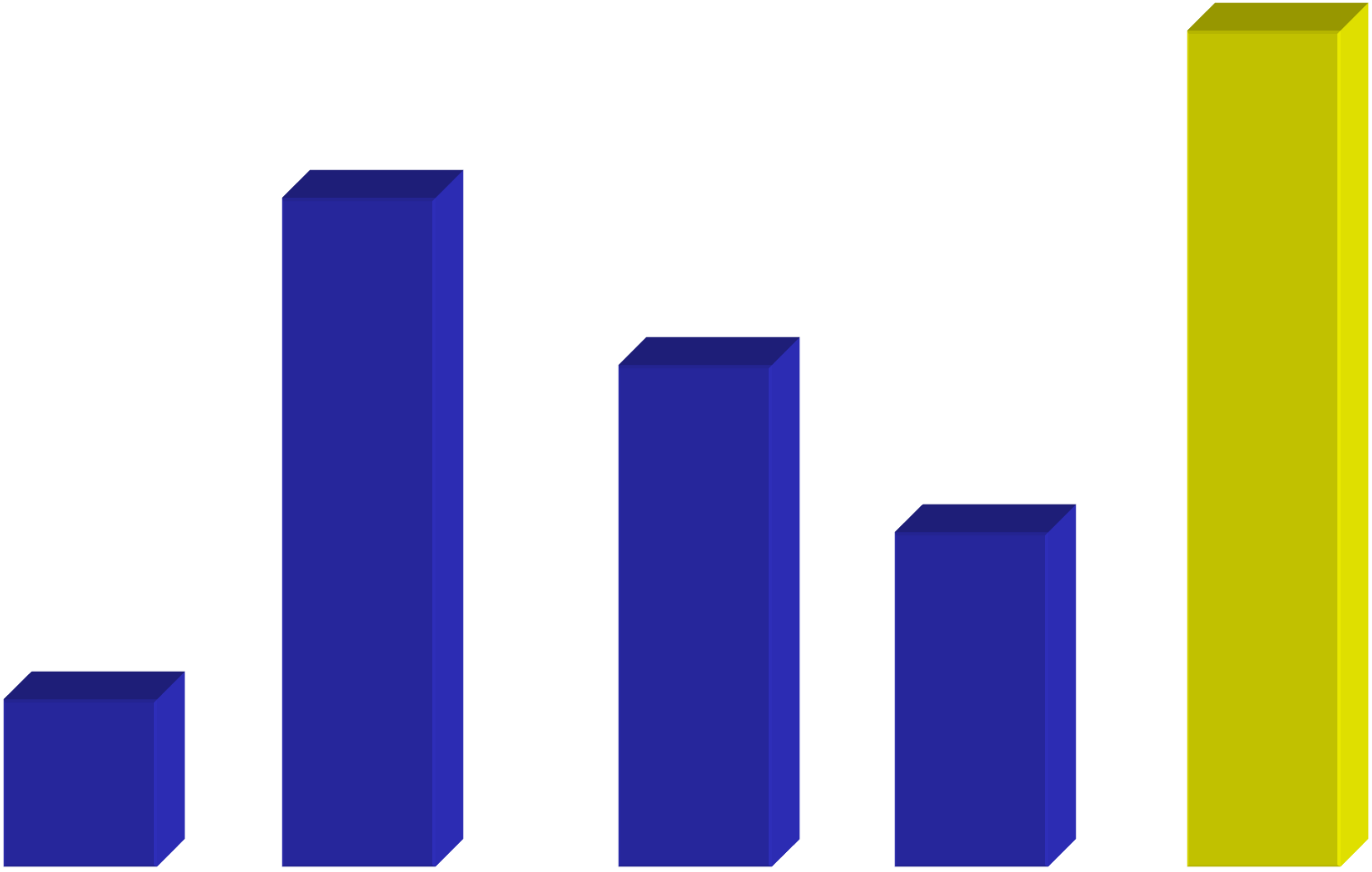
Bubble Sort



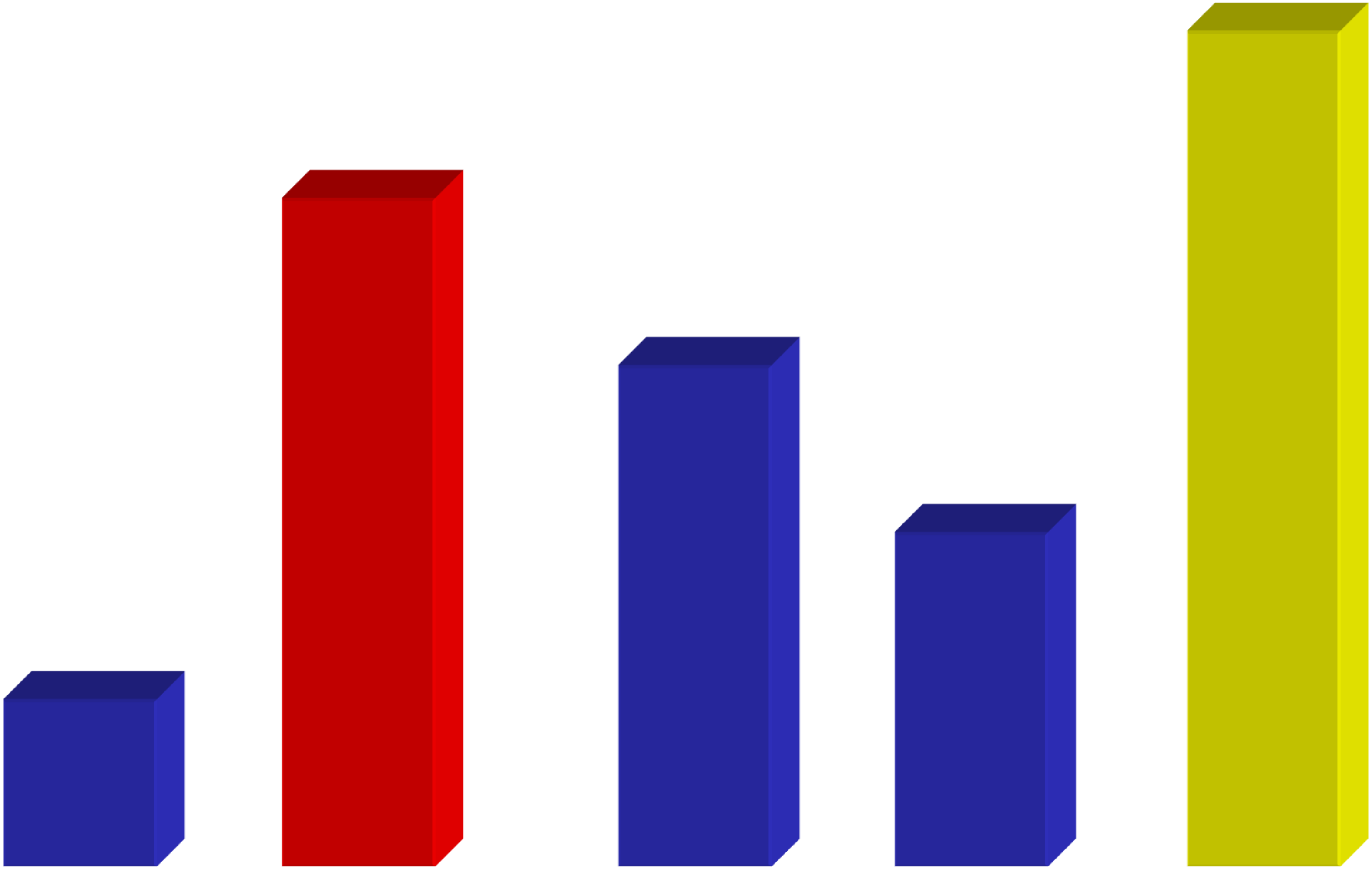
Bubble Sort



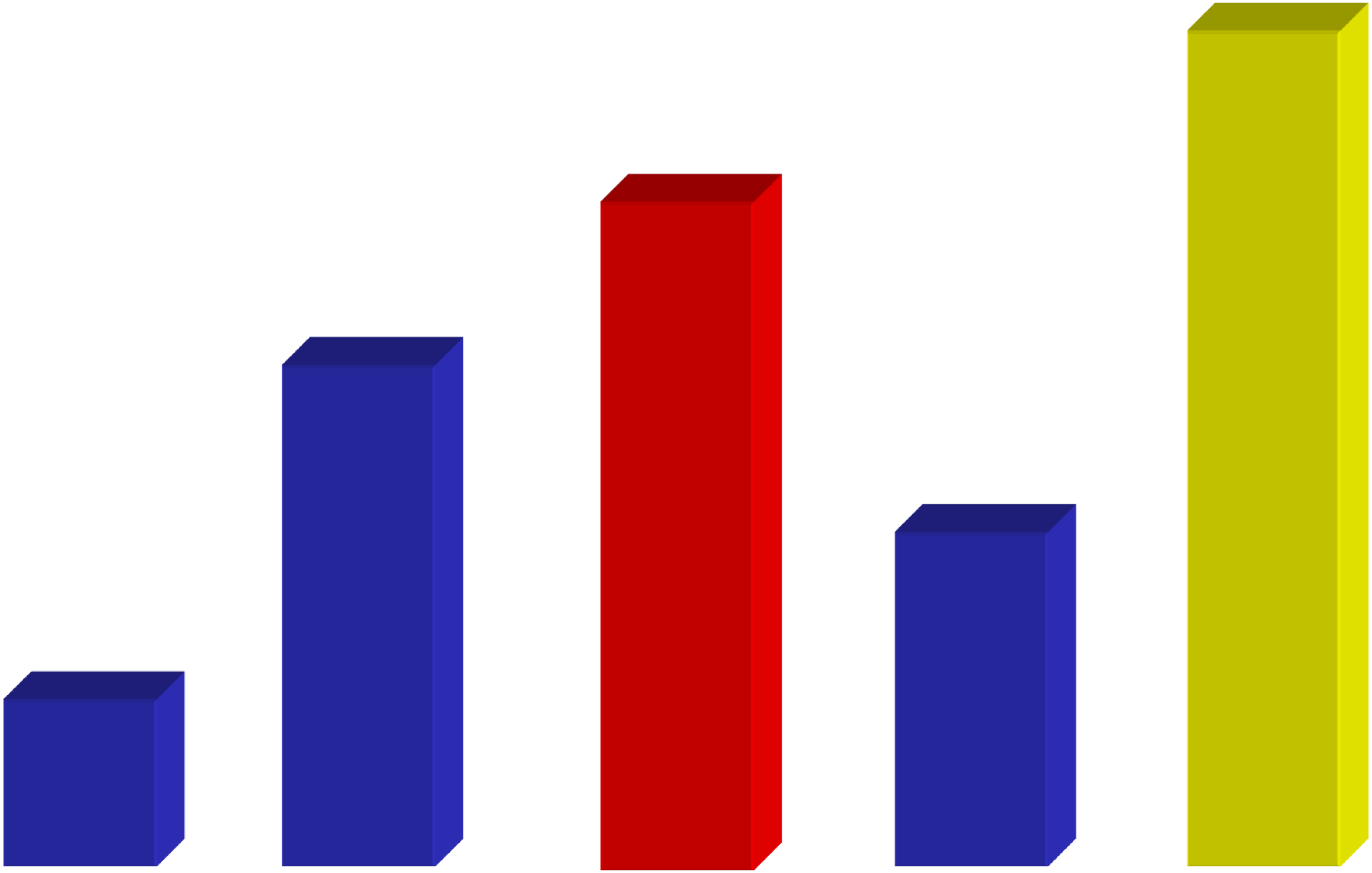
Bubble Sort



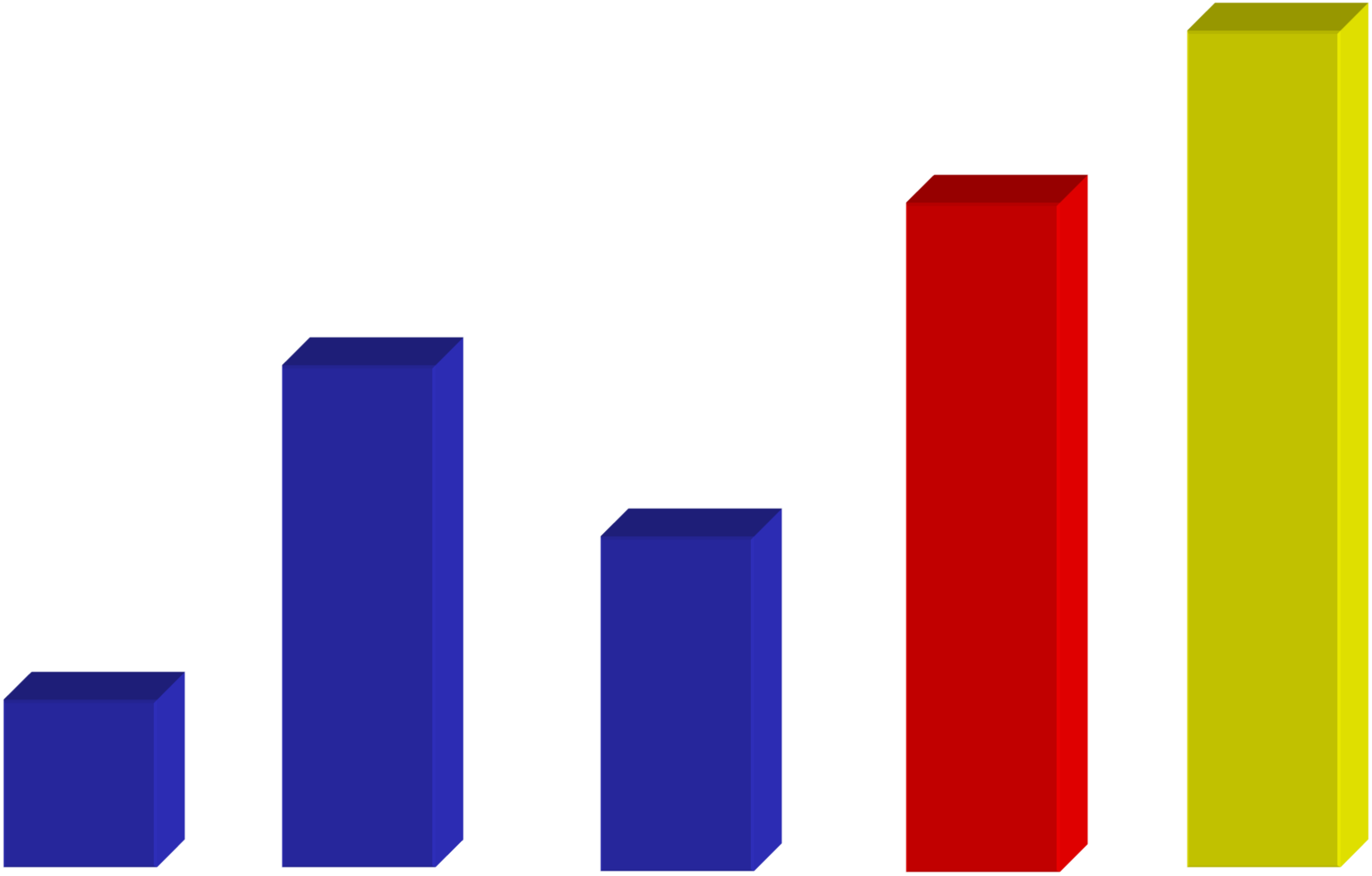
Bubble Sort



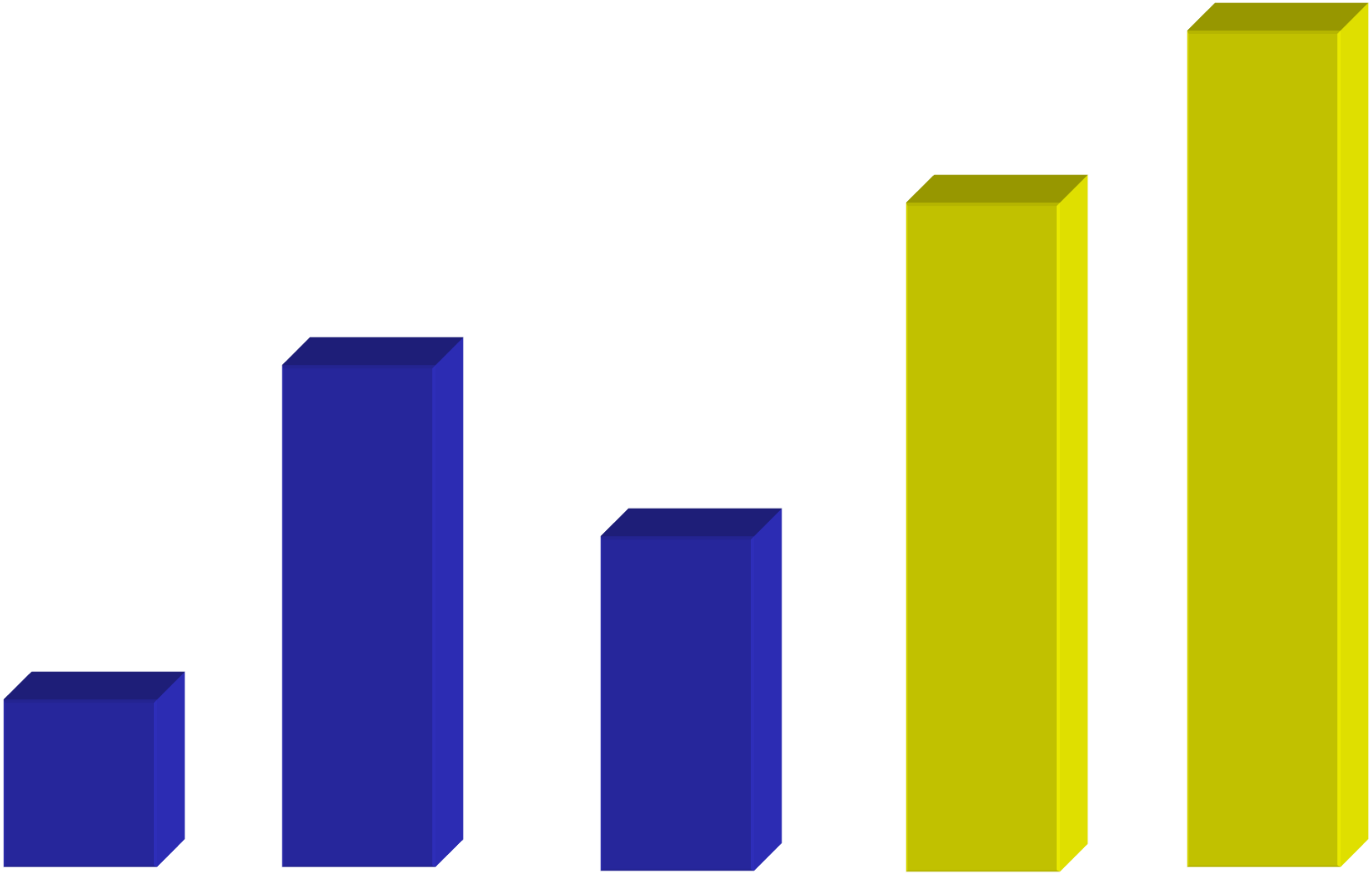
Bubble Sort



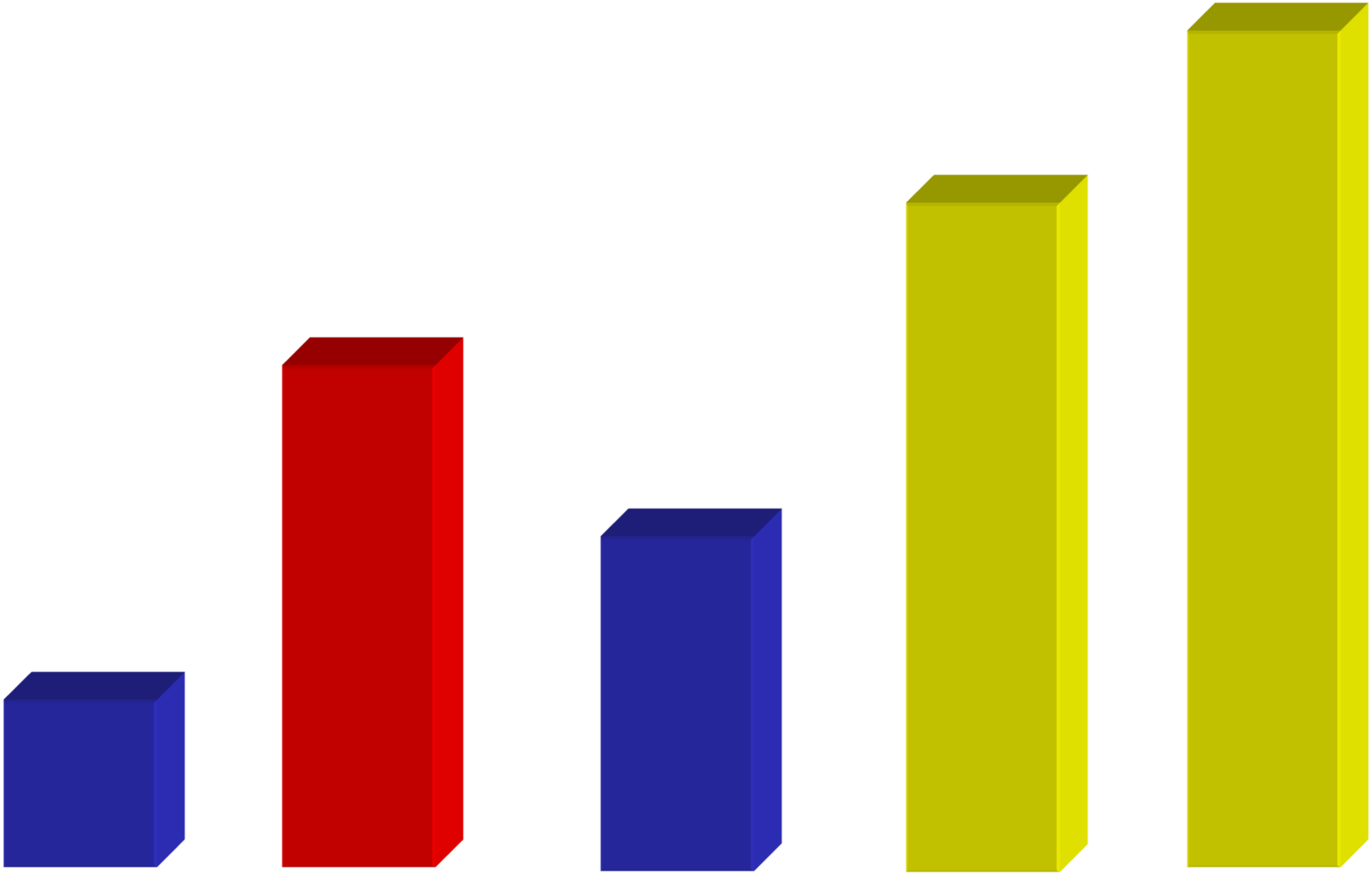
Bubble Sort



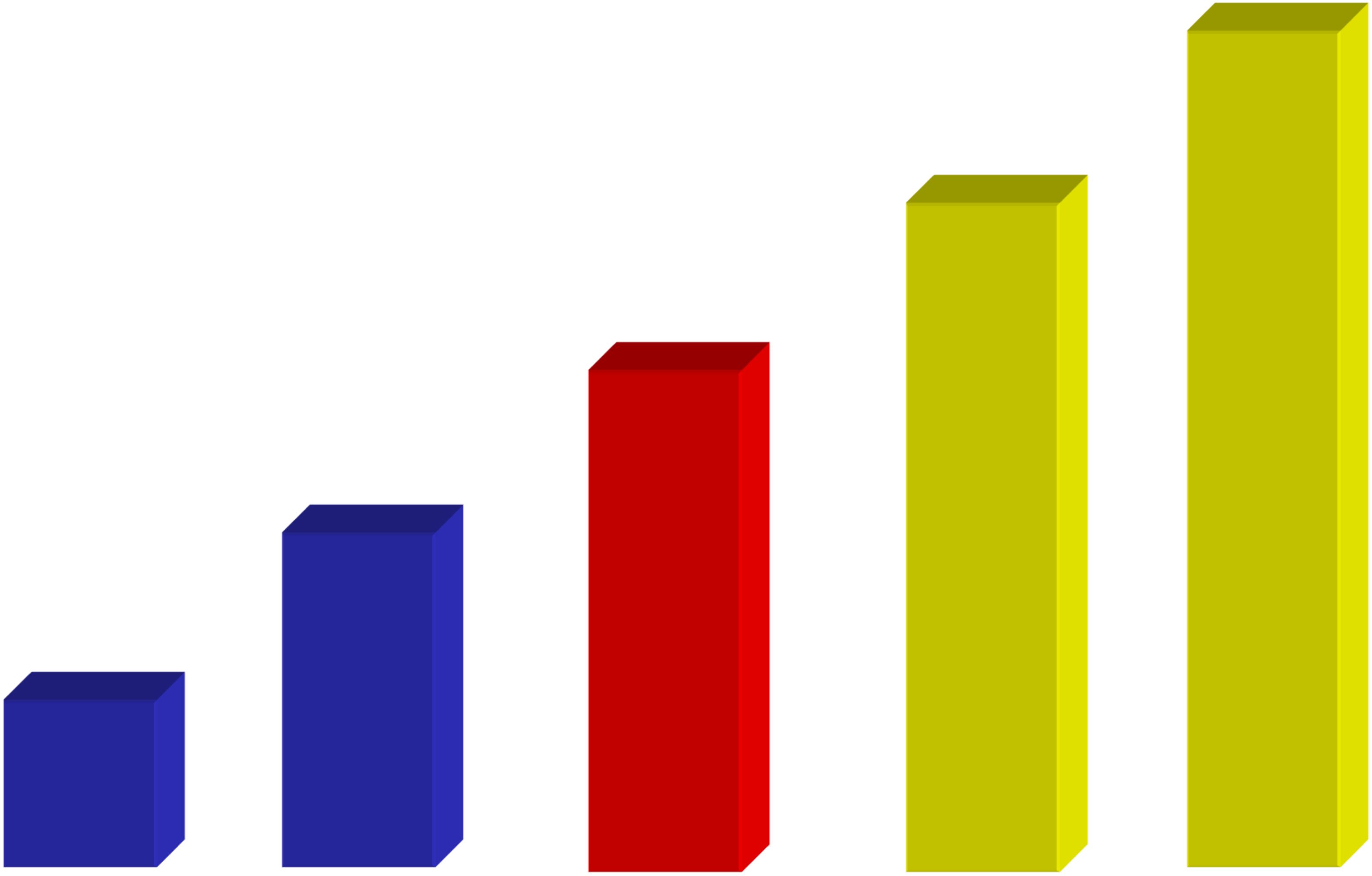
Bubble Sort



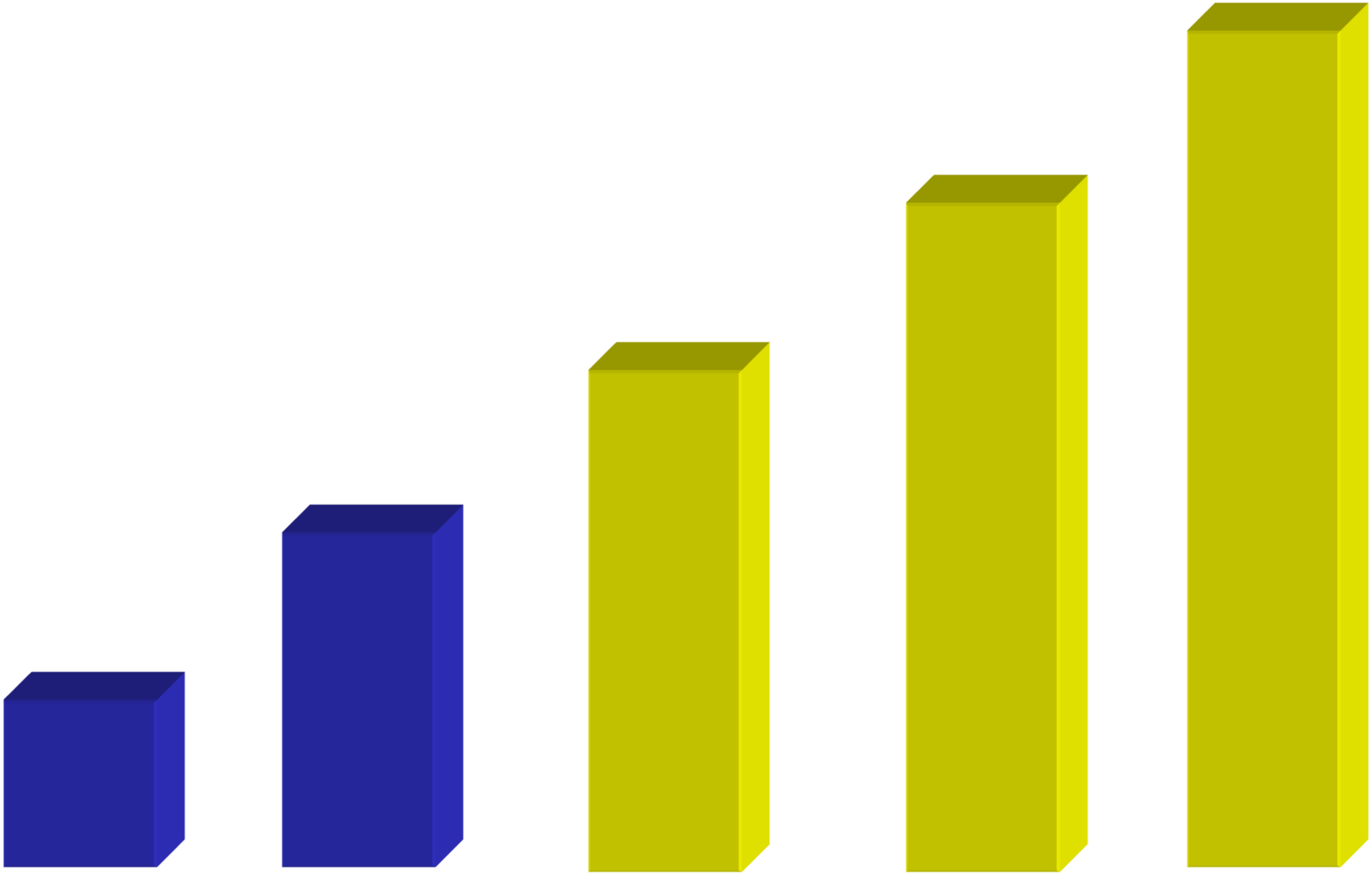
Bubble Sort



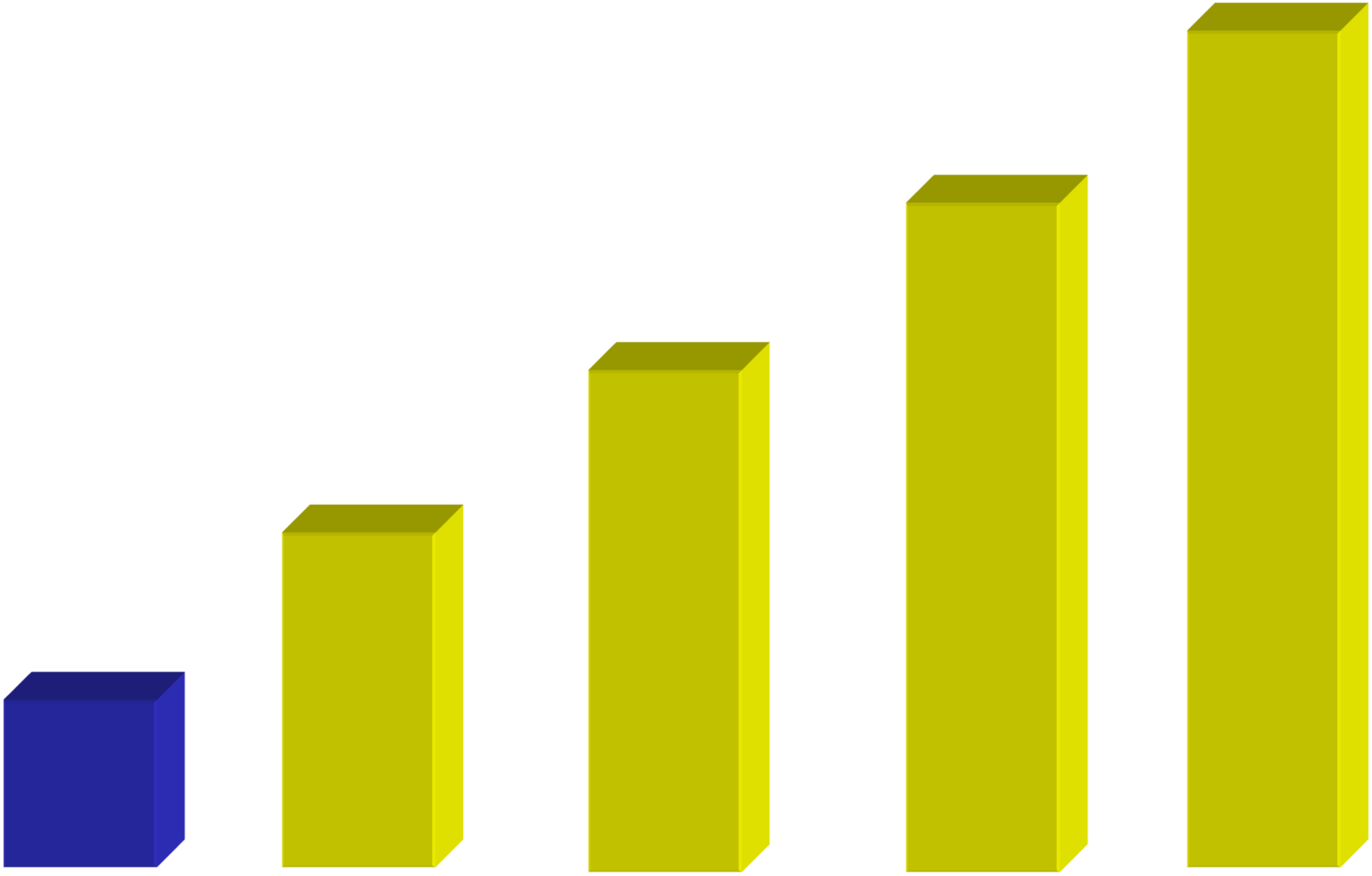
Bubble Sort



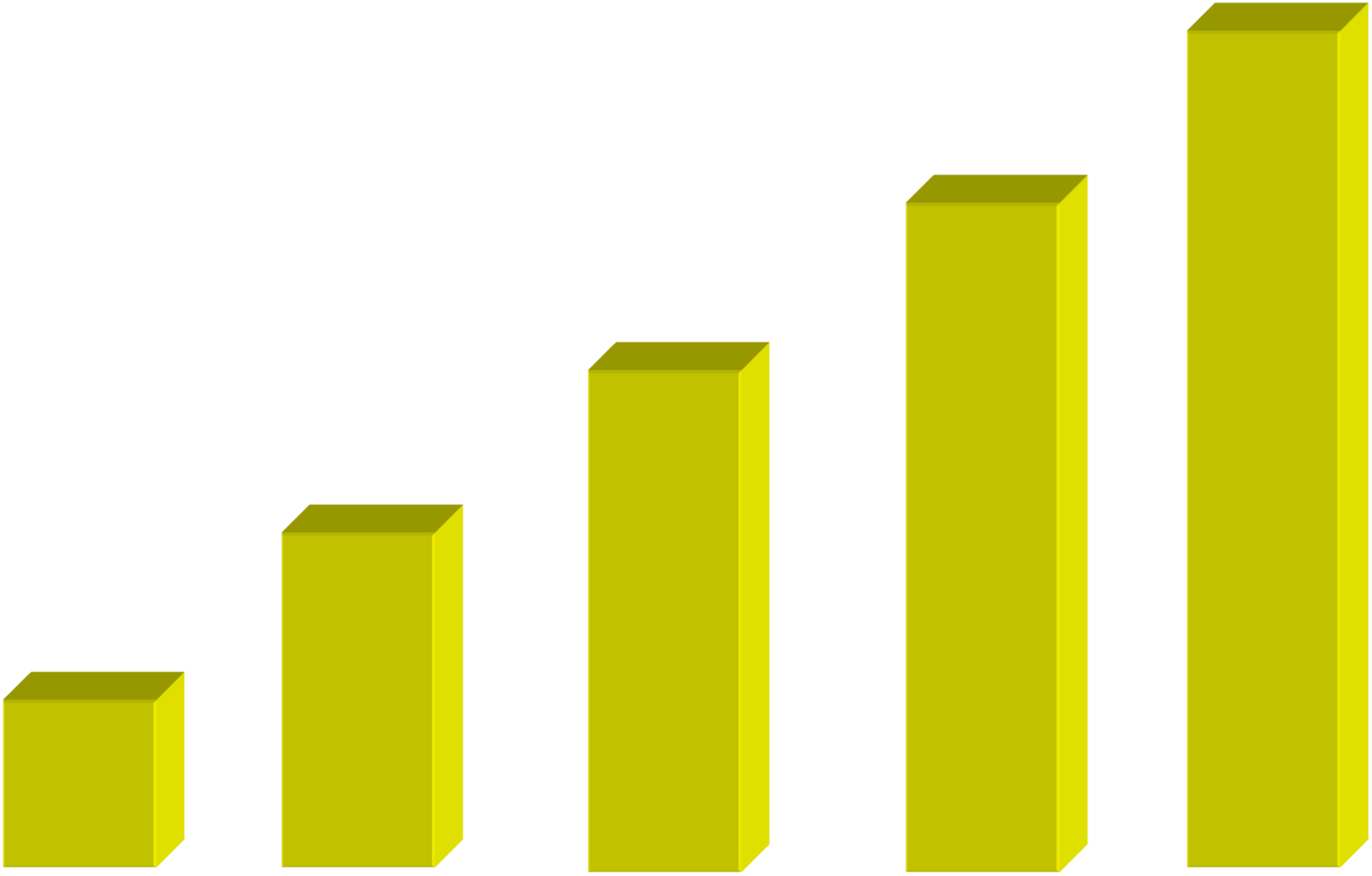
Bubble Sort



Bubble Sort



Bubble Sort



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

Algorithm bubbleSort(A,n):
  Input: Array A of size n
  Output: Array A sorted

  for $k \leftarrow 0$ **to** $n-2$ **do**
    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
end

$$\sum_{k=0}^{n-2} \sum_{j=0}^{n-2-k} 1$$

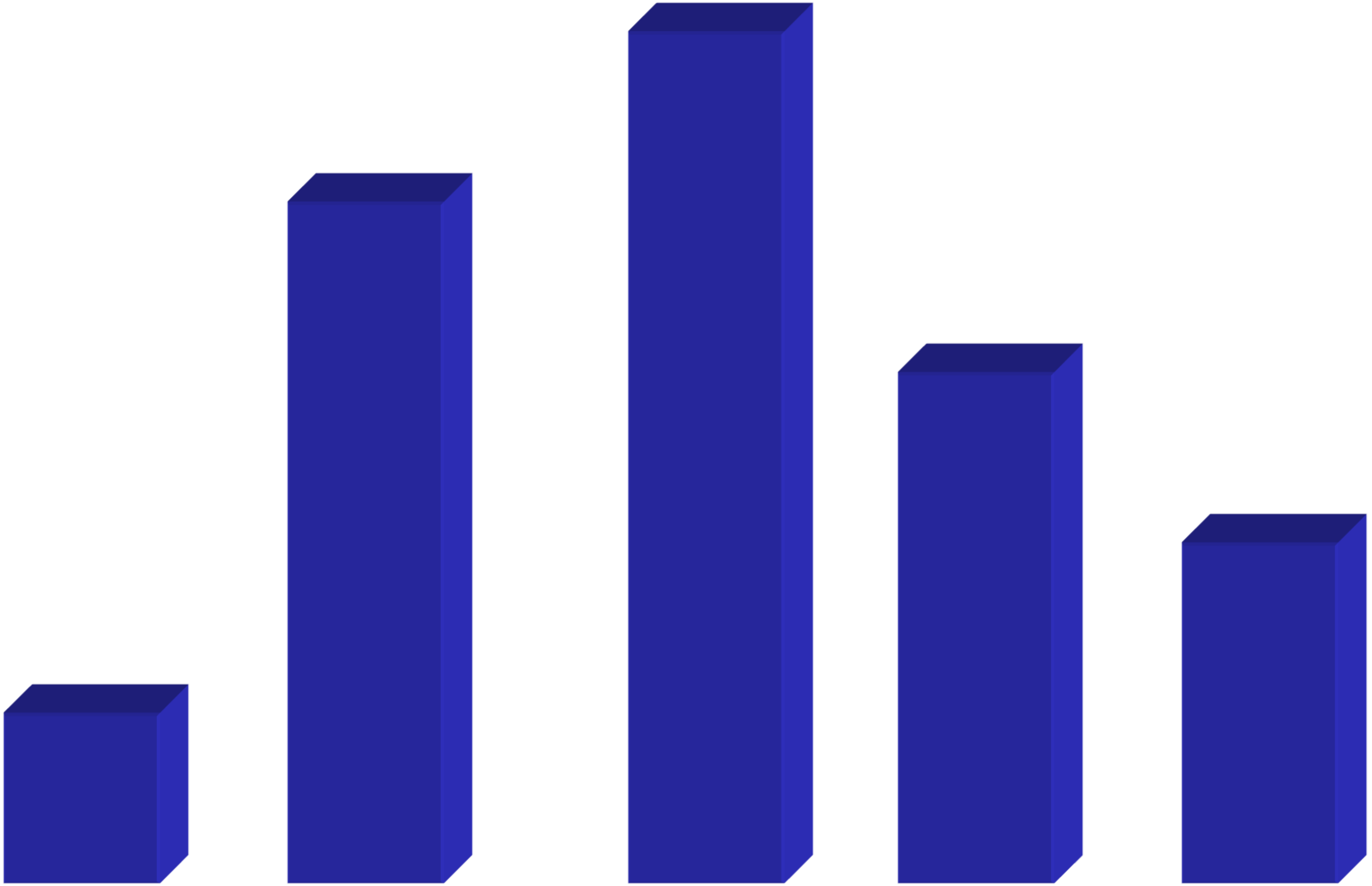
B.C. $\in O(n)$

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

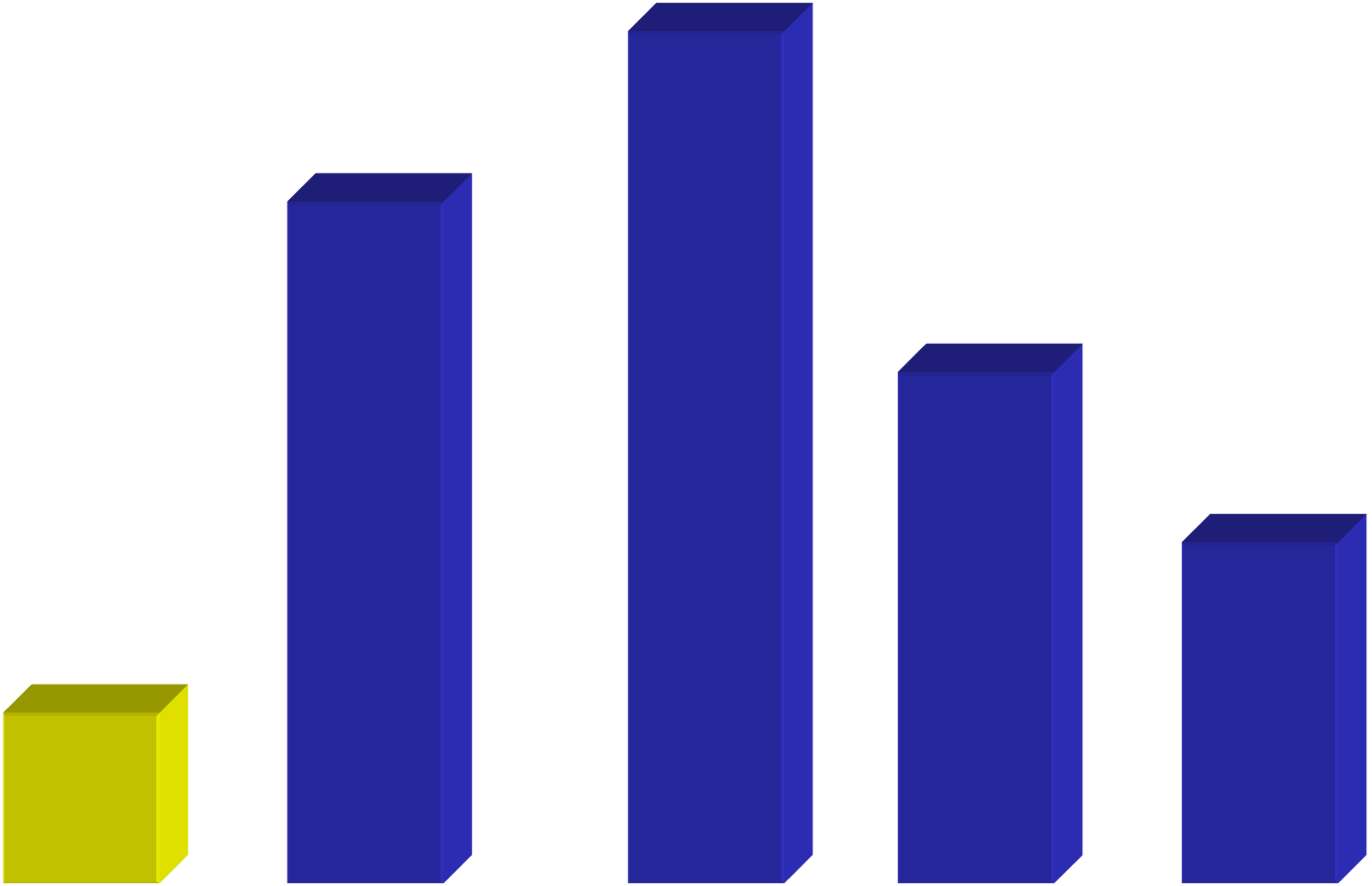
$$\sum_{k=0}^{n-2} \sum_{j=0}^{n-2-k} 1 = \sum_{k=0}^{n-2} \left((n-2-k) + 1 \right)$$

$$= \sum_{k=0}^{n-2} n - 1 - k = \frac{n^2 - n}{2} \in O(n^2)$$

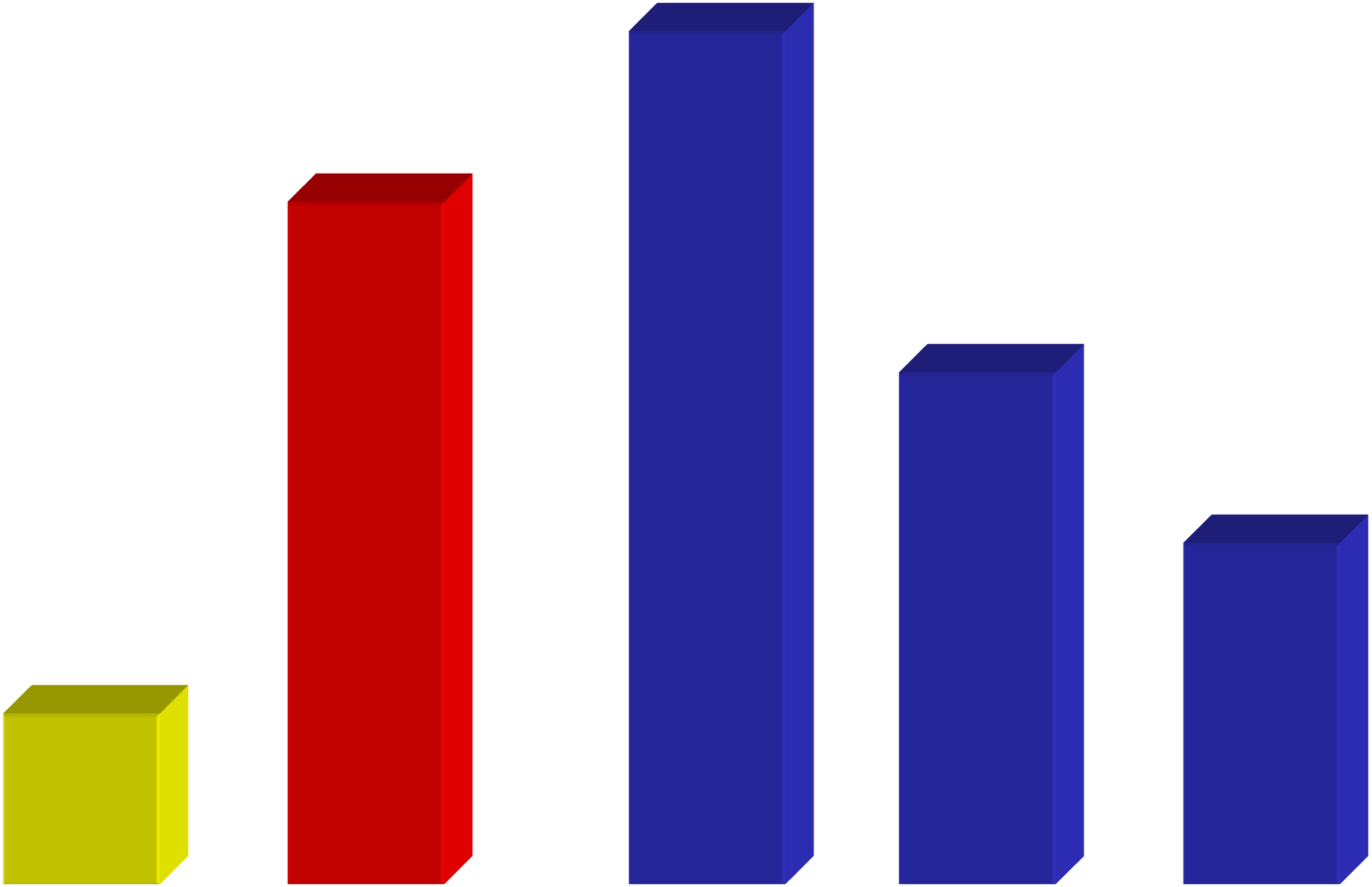
Insertion Sort



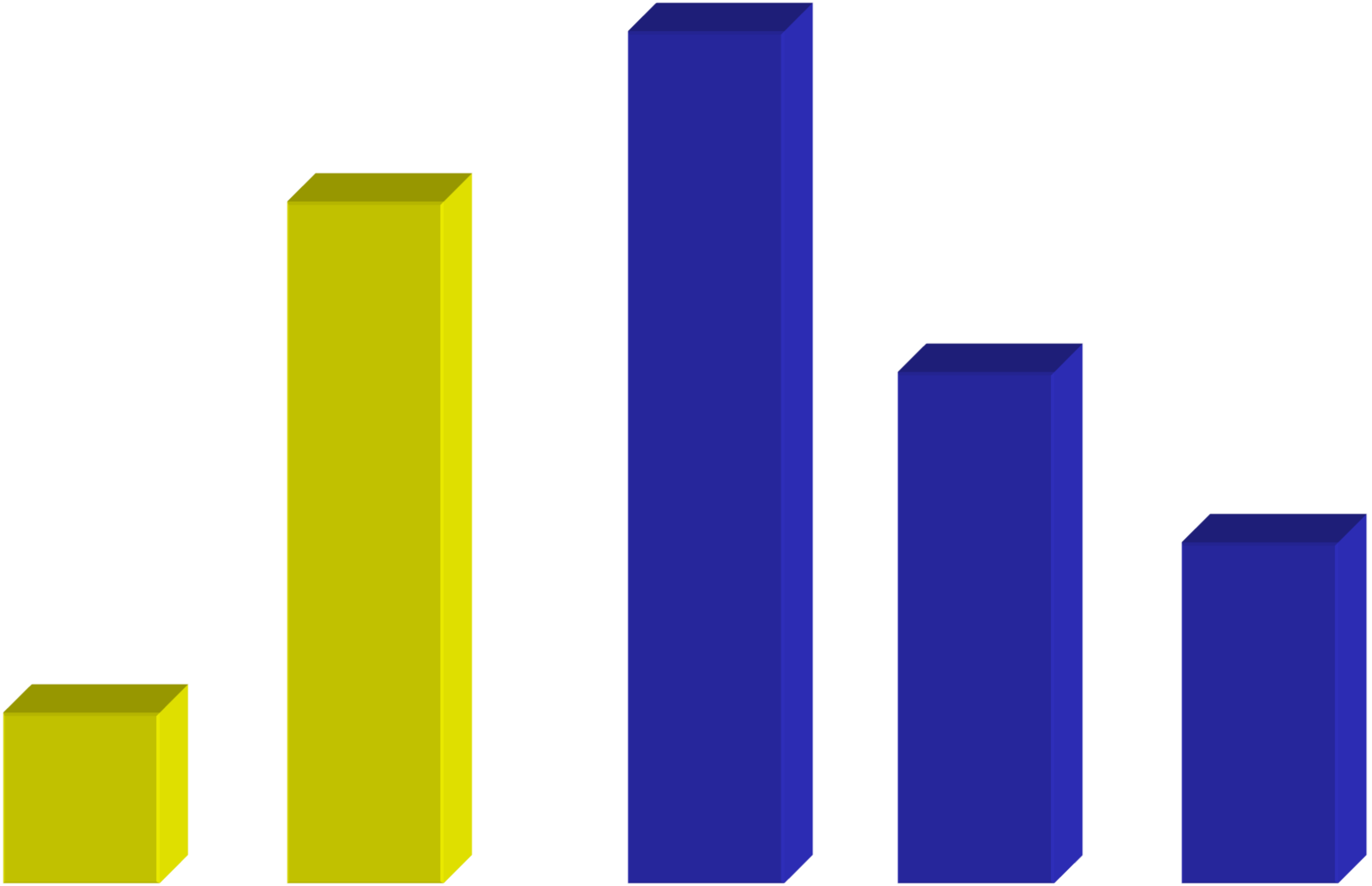
Insertion Sort



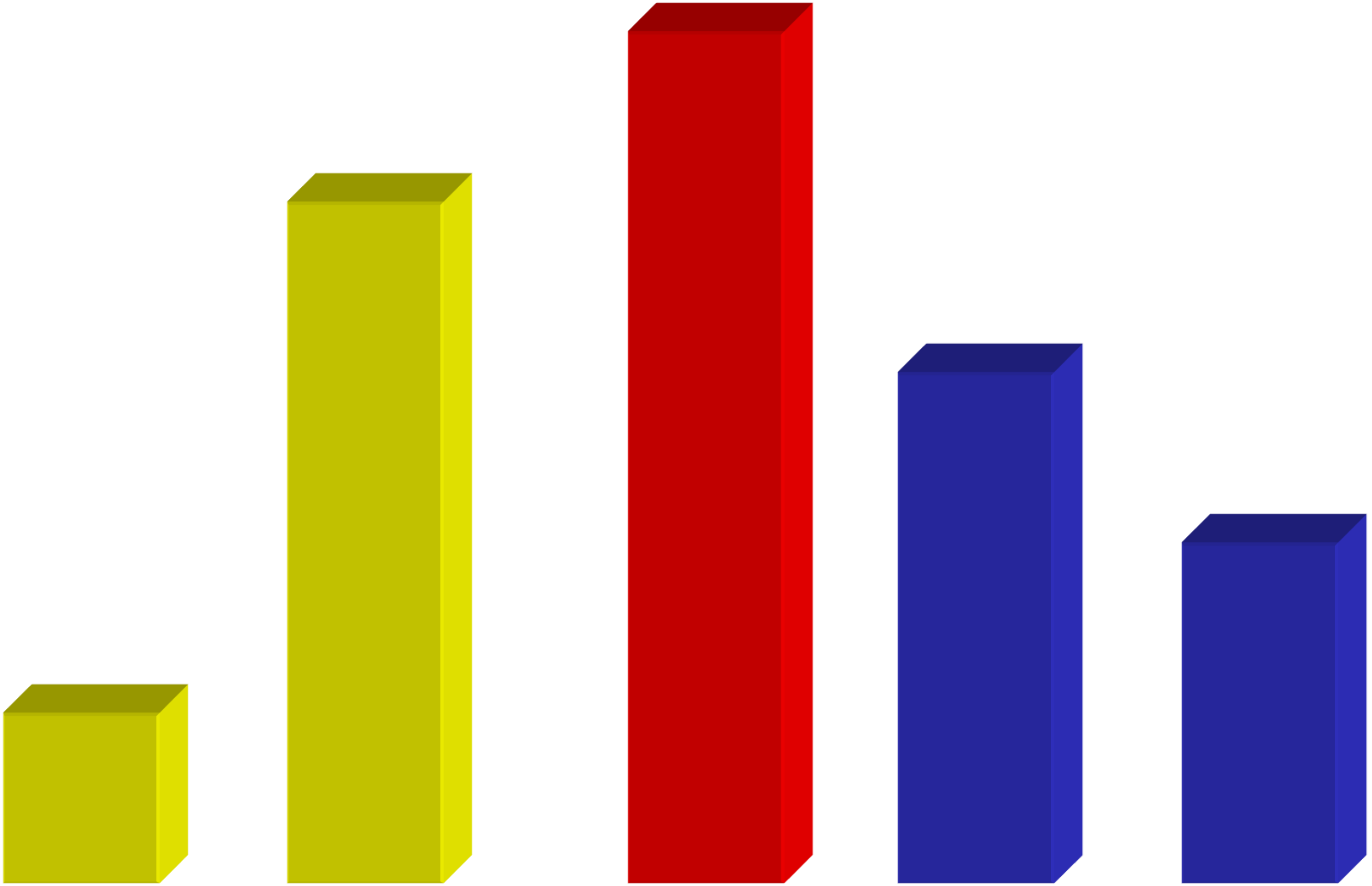
Insertion Sort



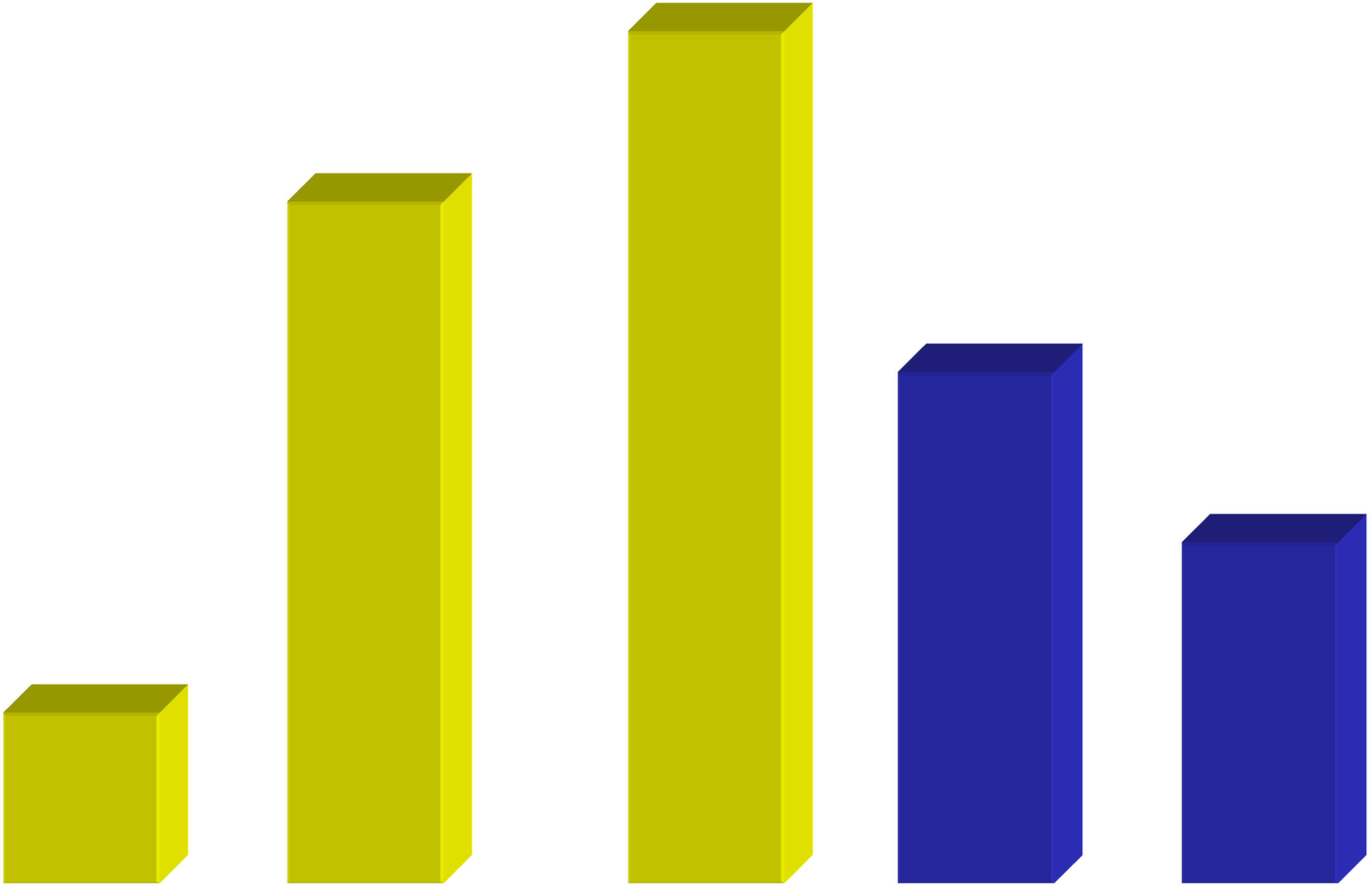
Insertion Sort



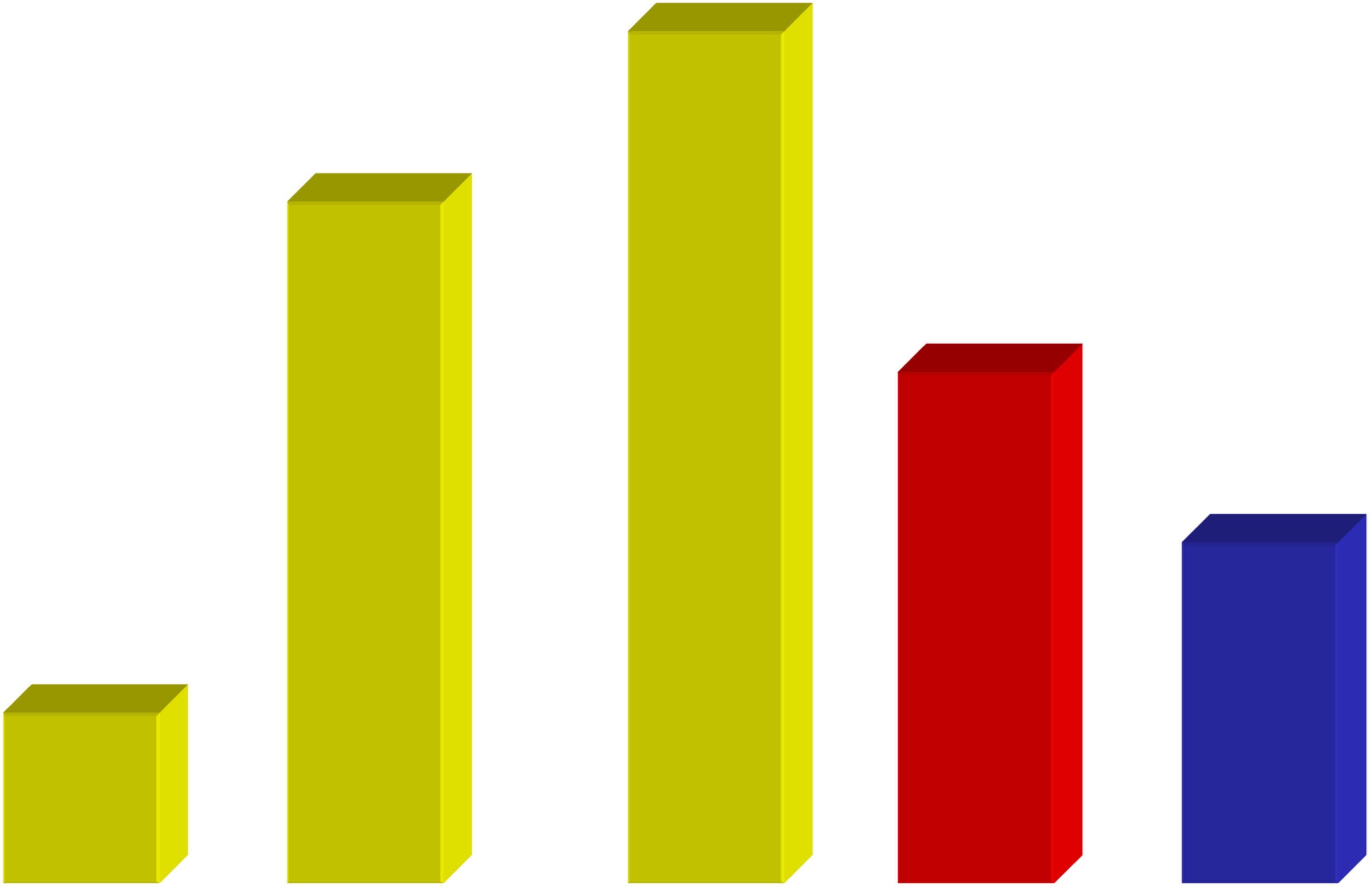
Insertion Sort



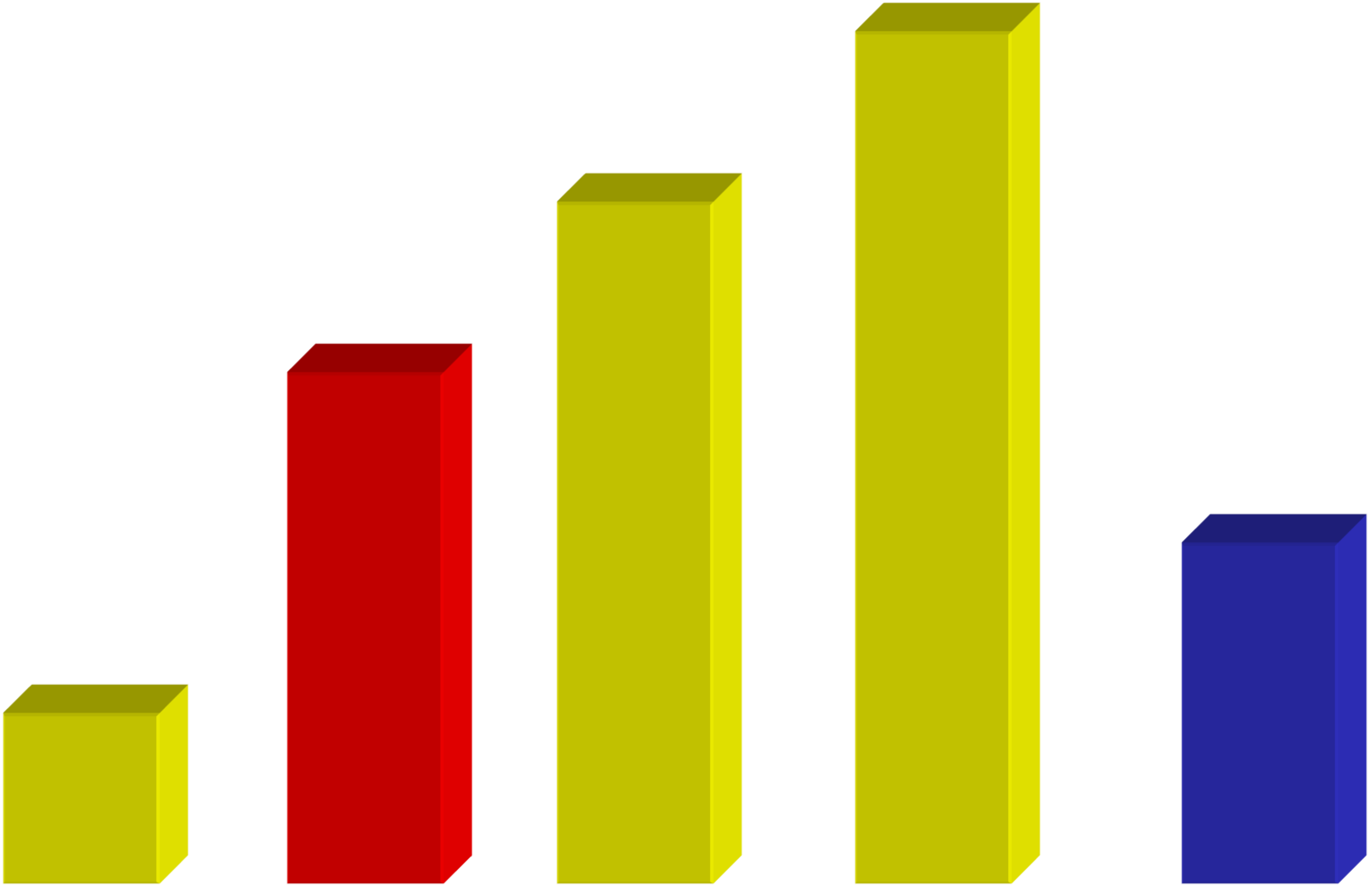
Insertion Sort



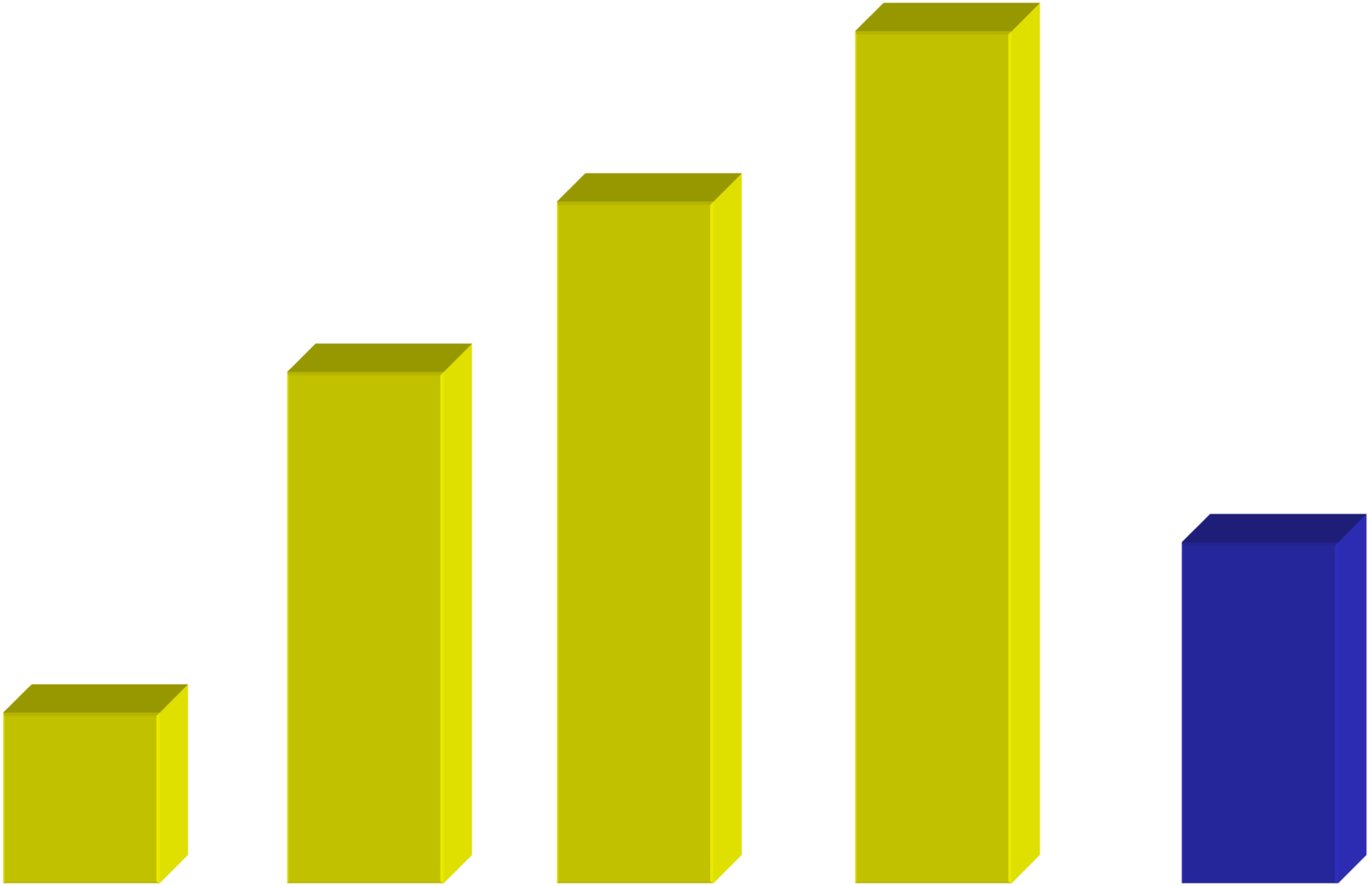
Insertion Sort



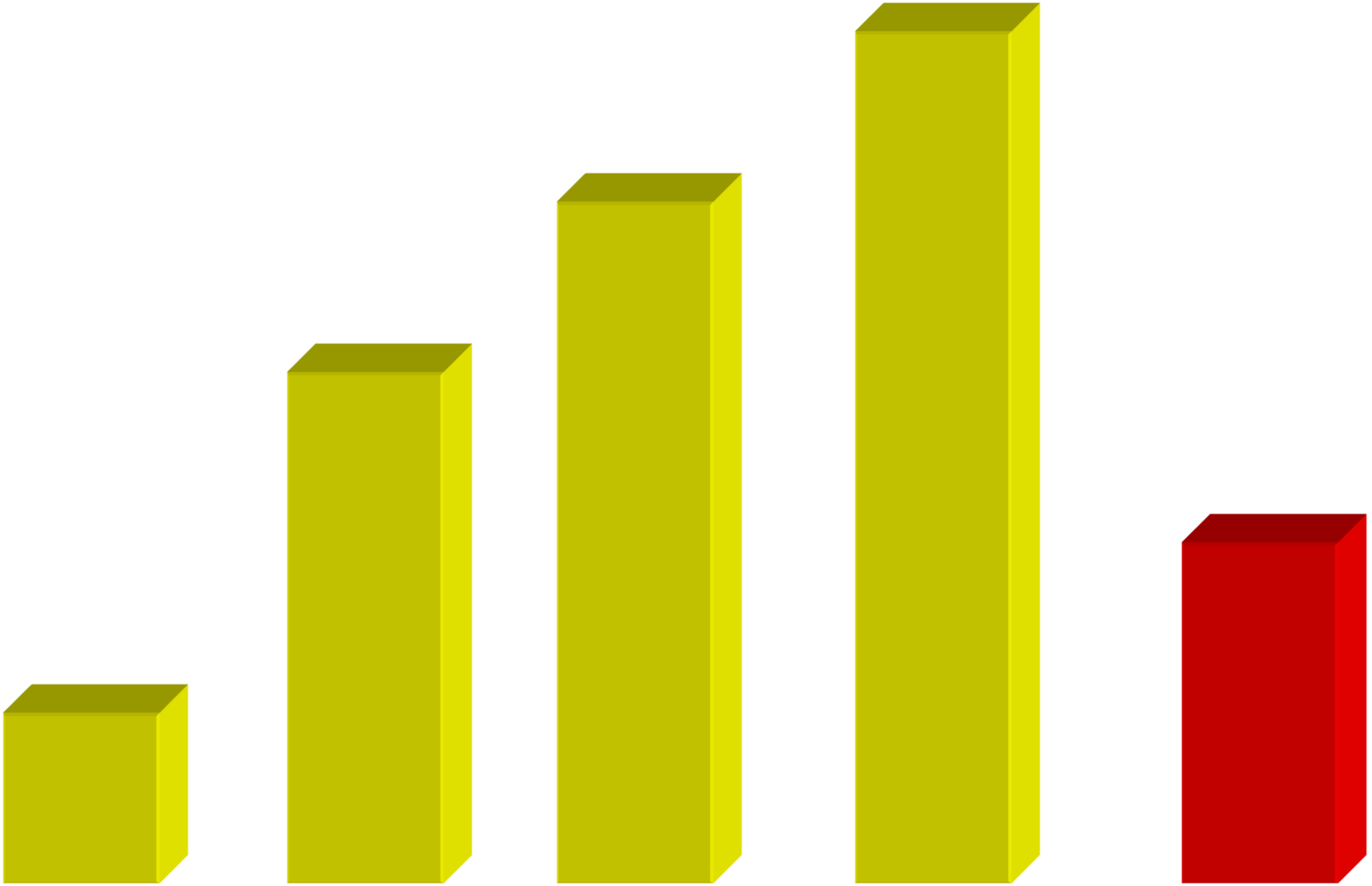
Insertion Sort



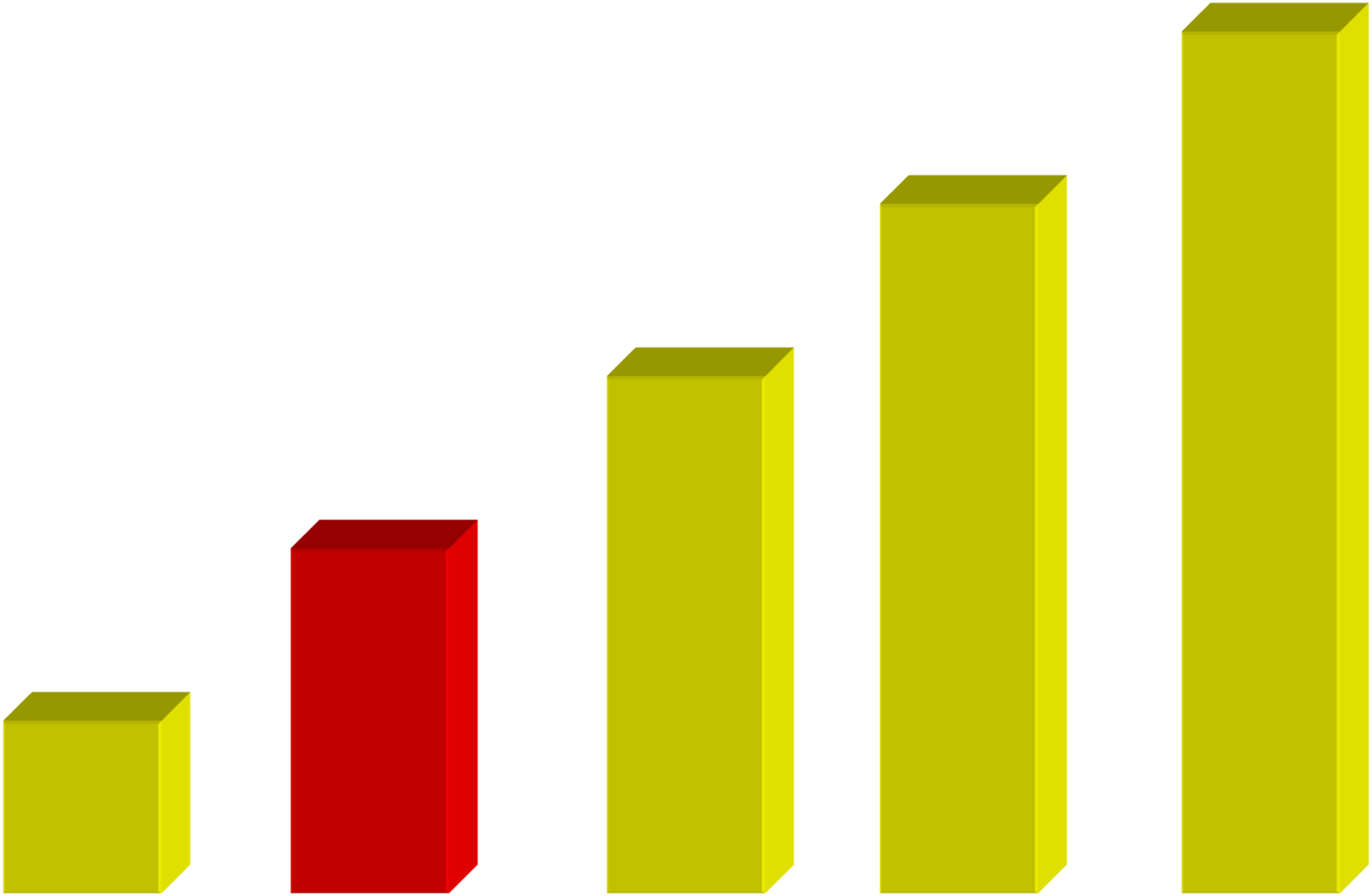
Insertion Sort



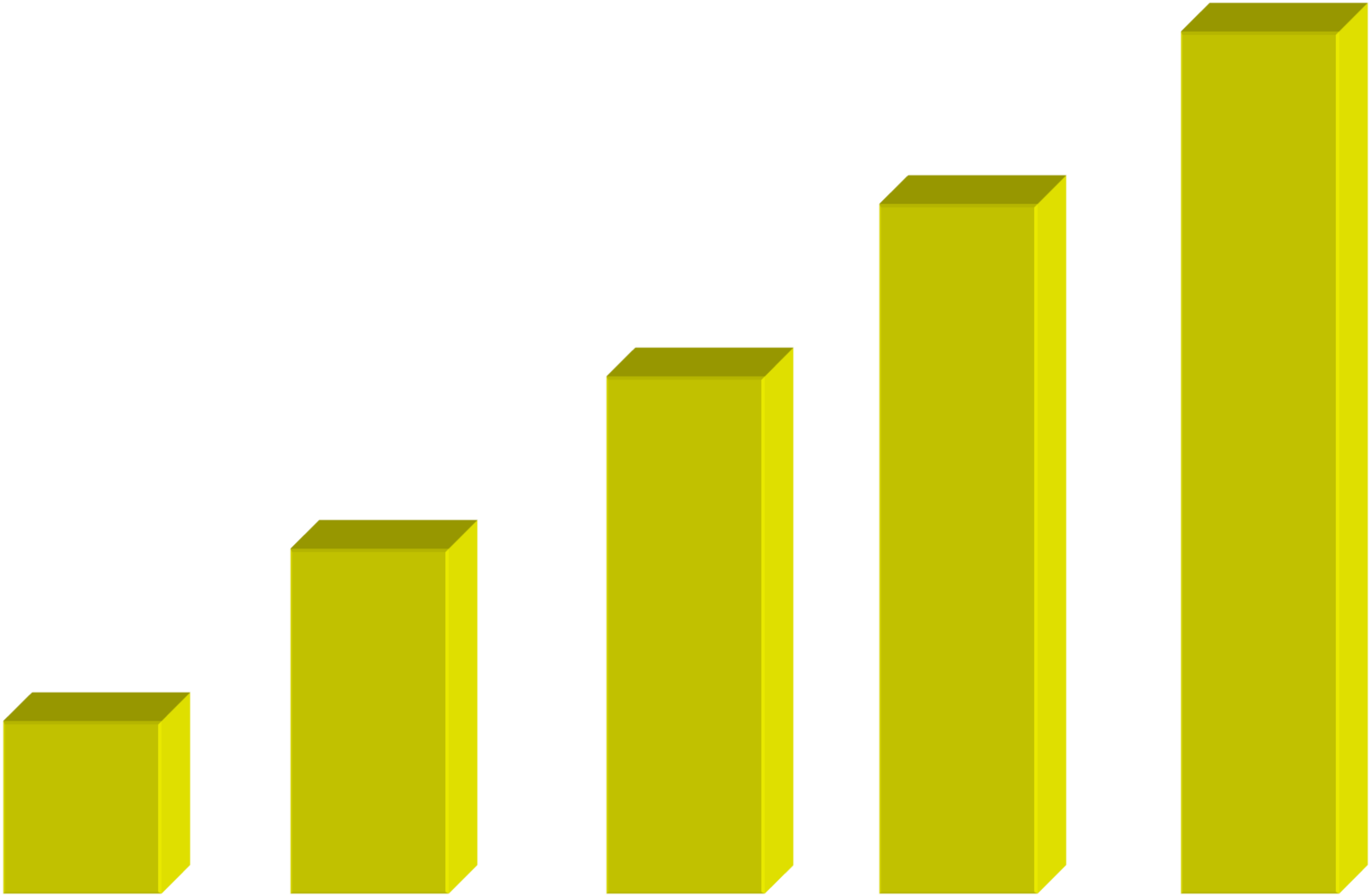
Insertion Sort



Insertion Sort



Insertion Sort



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

```
Algorithm insertionSort(A,n):
  Input: Array A of size n
  Output: Array A sorted

  for k ← 1 to n-1 do
      val ← A[k]
      j ← k-1
      while j ≥ 0 and A[j] > val do
          A[j+1] ← A[j]
          j ← j - 1
      end
      A[j+1] = val
  end
end
```

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

Dancing Sorts

- Selection sort - https://www.youtube.com/watch?v=Ns4TPTC8whw

- Bubble sort - https://www.youtube.com/watch?v=lyZQPjUT5B4

- Insertion sort - https://www.youtube.com/watch?v=ROalU379l3U