Sorting Algorithms

1. Seleksi
2. Gelembung
3. Penyisipan
4. Merge
5. Quick

Metode Seleksi

(one of the simplest sorting algorithms)

3	10	4	6	8	9	7	2	1	5

Cari dalam keseluruhan array, temukan nilai terbesar, (10) dan tukar nilai ini dengan nilai yang tersimpan dalam lokasi terakhir dari array (5)

3	**10**	4	6	8	9	7	2	1	**5**

3	**5**	4	6	8	9	7	2	1	**10**

Temukan nilai terbesar kedua dalam array (9), tukar dengan nilai yang tersimpan dalam lokasi terakhir kedua(1).

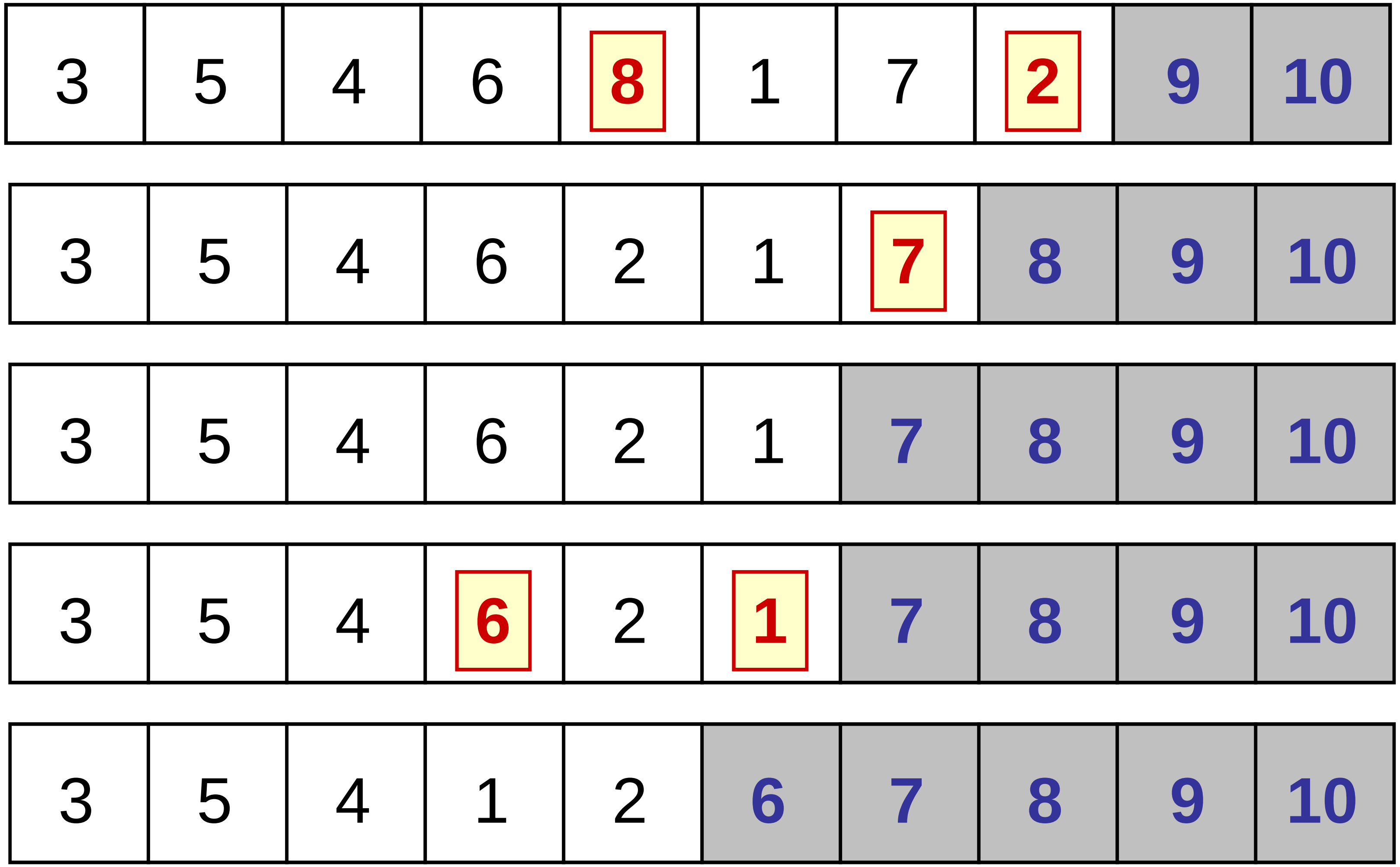
3	5	4	6	8	**9**	7	2	**1**	**10**

3	5	4	6	8	1	7	2	**9**	**10**

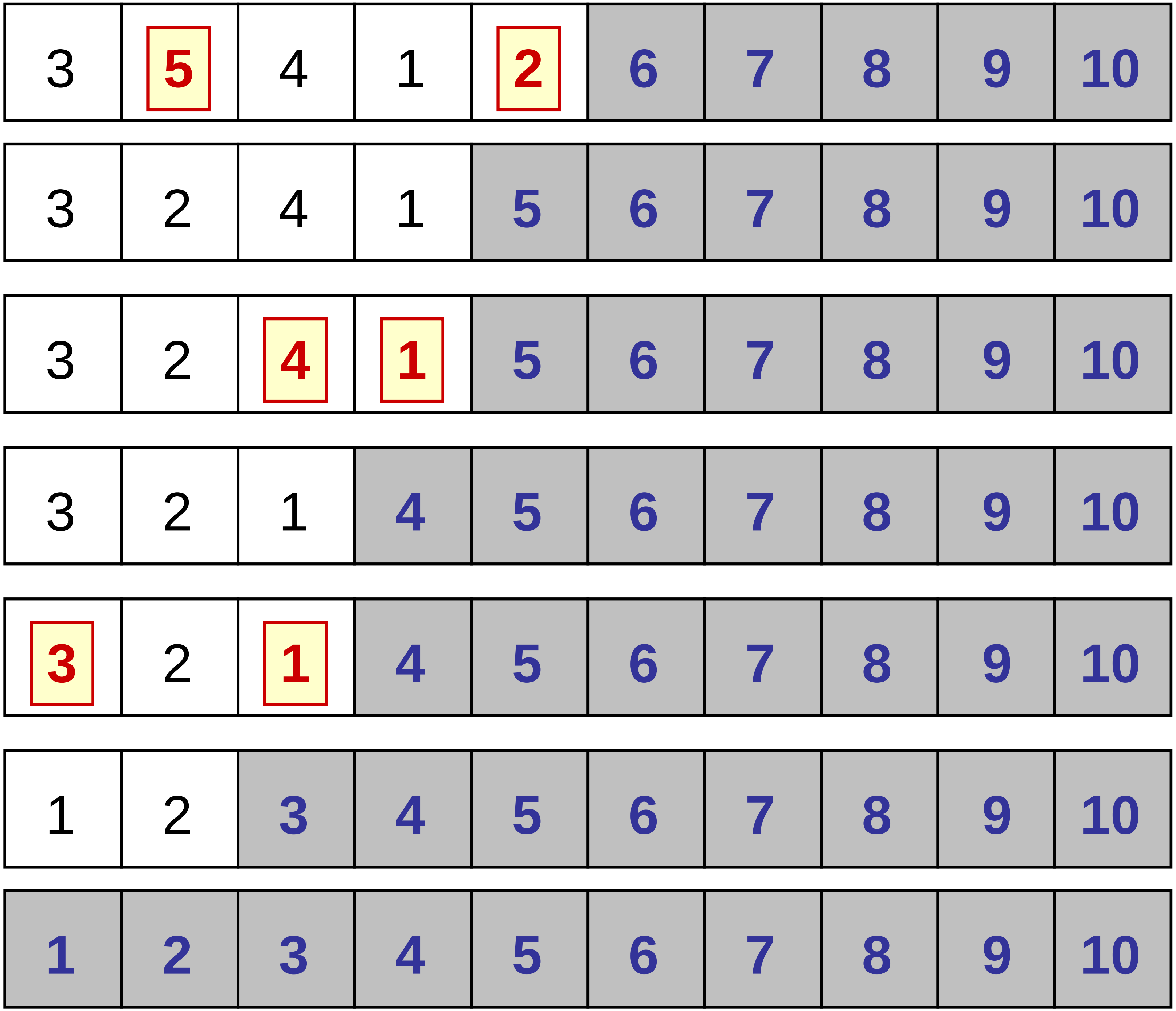
3	5	4	6	8	1	7	2	**9**	**10**

Dua nilai terakhir yang bertanda biru merupakan posisi yang pasti karena keduanya merupakan nilai terbesar dan nilai terbesar kedua.

Sekarang, ulangi proses “seleksi dan tukar” …



3	5	4	6	**8**	1	7	**2**	**9**	**10**
3	5	4	6	2	1	**7**	**8**	**9**	**10**
3	5	4	6	2	1	**7**	**8**	**9**	**10**
3	5	4	**6**	2	**1**	**7**	**8**	**9**	**10**
3	5	4	1	2	**6**	**7**	**8**	**9**	**10**



3	**5**	4	1	**2**	**6**	**7**	**8**	**9**	**10**
3	2	4	1	**5**	**6**	**7**	**8**	**9**	**10**
3	2	**4**	**1**	**5**	**6**	**7**	**8**	**9**	**10**
3	2	1	**4**	**5**	**6**	**7**	**8**	**9**	**10**
3	2	**1**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

Algoritma

1. Ulangi langkah 2-7 untuk i=n-1 s/d 0
2. Indeks=0
3. Ulangi langkah 4 untuk j=0 s/d i
4. Lakukan langkah 5-6 jika data[indeks]>data[j]
5. indeks=j
6. tukar data[i] dengan data[indeks]

Selection Sort → Analysis

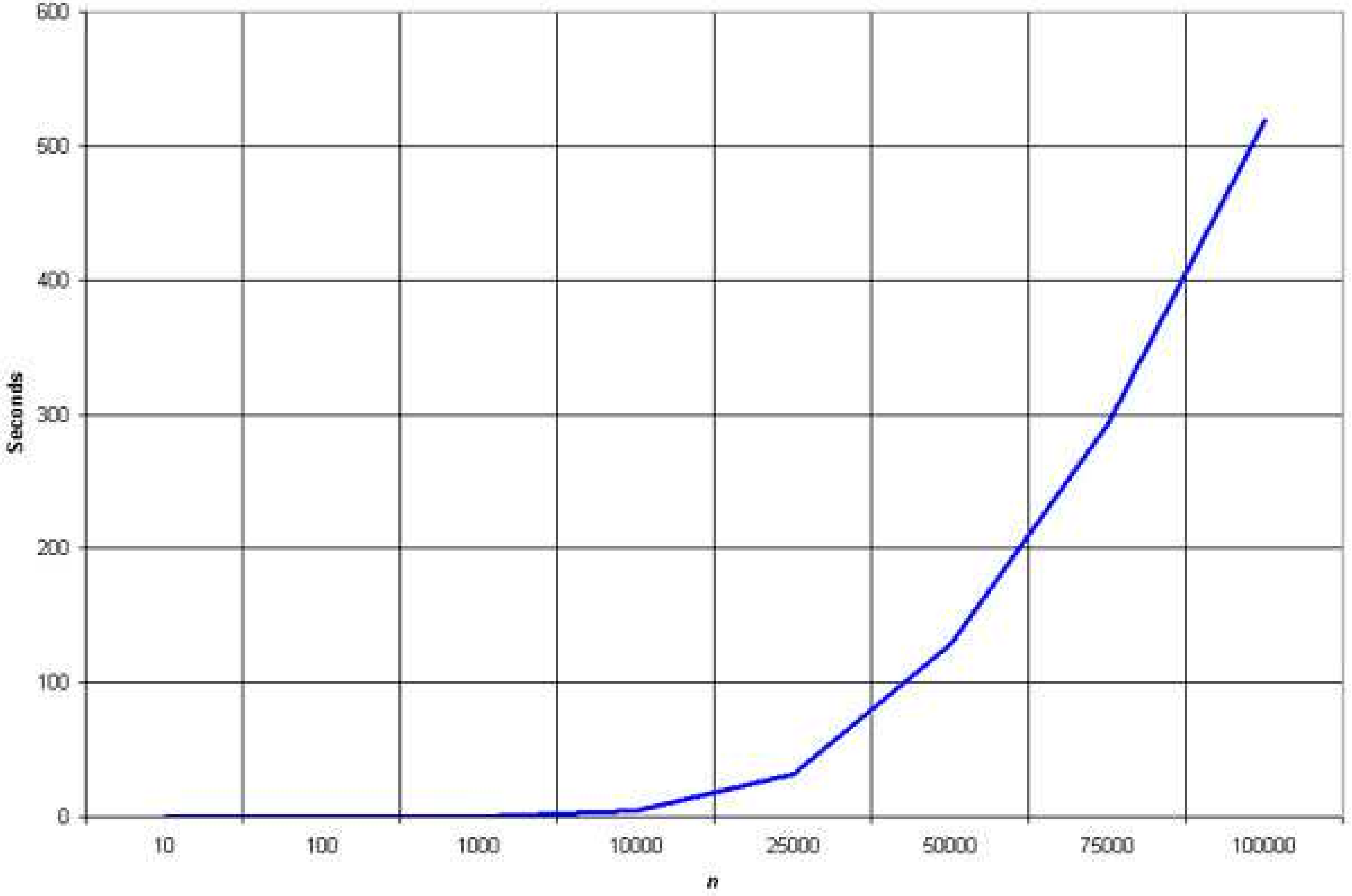
- Moves from left to right, putting elements into their final position without looking back

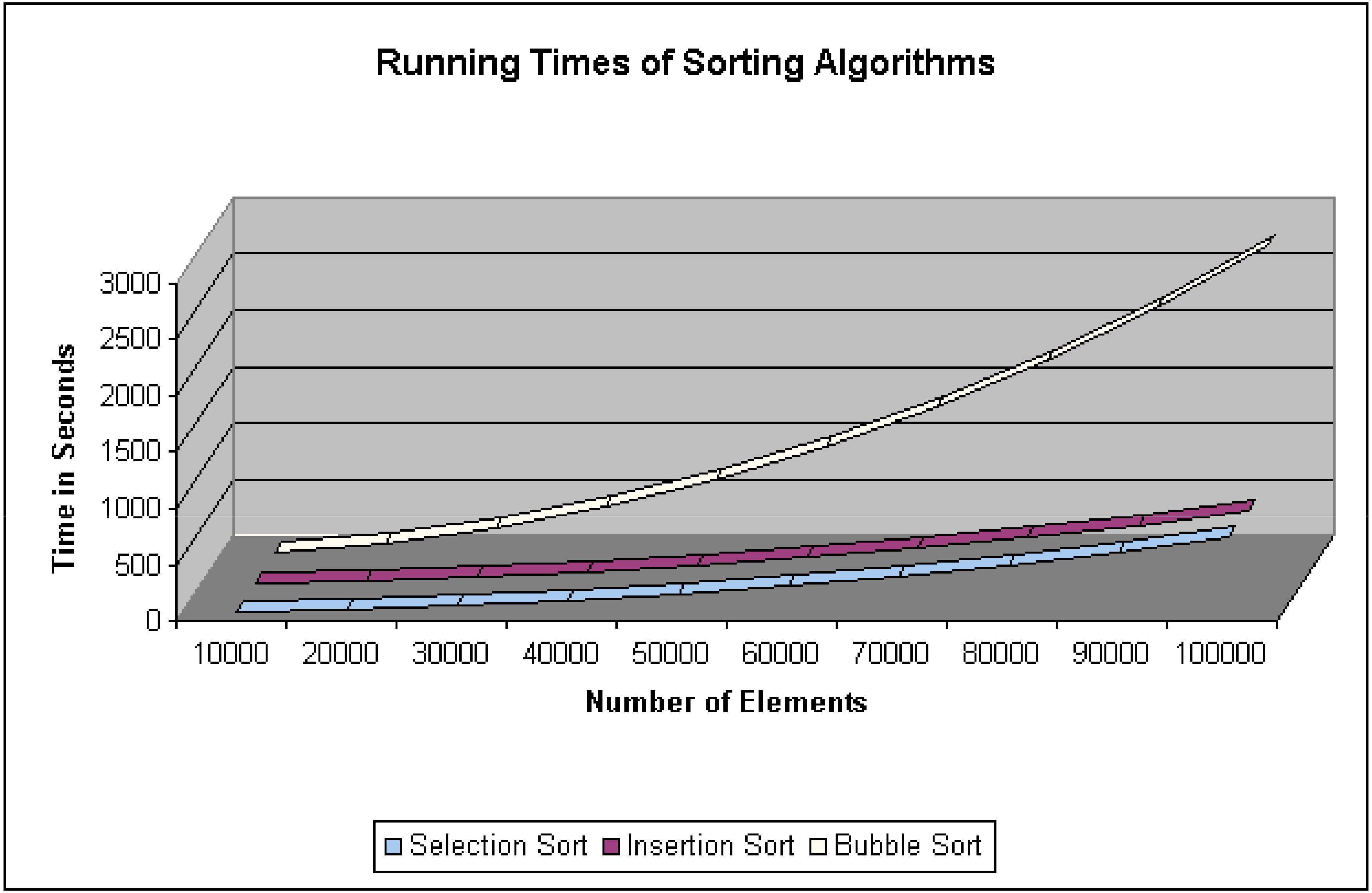
- Wastes most of the time looking for the minimum element in the unsorted part of the array

Selection Sort → Analysis

- $O(n^2)$ for worst and average cases
- Uses about $n^2/2$ comparisons
- Uses about n exchanges

Selection Sort → Empirical Analysis





Selection Sort yields a 60% performance improvement over the Bubble Sort