



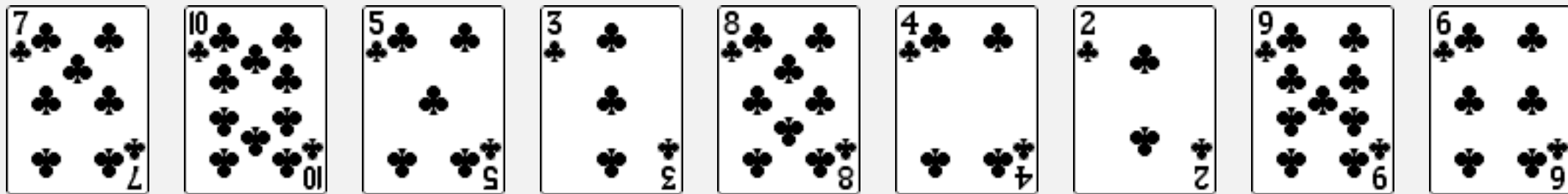
<http://algs4.cs.princeton.edu>

2.1 ELEMENTARY SORTS

- ▶ *rules of the game*
- ▶ *selection sort*
- ▶ *insertion sort*
- ▶ *shellsort*
- ▶ *shuffling*

Selection sort demo

- In iteration i , find index \min of smallest remaining entry.
- Swap $a[i]$ and $a[\min]$.



initial

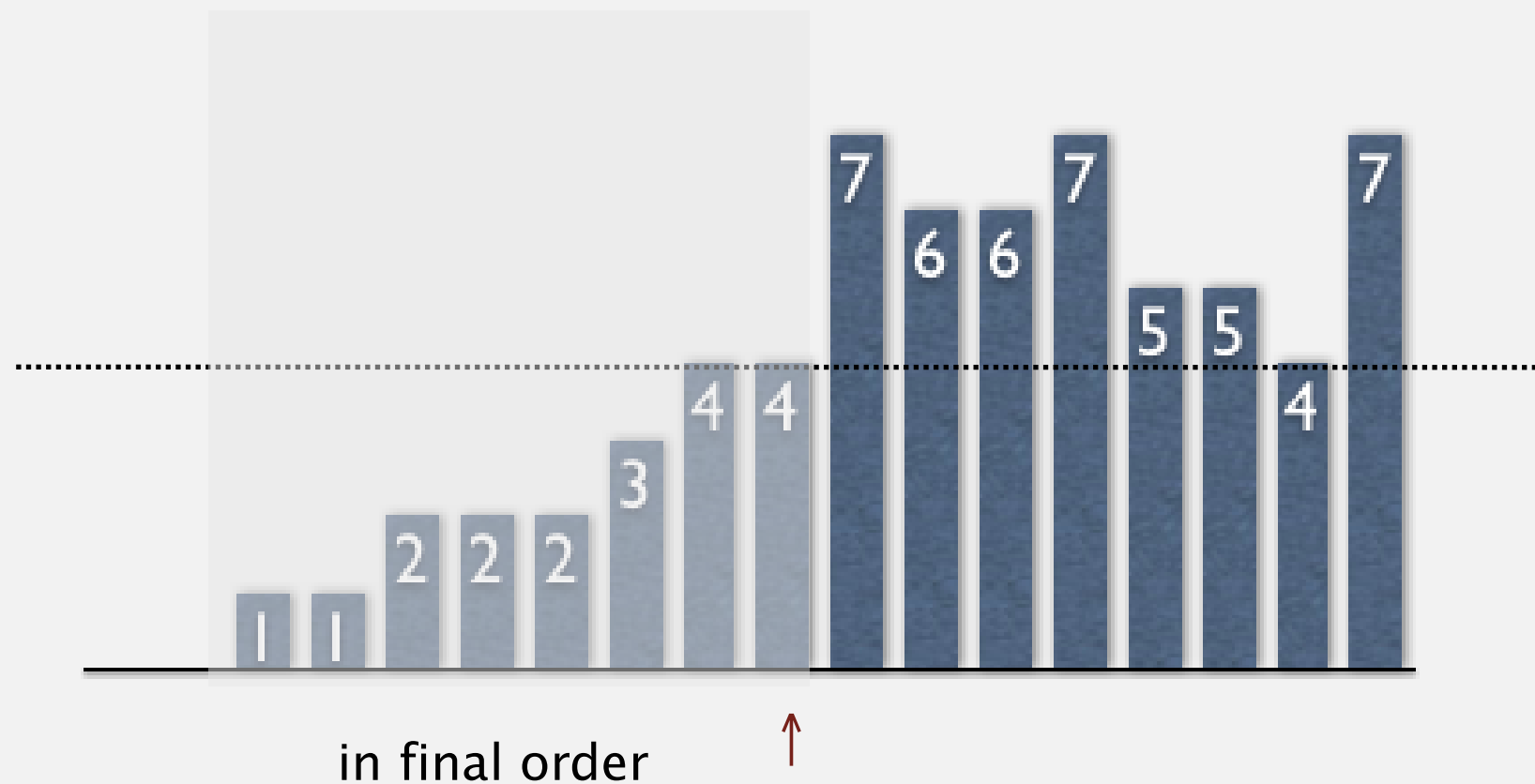


Selection sort

Algorithm. ↑ scans from left to right.

Invariants.

- Entries the left of ↑ (including ↑) fixed and in ascending order.
- No entry to right of ↑ is smaller than any entry to the left of ↑.



Two useful sorting abstractions

Helper functions. Refer to data through compares and exchanges.

Less. Is item v less than w ?

```
private static boolean less(Comparable v, Comparable w)
{   return v.compareTo(w) < 0;   }
```

Exchange. Swap item in array $a[]$ at index i with the one at index j .

```
private static void exch(Comparable[] a, int i, int j)
{
    Comparable swap = a[i];
    a[i] = a[j];
    a[j] = swap;
}
```

Selection sort inner loop

To maintain algorithm invariants:

- Move the pointer to the right.

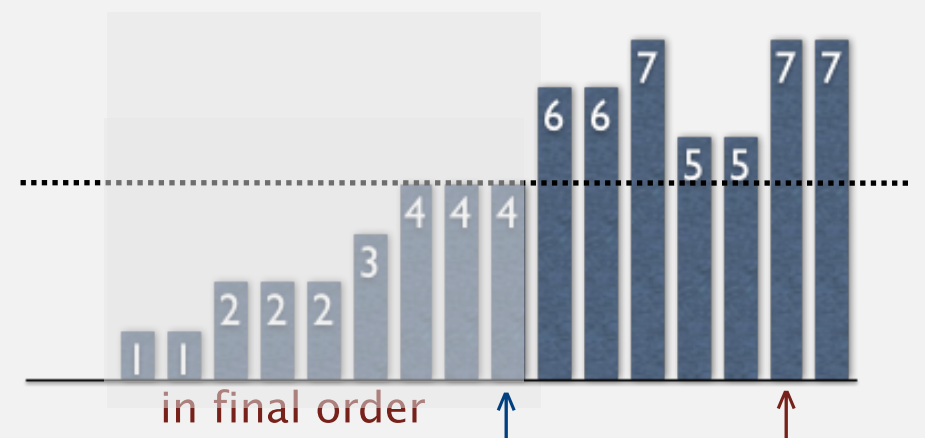
```
i++;
```

- Identify index of minimum entry on right.

```
int min = i;  
for (int j = i+1; j < N; j++)  
    if (less(a[j], a[min]))  
        min = j;
```

- Exchange into position.

```
exch(a, i, min);
```



Selection sort: Java implementation

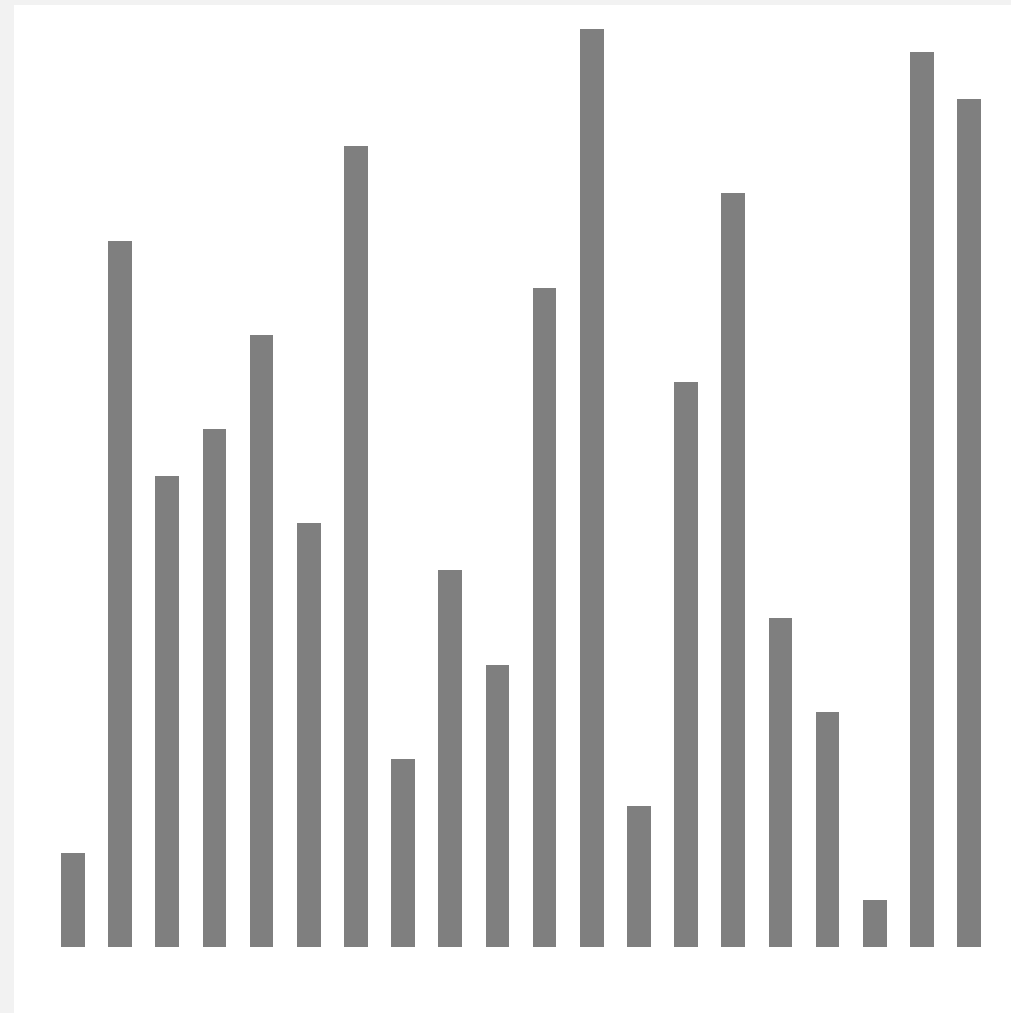
```
public class Selection
{
    public static void sort(Comparable[] a)
    {
        int N = a.length;
        for (int i = 0; i < N; i++)
        {
            int min = i;
            for (int j = i+1; j < N; j++)
                if (less(a[j], a[min]))
                    min = j;
            exch(a, i, min);
        }
    }

    private static boolean less(Comparable v, Comparable w)
    { /* as before */ }

    private static void exch(Comparable[] a, int i, int j)
    { /* as before */ }
}
```

Selection sort: animations

20 random items

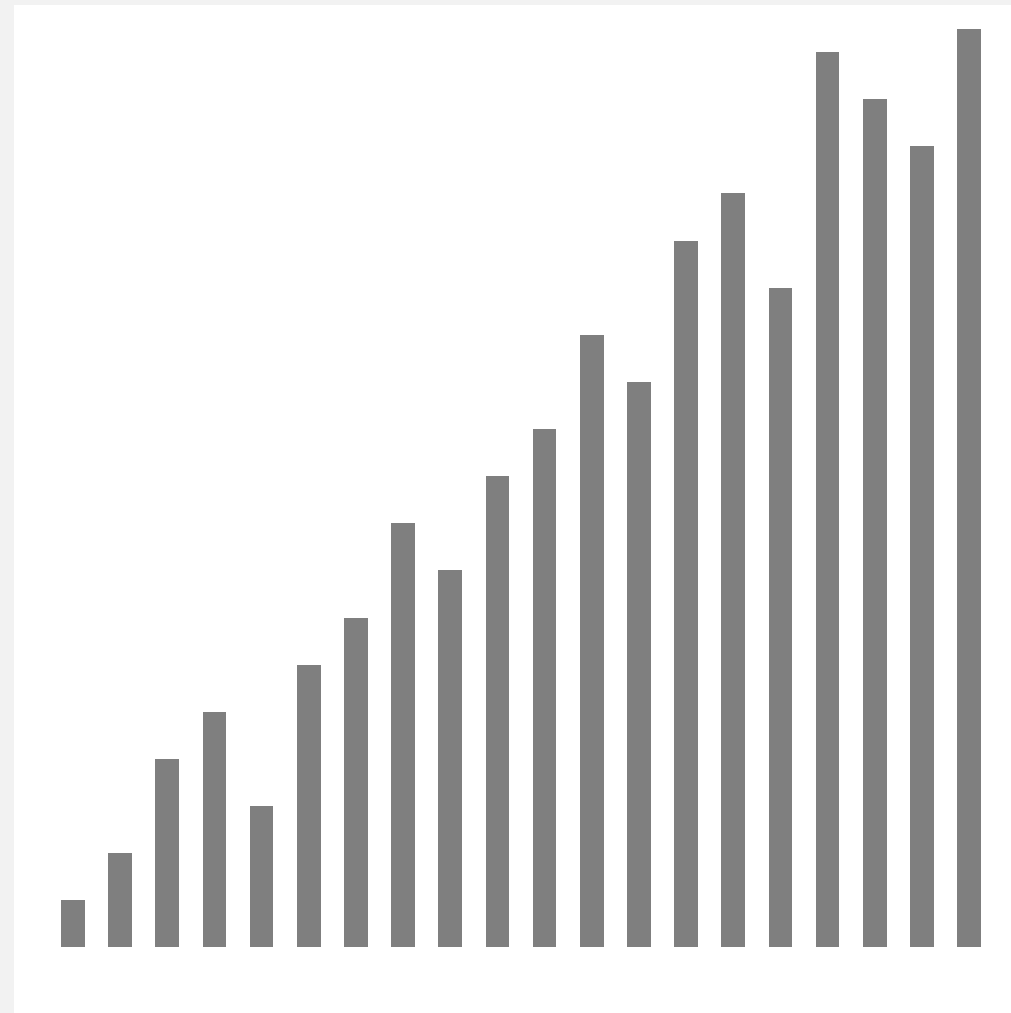


- ▲ algorithm position
- █ in final order
- █ not in final order

<http://www.sorting-algorithms.com/selection-sort>

Selection sort: animations

20 partially-sorted items



- ▲ algorithm position
- in final order
- not in final order

<http://www.sorting-algorithms.com/selection-sort>

Selection sort: mathematical analysis

Proposition. Selection sort uses $(N-1) + (N-2) + \dots + 1 + 0 \sim N^2/2$ compares and N exchanges.

| | | a[] | | | | | | | | | | |
|----|-----|-----|---|---|---|---|---|---|---|---|---|----|
| i | min | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | | S | O | R | T | E | X | A | M | P | L | E |
| 0 | 6 | S | O | R | T | E | X | A | M | P | L | E |
| 1 | 4 | A | O | R | T | E | X | S | M | P | L | E |
| 2 | 10 | A | E | R | T | O | X | S | M | P | L | E |
| 3 | 9 | A | E | E | T | O | X | S | M | P | L | R |
| 4 | 7 | A | E | E | L | O | X | S | M | P | T | R |
| 5 | 7 | A | E | E | L | M | X | S | O | P | T | R |
| 6 | 8 | A | E | E | L | M | O | S | X | P | T | R |
| 7 | 10 | A | E | E | L | M | O | P | X | S | T | R |
| 8 | 8 | A | E | E | L | M | O | P | R | S | T | X |
| 9 | 9 | A | E | E | L | M | O | P | R | S | T | X |
| 10 | 10 | A | E | E | L | M | O | P | R | S | T | X |
| | | A | E | E | L | M | O | P | R | S | T | X |

entries in black are examined to find the minimum

entries in red are a[min]

entries in gray are in final position

Trace of selection sort (array contents just after each exchange)

Running time insensitive to input. Quadratic time, even if input is sorted.
Data movement is minimal. Linear number of exchanges.