Brute Force

A straightforward approach, usually based **directly** on the problem's **statement and definitions** of the concepts involved

Examples – based directly on definitions:

1. Computing a^n ($a > 0$, n a nonnegative integer)

2. Computing $n!$

3. Multiplying two matrices

4. Searching for a key of a given value in a list

Sorting by Brute Force

Use definition of sorted and obvious algorithm?

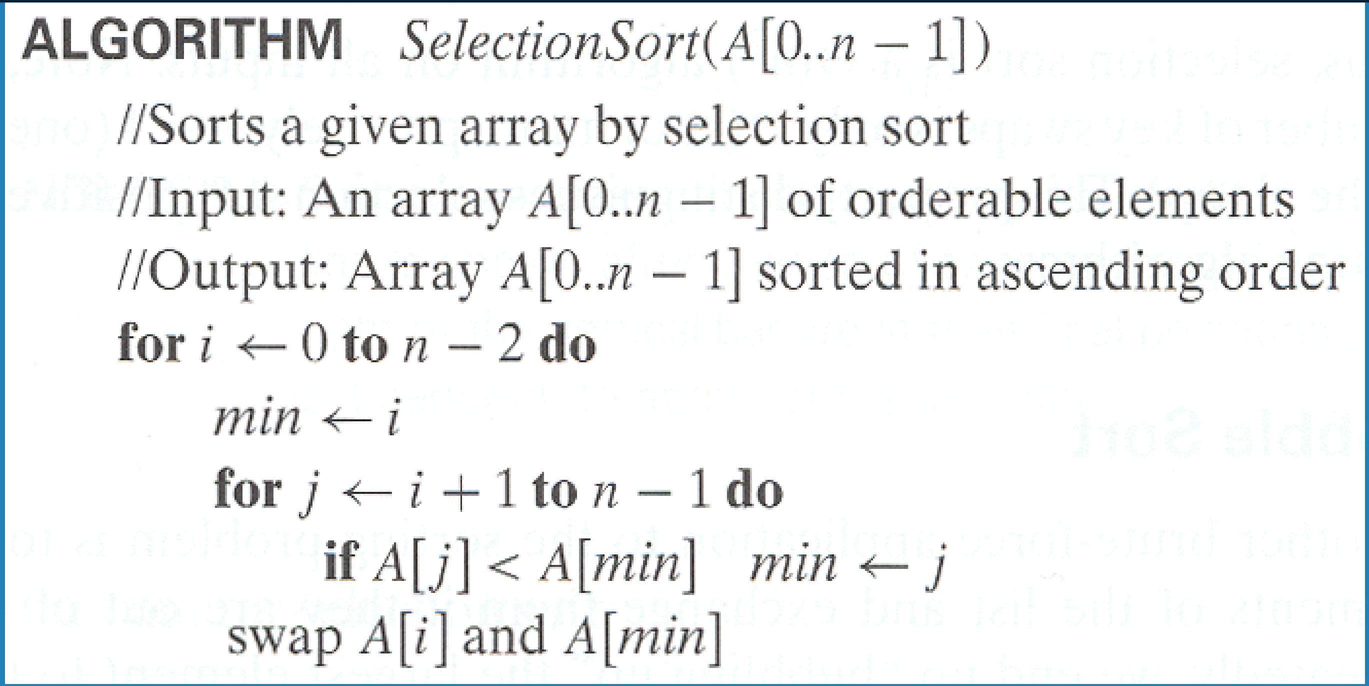
Selection Sort **Scan the array to find its smallest element and swap it with the first element. Then, starting with the second element, scan the elements to the right of it to find the smallest among them and swap it with the second elements. Generally, on pass *i* (0 ≤ *i* ≤ *n*-2), find the smallest element in *A*[*i*..*n*-1] and swap it with *A*[*i*]:**

$$A[0] \le \;.\;\;.\;\;.\; \le A[i\text{-}1] \;\mid\; A[i], \;.\;\;.\;\;.\;, A[min], \;.\;\;.\;\;., A[n\text{-}1]$$

in their final positions

Example: 7 3 2 5

Analysis of Selection Sort



```
ALGORITHM  SelectionSort(A[0..n − 1])
    //Sorts a given array by selection sort
    //Input: An array A[0..n − 1] of orderable elements
    //Output: Array A[0..n − 1] sorted in ascending order
    for i ← 0 to n − 2 do
        min ← i
        for j ← i + 1 to n − 1 do
            if A[j] < A[min]   min ← j
        swap A[i] and A[min]
```

Time efficiency: $C(n) = \sum_{i=} \quad \sum_{j=} 1$

Space efficiency: ?

Stability: