What is Brute Force?

- The first algorithm design technique we shall explore
- A straightforward approach to solving problem, usually based on problem statement and definitions of the concepts involved
- “Force” comes from using computer power not intellectual power
- In short, “brute force” means “Just do it!”

Brute Force Example

- We want to compute $a^n = \underbrace{a \times a \times \ldots \ldots \times a}_{\text{n times}}$
- First response: Multiply 1 by a n times which is the “Brute Force” approach.

Why Brute Force ?

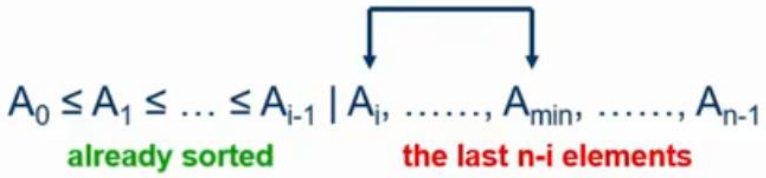
- We have already seen brute force algorithms:
  - Consecutive Integer Checking for gcd(m, n)
- It is the only general approach that always works
- Seldom gives efficient solution, but one can easily improve the brute force version.
- Usually can solve small sized instances of a problem
- A yardstick to compare with more efficient ones

Why Brute Force ?

- It is the only general approach that always works
- Seldom gives efficient solution, but one can easily improve the brute force version.
- Usually can solve small sized instances of a problem
- A yardstick to compare with more efficient ones

Brute force case studies

- Given n orderable items (e.g., numbers, characters, etc.) how can you rearrange them in non-decreasing order?
- Selection Sort:
  - On the i-th pass (i goes from 0 to n-2) the algo searches for the smallest item among the last n-i elements and swaps it with A_i



Brute Force: Selection Sort

```
ALGORITHM SelectionSort(A[0,..n-1])
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]
```

89	45	68	90	29	34	**17**

Brute Force: Selection Sort

ALGORITHM SelectionSort(A[0,..n-1])
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]

89	45	68	90	29	34	**17**
17 |	45	68	90	**29**	34	89
17	29 |	68	90	45	**34**	89
17	29	34	45 |	90	**68**	89
17	29	34	45	68 |	90	**89**
17	29	34	45	68	89 |	90

Input size: n, basic op: "<", does not depend on type

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$
$$= \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1]$$
$$= \frac{(n-1)n}{2}$$

Brute Force: Bubble Sort

- Compare adjacent elements and exchange them if out of order
- Essentially, it bubbles up the largest element to the last position

$A_0, \ldots \overset{?}{\ldots}, A_j <-> A_{j+1}, \ldots \ldots, A_{n-i-1} \mid A_{n-i} \leq \ldots \leq A_{n-1}$

Brute Force: Bubble Sort (contd.)

```
ALGORITHM BubbleSort(A[0..n-1])
for i <- 0 to n-2 do
        for j <- 0 to n-2-i do
                if A[j+1] < A[j]
                        swap A[j] and A[j+1]
```

What about 89, 45, 68, 90, 29, 34, 17 ?

Brute force case studies

- We saw two brute-force approach to sorting.
- Let's see brute-force to searching
- How would you search for a key, K in an array A[0..n-1]?

Sequential Search

ALGORITHM SequentialSearch(A[0..n-1], K)

//Output: index of the first element in A, whose //value is equal to K or -1 if no such element is found

```
i <- 0
while i < n and A[i] ≠ K do
        i <- i+1
if i < n
        return i
else
        return -1
```