Design and Analysis of Algorithms

Insertion Sort

National University of Computer and Emerging Sciences,
Islamabad

The problem of sorting

Input: sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

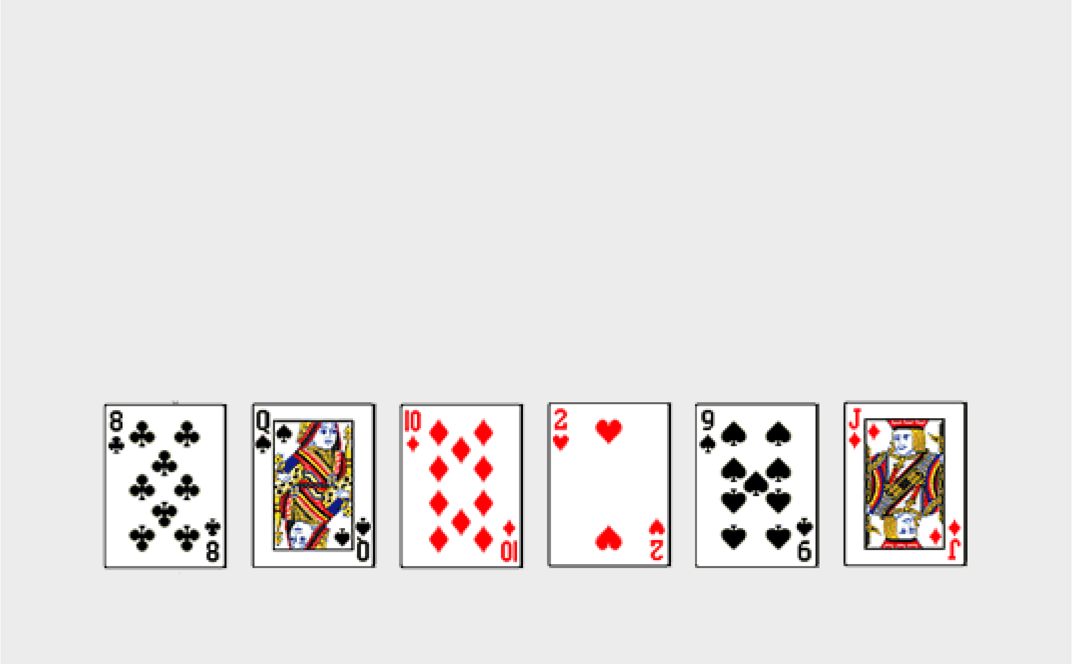
Output: permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ such that $a'_1 \le a'_2 \le \cdots \le a'_n$.

Example:

Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

Insertion Sort of Cards



Insertion Sort

- **The list is divided into two parts: sorted and unsorted**
- **In each pass, the following steps are performed**
  - **First element of the unsorted part (i.e., sub-list) is picked up**
  - **Transferred to the sorted sub-list**
  - **Inserted at the appropriate place**
- **A list of n elements will take at most n-1 passes to sort the data**

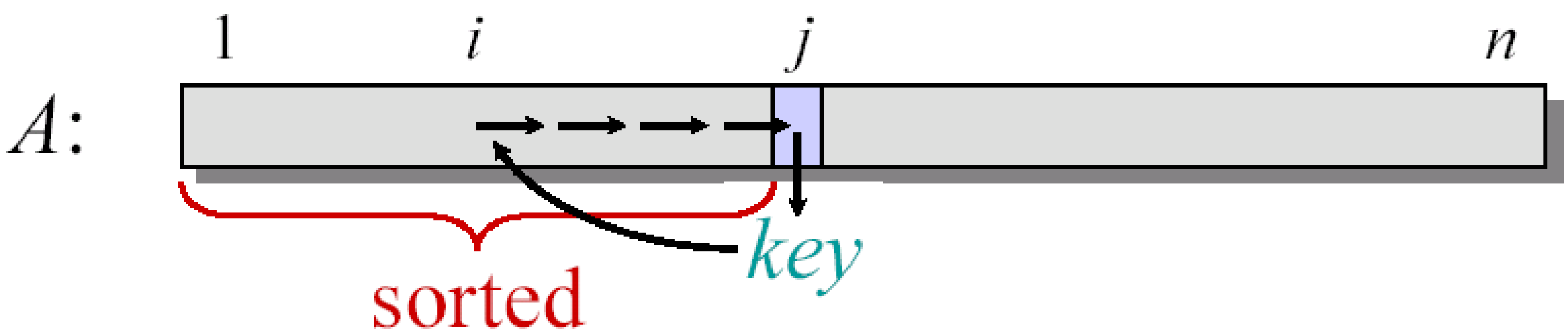
Animation

- https://visualgo.net/bn/sorting

Insertion Sort

"pseudocode"

```
INSERTION-SORT (A, n)        ▷ A[1 . . n]
    for j ← 2 to n
        do key ← A[ j]
           i ← j – 1
           while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
           A[i+1] = key
```



Example of Insertion Sort

8 2 4 9 3 6

Example of Insertion Sort

8 2 4 9 3 6

Example of Insertion Sort

8 2 4 9 3 6

2 8 4 9 3 6

Example of Insertion Sort

Example of Insertion Sort

8 2 4 9 3 6

2 8 4 9 3 6

2 4 8 9 3 6

Example of Insertion Sort

8 2 4 9 3 6

2 8 4 9 3 6

2 4 8 9 3 6

Example of Insertion Sort

8	2	4	9	3	6
2	8	4	9	3	6
2	4	8	9	3	6
2	4	8	9	3	6

Example of Insertion Sort

8	2	4	9	3	6
2	8	4	9	3	6
2	4	8	9	3	6
2	4	8	9	3	6

Example of Insertion Sort

8	2	4	9	3	6
2	8	4	9	3	6
2	4	8	9	3	6
2	4	8	9	3	6
2	3	4	8	9	6

Example of Insertion Sort

8	2	4	9	3	6
2	8	4	9	3	6
2	4	8	9	3	6
2	4	8	9	3	6
2	3	4	8	9	6

Example of Insertion Sort

8 2 4 9 3 6

2 8 4 9 3 6

2 4 8 9 3 6

2 4 8 9 3 6

2 3 4 8 9 6

2 3 4 6 8 9 *done*

Best Case

1	2	3	4	5

Worst Case

5	4	3	2	1

Running time of Insertion Sort

- The running time depends on the input: an already sorted sequence is easier to sort.
- Parameterize the running time by the size of the input,
  - short sequences are easier to sort than long ones.
- Generally, we seek upper bounds on the running time, because everybody likes a guarantee.

Kinds of Analyses

- Worst-case: (Usually)
  - T(n) = (maximum time of algorithm) on any input of size n.
- Average-case (Sometimes):
  - T(n) = (expected time of algorithm) over all inputs of size n.
  - Need assumption of statistical distribution of inputs
- Best-case: (bogus)
  - Cheat with a slow algorithm that works fast on some input

Machine-independent time: An example

A *pseudocode* for insertion sort (INSERTION SORT).

```
INSERTION-SORT(A)
for j ← 2 to length [A]
      do key ← A[ j]
      ∇ Insert A[j] into the sorted sequence A[1,..., j-1].
       i ← j – 1
       while i > 0 and A[i] > key
              do A[i+1] ← A[i]
                   i ← i – 1
     A[i +1] ← key
```

Analysis of Insertion Sort

- **Time to compute the running time as a function of the input size**

	cost	times
1.**for** j=2 **to** *length*(A)	c_1	n
2. **do** key=A[j]	c_2	n-1
3."insert A[j] into the sorted sequence A[1..j-1]"	0	n-1
4. i=j-1	c_4	n-1
5. **while** i>0 **and** A[i]>key	c_5	$\sum_{j=2}^{n} t_j$
6. **do** A[i+1]=A[i]	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7. i--	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8. A[i+1]=key	c_8	n-1

Insertion-Sort Running Time

$$T(n) = c_1 \bullet (n) + c_2 \bullet (n\text{-}1) + c_3 \bullet (n\text{-}1) + c_4 \bullet (n\text{-}1) + c_5 \bullet (\Sigma_{j=2,n}\ t_j) + c_6 \bullet (\ \Sigma_{j=2,n}\ (t_j\ \text{-}1)\ ) + c_7 \bullet (\ \Sigma_{j=2,n}\ (t_j\ \text{-}1)\ ) + c_8 \bullet (n\text{-}1)$$

$c_3 = 0$, of course, since it's the comment

Best/Worst/Average Case

- **Best case:**

elements already sorted, $t_j=1$,
running time = $f(n)$, i.e., *linear* time.

- **Worst case:**

elements are sorted in inverse order,
$t_j=j$, running time = $f(n^2)$, i.e., *quadratic* time

- **Average case:**

$t_j=j/2$, running time = $f(n^2)$,
i.e., *quadratic* time

Best Case Result

Occurs when array is already sorted.

For each j = 2, 3,…..n we find that A[i]<=key in line 5 when i has its initial value of j-1.

$$
\begin{aligned}
\mathbf{T(n)} &= c_1 \cdot n + (c_2 + c_4) \cdot (n-1) + c_5 \cdot (n-1) + c_8 \cdot (n-1) \\
&= n \cdot (c_1 + c_2 + c_4 + c_5 + c_8) \\
&\quad + (-c_2 - c_4 - c_5 - c_8) \\
&= c_9 n + c_{10} \\
&= f_1(n^1) + f_2(n^0)
\end{aligned}
$$

Worst Case T(n)

- Occurs when the loop of lines 5-7 is executed as many times as possible, which is when A[] is in reverse sorted order.
- key is A[j] from line 2
- i starts at j-1 from line 4
- i goes down to 0 due to line 7
- So, t_j in lines 5-7 is [(j-1) – 0] + 1 = j

The '1' at the end is due to the test that fails, causing exit from the loop.

Worst Case T(n), ctd.

$$T(n) = c_1 \bullet [n] + c_2 \bullet (n-1) + c_4 \bullet (n-1) + c_5 \bullet (\Sigma_{j=2,n}\ j) + c_6 \bullet [\ \Sigma_{j=2,n}\ (j - 1)\ ] + c_7 \bullet [\ \Sigma_{j=2,n}\ (j -1)\ ] + c_8 \bullet (n-1)$$

Worst Case T(n), ctd.

$$
\begin{aligned}
T(n) = {} & \\
& c_1 \bullet n + c_2 \bullet (n-1) + c_4 \bullet (n-1) + c_8 \bullet (n-1) \\
& + c_5 \bullet (\Sigma_{j=2,n}\ j) \\
& + c_6 \bullet [\ \Sigma_{j=2,n}\ (j-1)\ ] + c_7 \bullet [\ \Sigma_{j=2,n}\ (j-1)\ ]
\end{aligned}
$$

$$
\begin{aligned}
= {} & c_9 \bullet n + c_{10} + c_5 \bullet (\Sigma_{j=2,n}\ j) + c_{11} \bullet \\
& [\Sigma_{j=2,n}\ (j-1)\ ]
\end{aligned}
$$

Worst Case T(n), ctd.

$T(n) = \quad c_9 \bullet n + c_{10} + c_5 \bullet (\Sigma_{j=2,n}\, j) + c_{11} \bullet [\ \Sigma_{j=2,n}\, (j - 1)\ ]$

But

$\Sigma_{j=2,n}\, j = [n(n+1)/2] - 1$

so that

$\Sigma_{j=2,n}\, (j-1) = \Sigma_{j=2,n}\, j - \Sigma_{j=2,n}\, (1)$

$= [n(n+1)/2] - 1 - (n-2+1)$

$= [n(n+1)/2] - 1 - n + 1 = n(n+1)/2 - n$

$= [n(n+1)-2n]/2 = [n(n+1-2)]/2 = n(n-1)/2$

Wasn't that fun?

Worst Case T(n), ctd.

In conclusion,

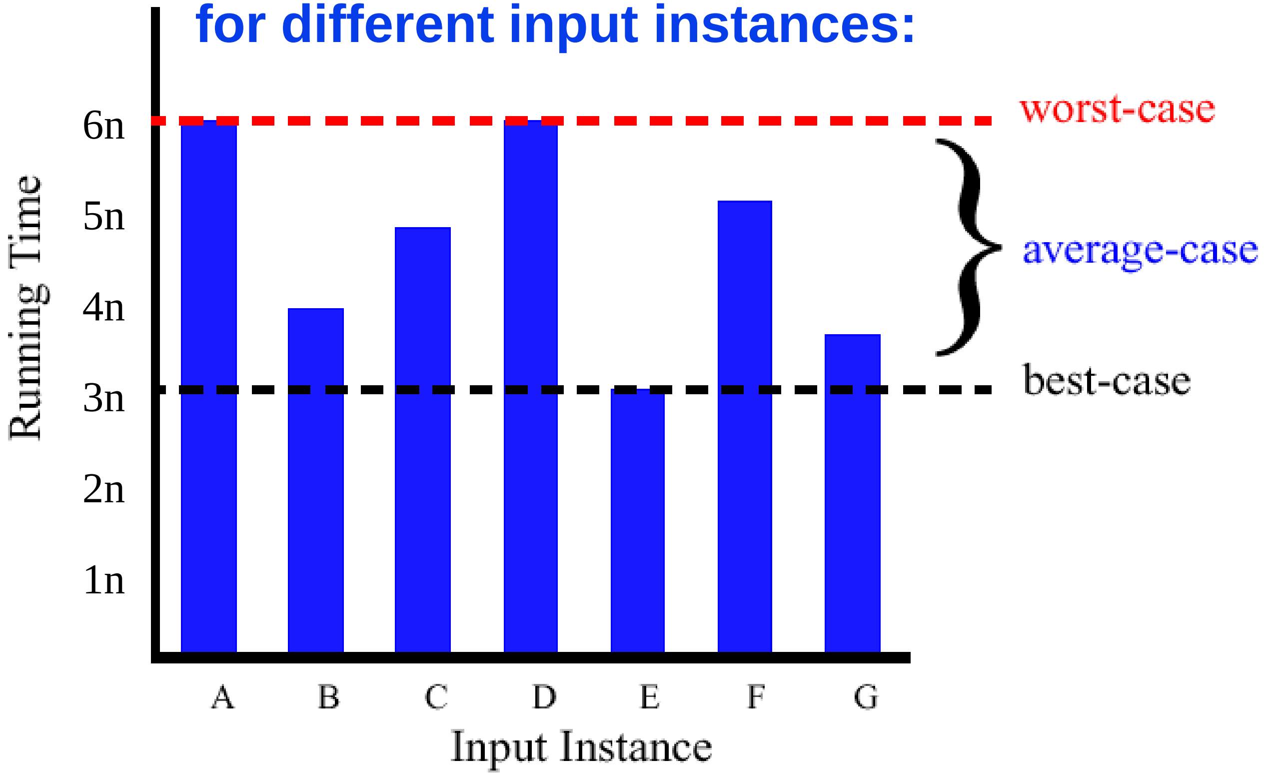
$$T(n) = c_9 \bullet n + c_{10} + c_5 \bullet [n(n+1)/2] - 1 + c_{11} \bullet n(n-1)/2$$

$$= c_{12} \bullet n^2 + c_{13} \bullet n + c_{14}$$

$$= f_1(n^2) + f_2(n^1) + f_3(n^0)$$

Best/Worst/Average Case (2)

- **For a specific size of input *n*, investigate running times for different input instances:**



Insertion Sort Analysis

- **Is insertion sort a fast sorting algorithm?**
  - **Moderately so, for small n**
  - **Not at all, for large n**
  - **sorting "almost sorted" lists**

Reference

- **Introduction to Algorithms**
- **Chapter # 2**
  - **Thomas H. Cormen**
  - **3rd Edition**

- **https://visualgo.net/bn/sorting**