Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

References

1. Introduction to Algorithms, T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, MIT Press, 2001.
2. Foundations of Algorithms Using C++ Pseudocode, R. E. Neapolitan ans K. Naimipour, 1998.
3. The Algorithm Design Manual, S. S. Skiena, 2008.
4. Algorithms, S. Dasgupta, C. Papadimitriou, and U. Vazirani, 2008.
5. The Design and Analysis of Computer Programs, Aho, Hopcropt, and Ullman, 1974.
6. Computer Algorithms: Introduction to design and Analysis, G. Brassard.

References

1. Introduction to Algorithms, T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, MIT Press, 2001.
2. Foundations of Algorithms Using C++ Pseudocode, R. E. Neapolitan ans K. Naimipour, 1998.
3. The Algorithm Design Manual, S. S. Skiena, 2008.
4. Algorithms, S. Dasgupta, C. Papadimitriou, and U. Vazirani, 2008.
5. The Design and Analysis of Computer Programs, Aho, Hopcropt, and Ullman, 1974.
6. Computer Algorithms: Introduction to design and Analysis, G. Brassard.

Grading

Final exam.	40%
Mid-Term exam. (1394/08/24)	20%
Exercises	30%
Teacher Assistant	10%

Algorithm

Definition (Algorithm)

An algorithm is a finite set of precise instructions for performing a calculation or solving a problem.

Algorithm

Definition (Algorithm)

An algorithm is a finite set of precise instructions for performing a calculation or solving a problem.

Important properties of the Algorithms:

1. Input
2. Output
3. Definiteness
4. Correctness
5. Finiteness
6. Effectiveness
7. Generality

Goals

The goals of this cours:

1. How to devise algorithms?
2. How to express algorithms?
3. How to validate algorithms?
4. How to analyze algorithms?
5. How to test algorithms?

Goals

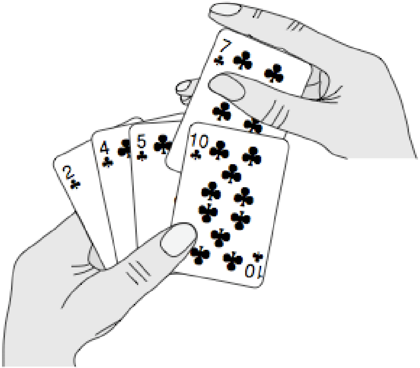
The goals of this cours:

1. How to devise algorithms?
2. How to express algorithms?
3. How to validate algorithms?
4. How to analyze algorithms?
5. How to test algorithms?

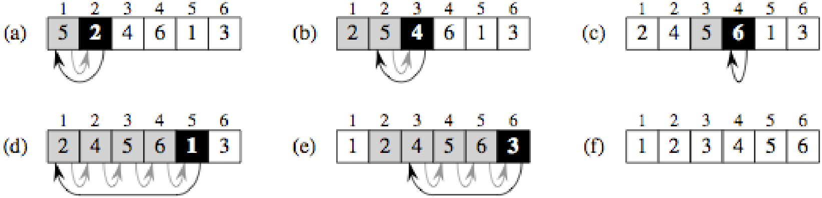
The following techniques will be discussed in this course:

1. Divide and Conquer
2. Dynamic Programming
3. Greedy Algorithms
4. Backtracking Algorithms
5. Branch and Bound Algorithms

Insertion Sort: How to devise?



Insertion Sort: How to devise?



Insertion Sort: How to express?

```
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
```

Insertion Sort: How to validate?

Theorem

After the termination of InsertionSort algorithm, the input array A is sorted.

Insertion Sort: How to validate?

Theorem

After the termination of InsertionSort algorithm, the input array A is sorted.

Lemma

At the start of each iteration of the for loop of lines 1-8, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$ but in sorted order.

Insertion Sort: How to validate?

Theorem

After the termination of InsertionSort algorithm, the input array A is sorted.

Lemma

At the start of each iteration of the for loop of lines 1-8, the subarray $A[1..j-1]$ *consists of the elements originally in* $A[1..j-1]$ *but in sorted order.*

Proof.

Proof based on induction (Loop Invariant, in this case).

- **Initialization:** $j = 2 \Longrightarrow A[1..1] = A[1]$, which is sorted.
- **Maintenance:** $A[j]$ is inserted in the correct position, so $A[1..j]$ is sorted.
- **Termination:** This happens when $j = n+1$. So $A[1..j-1] = A[1..n]$ is an ordered array.

$\square$

Insertion Sort: How to analyze?

Computing the amount of resources (Time, Space, etc.) needed by the algorithm.

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

where t_j is the number of times the while loop in line 5 is executed for that value of j. So:

$$T(n) = c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5 \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n} t_j - 1.$$

Insertion Sort: How to analyze?

$$T(n) = c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5 \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n} t_j - 1.$$

- **Best Case:** The input array is already sorted, so $t_j = 1$ and we have:

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

- **Worse Case:** The input array is already sorted in reverse order, so $t_j = j$ and we have:

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_4 + c_8)(n-1) + c_5(\frac{n(n+1)}{2} - 1) \\ &\quad + (c_6 + c_7)(\frac{n(n-1)}{2}) \\ &= (\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2})n^2 + (c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} \\ &\quad - \frac{c_7}{2} + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

Insertion Sort: How to test?

Just implement the pseudocode in any programming language and execute it with different instances of random arrays as input...

Exercises

1. Answer to the five mentioned questions for the following problems:
    a. Bubble Sort
    b. Sequential Search
    c. Binary Search