Lecture 1:

Basics of Algorithm Analysis

Course: Algorithms for Big Data

Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Spring 2021

Outline

- Basic Definitions
- Input Size
- Time Complexity
- Space Complexity
- Randomization
- Approximation

Basic Definitions I

An algorithm is a sequence of simple operations (addition, multiplication, comparison, ...) performed on input data to produce desired results.

Input $\Rightarrow$ Algorithm $\Rightarrow$ Output

Examples for Input Data:

- A list of n integers
- A graph with n nodes and m edges
- A n by m matrix
- A large text

Some Specific Tasks:

- Sorting
- Finding shortest paths
- Rank of a matrix
- Most frequent word

Basic Definitions

- Size of input (in bits or numbers)
- Size of output
- Time complexity (worst-case, average)
- Space Complexity
- Deterministic or Randomized?
- Exact or Approximate

Input Size

A few examples:

- An integer a: Task: Check if a is a prime.

  Input Size: $n = \log_2 a$ bits

- A list of n integers:

  Input Size: n numbers

- An undirected graph with t vertices and m edges:

  Input Size: depends on the representation.

  Adjacency Matrix: $n = t^2$ bits

  Adjacency list: $n = t + 2m$ numbers

- A t by t matrix:

  Input Size: $n = t^2$ numbers

Time Complexity

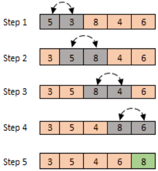
Time complexity of an algorithm: Maximum number of basic steps (addition, multiplication, comparison, ...) an algorithm takes on a given input of size n

$$\text{A function of the input size: } T(n)$$

Known as the worst-case time complexity.

Example:

Bubble Sort



Time complexity = $O(n^2)$

Quadratic time

Big O Notation

- (Informally speaking) $O(f(n))$ includes all functions with the leading term asymptotically smaller than or equal to $f(n)$ after eliminating the constant coefficient.

  $O(n^2) = \{100n^2,\ \ n^2 + 10n,\ \ 4n\log n + 2n,\ \ \log^2 n - 1, \ldots\}$

  $$n^3 \notin O(n^2),\ \ n^2 \log n \notin O(n^2),\ \ 2^n \notin O(n^2)$$

- $\Omega(f(n))$ includes all functions with the leading term asymptotically bigger than or equal to $f(n)$ after eliminating the constant coefficient.

  $\Omega(n^2) = \{100n^2,\ \ n^2 + 10n,\ \ n^3 + n^2 + 1,\ \ 2^n + 1, \ldots\}$

  $$2n \notin \Omega(n^2),\ \ n^{1.9} \notin \Omega(n^2),\ \ n^2/\log n \notin \Omega(n^2)$$

- $o(f(n))$ includes all functions with the leading term asymptotically bigger than $f(n)$ after eliminating the constant coefficient.

  $o(n) = \{100n^{0.99},\ \ 2\log n + 10,\ \ n/\log n + 1,\ \ \ldots\}$

  $$n \notin o(n),\ \ n\log n \in o(n^2),\ \ 2^n \in o(3^n)$$

Typical Running Times

- Polynomial time: $O(n^c)$ when c is a constant.
- Linear time: $O(n)$
- Quadratic time: $O(n^2)$
- Qubic time: $O(n^3)$
- Exponential time: $O(2^n)$
- Sublinear time: $o(n)$
- Logarithmic time: $O(\log n)$
- Constant time: $O(1)$

Running Time Comparison

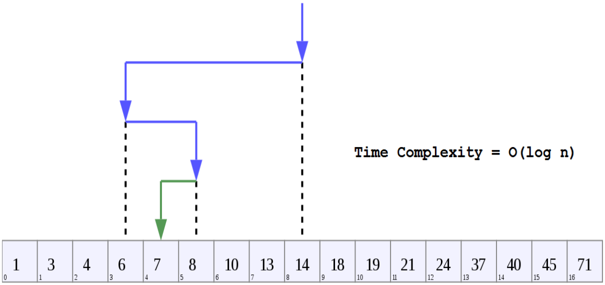
Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Source: Algorithm Design, Jon Kleinberg, Eva Tardos, 2006.

Sublinear Time: Example

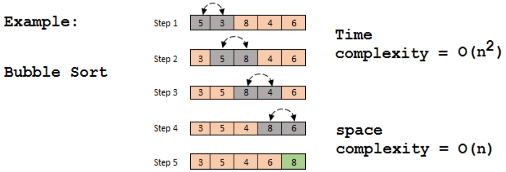
Searching in a sorted array: Binary Search



Space Complexity I

The space complexity of an algorithm is the amount of memory the algorithm uses during its execution (measured in bits or numbers).

A function of the input size: $S(n)$



Space Complexity II

In the RAM Model (Random Access Memory) Model it is assumed the entire input is saved in the memory.

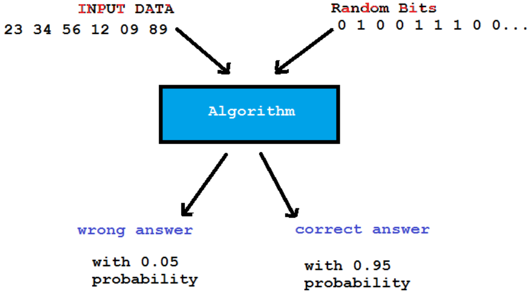
$\Rightarrow$ Space Complexity $\geq$ Input Size

We shall see in the Data Stream Model and the MPC (Massively Parallel Computation) Model the entire input is not present in the memory.

Here the Space Complexity could be smaller than the Input Size. Sublinear Space Complexity

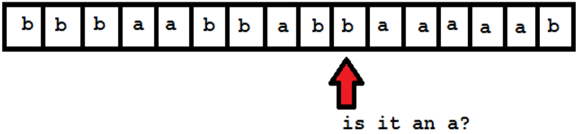
Deterministic or Randomized?

A randomized algorithm in addition to the input uses a series of random bits for its computation. The algorithm produces the desired output with a certain success probability. It fails with a certain probability.



Randomized algorithm: example

Problem: Given an array A of length n containing $n/2$ number of a's and $n/2$ number of b's, find the position of an a.



Randomized Algorithm I: Randomly pick a position and check if it is an a. Do this 3 times. If no a is found declare failure.

Analysis: $\Pr[\text{failure}] = 1/2 \times 1/2 \times 1/2 = 1/8$. The algorithm succeeds with probability 7/8. Running time $= O(1)$.

Assumption: Generating a random number between 0 and n takes $O(1)$ time.

Randomized algorithm: example continued

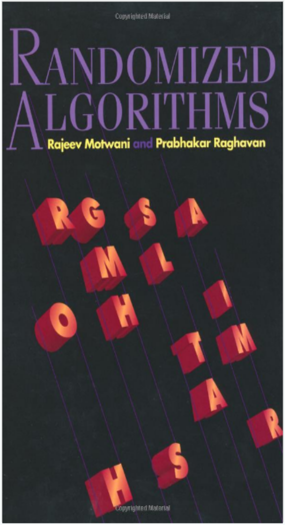
Randomized Algorithm II: Randomly pick a position and check if it is an a. Repeat this until an a is found.

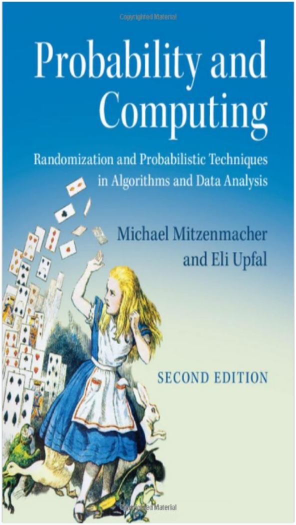
Analysis: $\Pr[\text{failure}] = 0$. The algorithm succeeds with probability 1.

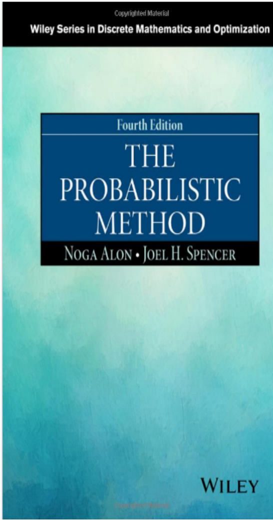
$$\text{Expected running time} = \sum_{i}^{\infty} \frac{i}{2^i} = 2$$

- Las Vegas Randomized Algorithm: Zero Failure Prob.
- Monte Carlo Randomized Algorithms: Positive Failure Prob.

Randomized Algorithms: books







Exact or Approximation

Given data A, suppose we want to compute the nonzero function $f(A)$

Exact Algorithm: The algorithm outputs $f(A)$.

Additive Error: Algorithm outputs F where $|F - f(A)| \leq E$. Here E is called the additive error.

Multiplicative Error: Algorithm outputs F where $|F - f(A)| \leq \epsilon f(A)$. Here $(1 \pm \epsilon)$ is called the approximation factor.

(ϵ, δ) Approximation: The algorithms with probability $1 - \delta$ computes F where $|F - f(A)| \leq \epsilon f(A)$.

Approximate vs Exact: Counting Inversions

Problem: Counting inversions in a permutations $\pi \in S_n$.

The pair (i, j) is called an inversion in permutation π iff

$$i < j \text{ and } \pi(i) > \pi(j)$$

$$\pi = 7,\ 1,\ 2,\ 3,\ 9,\ 5,\ 8,\ 4,\ 6,\ 10$$

$$K(\pi) = \text{number of inversions in } \pi$$

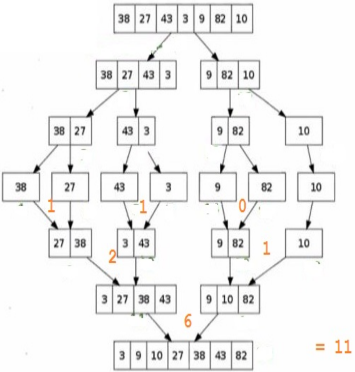
Brute-Force Strategy: Check all pairs.

Running time $= O(n^2)$

Approximate vs Exact: Counting Inversions

Divide and Conquer Strategy:

```
if list L has one element
   return 0 and the list L
divide the list into two halves A and B
(rA, A) ← Sort-and-Count(A)
(rB, B) ← Sort-and-Count(B)
(r , R) ← Merge-and-Count(A, B)
return r = rA + rB + r and the sorted list R
```



Running Time:

$$T(n) = 2T(n/2) + O(n) \quad \Rightarrow \quad T(n) = O(n \log n)$$

Approximate vs Exact: Counting Inversions

An Idea: Compute $K'(\pi) = \sum_i^n |\pi(i) - i|$

Lemma: $K(\pi) \leq K'(\pi) \leq 2K(\pi)$

Running Time: $O(n)$

Streaming Space Usage: $O(\log n)$ bits

Next Lecture

Sublinear Time Algorithms: Some Examples

Two Concentration Lemmas: Markov and Chebyshev bounds