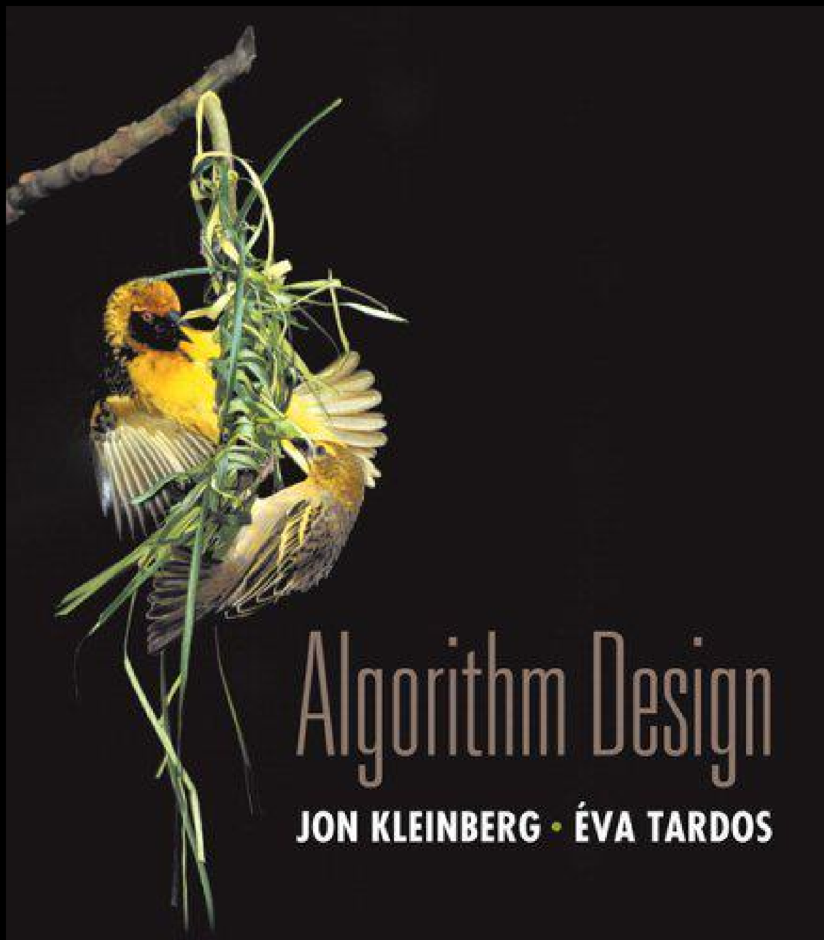


Chapter 2

Basics of Algorithm Analysis

Computational Tractability

``For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing."-- Francis Sullivan, Science, Vol. 287, No. 5454, p. 799, February 2000.

Polynomial-Time

Brute force. For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution.

- Typically takes 2^n time or worse for inputs of size n.
- Unacceptable in practice.

Desirable scaling property. When the input size doubles, the algorithm should only slow down by some constant factor C.

> There exists constants $c > 0$ and $d > 0$ such that on every input of size n, its running time is bounded by $c\,n^d$ steps.

Def. An algorithm is poly-time if the above scaling property holds.

Worst-Case Analysis

Worst case running time. Obtain bound on largest possible running time of algorithm on input of a given size n.

- Generally captures efficiency in practice.
- Hard to find effective alternative.

Def. An algorithm is efficient if its running time is polynomial.

Justification: It really works in practice!

- In practice, the poly-time algorithms that people develop almost always have low constants and low exponents.
- Breaking through the exponential barrier of brute force typically exposes some crucial structure of the problem.

Exceptions.

- Some poly-time algorithms do have high constants and/or exponents, and are useless in practice.
- Some exponential-time algorithms are widely used because the worst-case instances seem to be rare.

Why It Matters

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

2.2 Asymptotic Order of Growth

Asymptotic Order of Growth

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

- $T(n)$ is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.
- $T(n)$ is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$.

Equals sign. $O(f(n))$ is a set of functions, but we often write $T(n) = O(f(n))$ instead of $T(n) \in O(f(n))$.

Properties

Transitivity.

- If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.
- If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$.
- If $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$.

Additivity.

- If $f = O(h)$ and $g = O(h)$ then $f + g = O(h)$.
- If $f = \Omega(h)$ and $g = \Omega(h)$ then $f + g = \Omega(h)$.
- If $f = \Theta(h)$ and $g = \Theta(h)$ then $f + g = \Theta(h)$.

Asymptotic Bounds for Some Common Functions

Polynomials. $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta(n^d)$ if $a_d > 0$.

Polynomial time. Running time is $O(n^d)$ for some constant d independent of the input size n.

Logarithms. $O(\log_a n) = O(\log_b n)$ for any constants $a, b > 1$.

Logarithms. For every $x > 0$, $\log n = O(n^x)$.

log grows slower than every polynomial

Exponentials. For every $r > 1$ and every $d > 0$, $n^d = O(r^n)$.

every exponential grows faster than every polynomial

2.4 A Survey of Common Running Times

Linear Time: $O(n)$

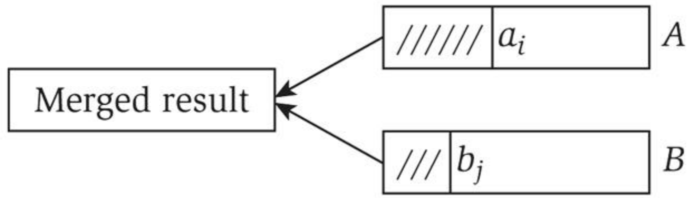
Linear time. Running time is at most a constant factor times the size of the input.

Computing the maximum. Compute maximum of n numbers $a_1, \ldots, a_n$.

```
max ← a_1
for i = 2 to n do
    if a_i > max then
        max ← a_i
    end if
end for
return max.
```

Linear Time: O(n)

Merge. Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole.




```
i = 1, j = 1
while both lists are nonempty do
    if a_i ≤ b_j then
        append a_i to output list and increment i
    else
        append b_j to output list and increment j
    end if
end while
append remainder of nonempty list to output list
return output list.
```

Claim. Merging two lists of size n takes O(n) time.

Pf. After each comparison, the length of output list increases by 1.

O(n log n) Time

O(n log n) time. Arises in divide-and-conquer algorithms.

Sorting. Mergesort and heapsort are sorting algorithms that perform O(n log n) comparisons.

Largest empty interval. Given n time-stamps $x_1, \ldots, x_n$ on which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?

O(n log n) solution. Sort the time-stamps. Scan the sorted list in order, identifying the maximum gap between successive time-stamps.

Quadratic Time: $O(n^2)$

Quadratic time. Enumerate all pairs of elements.

Closest pair of points. Given a list of n points in the plane $(x_1, y_1), \ldots, (x_n, y_n)$, find the pair that is closest.

$O(n^2)$ solution. Try all pairs of points.

```
min ← (x_1 − x_2)^2 + (y_1 − y_2)^2
for i = 1 to n do
    for j = i + 1 to n do
        d ← (x_i − x_j)^2 + (y_i − y_j)^2
        if d < min then
            min ← d
        end if
    end for
end for
```

Remark. $\Omega(n^2)$ seems inevitable, but this is just an illusion.

Cubic Time: $O(n^3)$

Cubic time. Enumerate all triples of elements.

Set disjointness. Given n sets $S_1, \ldots, S_n$ each of which is a subset of 1, 2, …, n, is there some pair of these which are disjoint?

$O(n^3)$ solution. For each pairs of sets, determine if they are disjoint.

```
for each set S_i do
    for each other set S_j do
        for element p of S_i do
            determine whether p also belongs to S_j
            if (no element of S_i belongs to S_j then
                report that S_i and S_j are disjoint
            end if
        end for
    end for
end for
```

Exponential Time

Independent set. Given a graph, what is maximum size of an independent set?

$O(n^2 \, 2^n)$ solution. Enumerate all subsets.

Homework

- Read Chapter 2 of the textbook.

- Exercises 6 & 8 in Chapter 2.