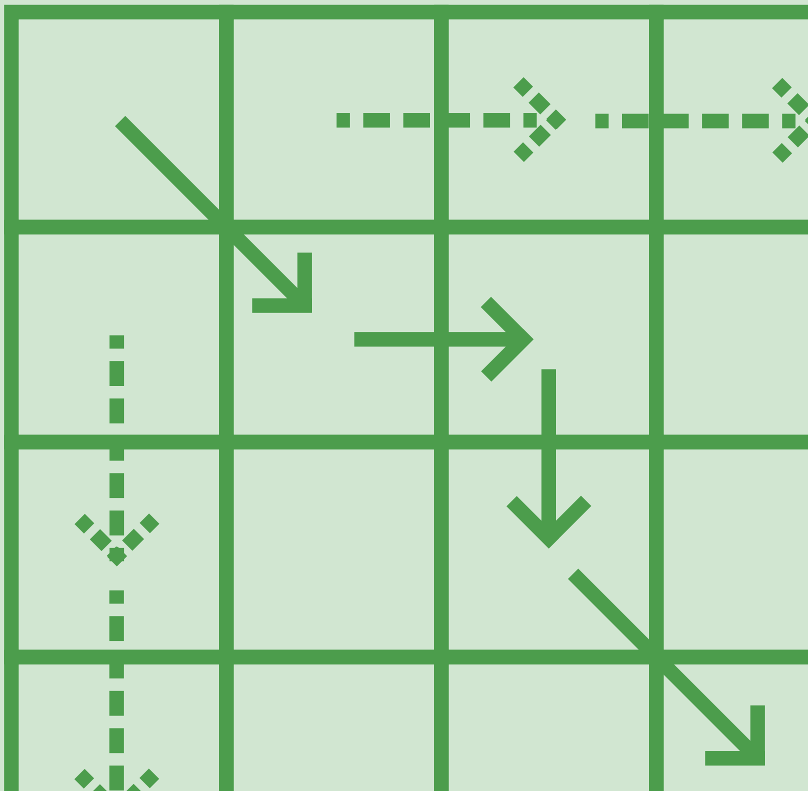
Algorithm Design and Analysis



The cover depicts a dynamic programming matrix used in the edit distance algorithm. Variations of this algorithm such as the Needleman-Wunsch and Smith-Waterman algorithms are frequently used in computational biology to align DNA or protein sequences.

Contents

Divide and Conquer

Chapter 1 Divide and Conquer Primer

1.1 Divide and Conquer Paradigm

Most divide and conquer algorithms follow this paradigm.

- Divide up problem into several subproblems. Note that in some cases, we have to generalize the given problem.
- Recursively solving these subproblems.
- Combining the results from subproblems.

In this chapter and the following chapter, we will examine a few examples of divide-and-conquer algorithms, including but not limited to: maximum subarray, Strassen's matrix multiplication algorithm, quicksort, mergesort, median and selection in sorted array, fast Fourier transform (FTT), inversion counting, etc.

1.2 The Master Theorem

We have used the Master Theorem many times in both the introduction to theory of computation and data structure courses. We will once again present the theorem as it appears in CLRS here. For a careful proof of the theorem, see Section 4.6 of CLRS or Tutorial 7 note for CSC240 (Enriched Introduction to Theory of Computation).

Theorem 1.2.1 — The Master Theorem. Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a function. Let $T(n)$ on the nonnegative integers by the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $\frac{n}{b}$ is interchangeable with $\lfloor n/b \rfloor$ and $\lceil n/b \rceil$. Then, $T(n)$ has the following asymptotic bounds:

- If $f(n) \in O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) \in \Theta(n^{\log_b a})$.
- If $f(n) \in \Theta(n^{\log_b a})$, then $T(n) \in \Theta(n^{\log_b a} \lg n)$.
- If $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) \in \Theta(f(n))$.

An equivalent formulation of the theorem is as follows.

Theorem 1.2.2 — The Master Theorem. Suppose that for $n \in \mathbb{Z}^+$.

$$T(n) = \begin{cases} c & \text{if } n < B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + dn^i & \text{if } n \geq B \end{cases}$$

where $a_1, a_2, B, b \in \mathbb{N}$.

Let $a = a_1 + a_2 \geq 1$, $b > 1$, and $c, d, i \in \mathbb{R} \cup \{0\}$. Then,

$$T(n) \in \begin{cases} O(n^i \log n) & \text{if } a = b^i \\ O(n^i) & \text{if } a < b^i \\ O(n^{\log_b a}) & \text{if } a > b^i \end{cases}$$

1.3 Counting Inversions

The objective of array inversion problems is to find the number inversions in an unsorted array compared to a sorted array. That is, given an array A, how many pairs (i, j) are there such that $i < j$ and $A[i] > A[j]$. In other words, it counts the number of element-wise swaps needed in order to sort the given array.

For example, given the array $[8, 4, 2, 1]$, the algorithm should answer 6 since the array has these six inversions: $(8, 4)$, $(4, 2)$, $(8, 2)$, $(8, 1)$, $(4, 1)$, and $(2, 1)$. If we follow these inversions, we will get a sorted array.

An naive algorithm for this problem is naturally to examine every pair of elements in the given array, which will take $O(n^2)$ comparisons. However, due to its resemblance to sorting, it is not hard to come up with a divide-and-conquer algorithm similar to mergesort that solves this problem in $O(n \log n)$ time.

At a high level, the algorithm should follow the aforementioned paradigm:

- Divide: split list into two halves A and B
- Conquer: recursively count inversions in each list
- Combine: count inversions (a, b) with $a \in A$ and $b \in B$
- Return the sum of the tree counts

For the combine step, we assume that the two subarrays A and B are sorted. Then, we can scan A and B from left to right in parallel and compare $A[i]$ and $B[j]$. If $A[i] < B[j]$, then $A[i]$ is not inverted with any element in B. If $A[i] > B[j]$ then $B[j]$ is inverted with every element in left in A.

A pseudocode for the algorithm is shown below. Equivalently, instead of explicitly splitting the array, we can perform this in place by passing around the indices p and q, as shown in Section 2.3.1 of CLRS.

```
SORT-AND-COUNT($L$)
    if $L.length == 0$
        return $(0, L)$
    $mid = \lfloor (1 + L.length)/2 \rfloor$
    $(count\text{-}a, A) =$ SORT-AND-COUNT($A[1 \dots mid]$)
    $(count\text{-}b, B) =$ SORT-AND-COUNT($A[mid+1 \dots A.length]$)
    $(count\text{-}ab, L') =$ MERGE-AND-COUNT($A, B$)
    return $(count\text{-}a + count\text{-}b + count\text{-}ab,\ L')$
```

```
MERGE-AND-COUNT($A, B$)
    $count = 0$
    $i, j, k = 1$
    $L = []$
    while $i \le A.length$ and $j \le B.length$
        if $A[i] \le B[j]$
            $L[k] = A[i]$
            $i = i + 1$
        else
            $count = count + 1$
            $L[k] = B[j]$
            $j = j + 1$
        $k = k + 1$
    while $i \le A.length$
        $L[k] = A[i]$
        $k = k + 1$
    while $j \le B.length$
        $L[k] = B[j]$
        $k = k + 1$
    return $(count, L)$
```

The number of comparisons made by the algorithm is given by this recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{otherwise} \end{cases}$$

By Masters Theorem, $T(n) \in O(n \log n)$.

1.4 Closest Pair

Lemma 1.4.1 Let p be a point in the 2δ strip within δ distance horizontally. There are at most 7 points p such that $|y_p - y_q| \le \delta$.

Proof. p must lie either in the left or right $\delta \times \delta$ square. Within each square, each point have distance at least k from each other. So, we can pack at most 4 such points into one square, and since we have a left square and right square, we have 8 points in total. Other than p, there are at most 7 points. ■

```
CLOSEST-PAIR($P = p_1, p_2, \ldots, p_n$)
 1  compute a vertical line $L$ such that half the points
    are on each side                                        // $O(n \log n)$
    // consider sorting based on $x$-axis
 2  $\delta_1$ = CLOSEST-PAIR($P_L$)
 3  $\delta_2$ = CLOSEST-PAIR($P_R$)
 4  $\delta = \min\{\delta_1, \delta_2\}$
 5  for $p$ in $p_1, p_2, \ldots, p_n$                      // $O(n)$
 6      if Y-DISTANCE($p, L$)
 7          delete $p$
 8  sort remaining points by $y$-coordinate                 // $O(n \log n)$
 9  for $p$ in $p_1, p_2, \ldots, p_n$                      // $O(n)$
10      for $i$ from 1 to 7
11          $p_i$ = $i$th neighbor of $p$
12          if DISTANCE($p, p_i$) < $\delta$
13              $\delta$ = DISTANCE($p, p_i$)
14  return $\delta$
```

The number of operations performed by this algorithm is

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n \log n) & \text{otherwise} \end{cases}$$

By Masters Theorem, $T(n) \in O(n \log^2 n)$.

Theorem 1.4.2 — Lower Bound for Closest Pair. In a quadradic decision tree model, the closest pair problem requires at least $\Omega(n \log n)$ quadratic tests.

A quadradic decision tree is a comparison tree whose internal nodes are labeled with quadradic comparisons in the form of $x_i < x_j$ or $(x_i - x_k)^2 - (x_j - x_k)^2 < 0$. More generally, each internal node contains a comparison between a polynomial of degree at most 2 and 0 (2nd-order algebraic decision tree).

A hand-wavy proof of this lower bound follows by reduction from the result that the element uniqueness problem has a $\Omega(n \log n)$ lower bound, first shown by Ben-Or in *Lower Bounds For Algebraic Computation Trees*. Element distinctness reduces to 2D closest pair problem.

1.5 Karatsuba's Integer Multiplication Algorithm

To multiply two n-bit integers a and b, we can follow this recursive procedure:

- add two $\frac{1}{2}n$ bit integers
- multiply three $\frac{1}{2}n$-bit integers recursively
- add, subtract, and shift to obtain the result

$$\begin{aligned} a &= 2^{n/2}a_1 + a_0 \\ b &= 2^{n/2}b_1 + b_0 \\ ab &= 2^n a_1 b_1 + 2^{n/2}(a_1 b_0 + a_0 b_1) + a_0 b_0 \\ &= 2^n \underbrace{a_1 b_1} + 2^{n/2}(\underbrace{(a_1 + a_0)(b_1 + b_0)} - \underbrace{a_1 b_1} - \underbrace{a_0 b_0}) + \underbrace{a_0 b_0} \end{aligned}$$

The recursive steps are labeled with underbrace. By Masters Theorem, Karatsuba's algorithm performs

$$T(n) = 3T(\lceil n/2 \rceil) + cn + d \in \Theta(n^{\log_2 3})$$

bit-wise operations.

Chapter 2 Strassen's Algorithm for Matrix Multiplication

2.1 Matrix Multiplication

The standard method to multiply two $n \times n$ matrices requires $O(n^3)$ scalar operations (multiplication and addition). The standard algorithm follows from the definition of matrix multiplication that the i, j entry of $C = A \cdot B$ is given by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

```
SQUARE-MATRIX-MULTIPLY(A, B)
1  $n = A.rows$
2  $C = n \times n$ matrix
3  for $i = 1$ to $n$
4      for $j = 1$ to $n$
5          $c_{ij} = 0$
6          for $k = 1$ to $n$
7              $c_{ij} = c_{ij} + a_{ij} b_{ij}$
8  return $C$
```

Clearly, this algorithm runs in $\Theta(n^3)$.

2.2 A Recursive Approach

Without loss of generality, let $n = 2^k$ for some k. Then, we can divide any $n \times n$ matrix into four $n/2 \times n/2$ matrices.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Since matrices are a non-commutative ring, $n \times n$ matrix multiplication can be realized in $7 n/2 \times n/2$ matrix multiplications and 18 matrix additions. Matrix addition only takes $O(n^2)$ scalar operations. Then, we have the following recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 7T(n/2) + O(n^2) & \text{otherwise} \end{cases}$$

which implies that $T(n) \in O(n^{\log_2 7})$.

An implementation of this recursive approach is presented in pseudocode below

```
SQUARE-MATRIX-MULTIPLY($A, B$)
 1  $n = A.rows$
 2  $C = n \times n$ matrix
 3  if $n == 1$
 4  else
 5      partition $A$, $B$, and $C$ each into four sub-matrices
 6      $C_{11}$ = SQUARE-MATRIX-MULTIPLY($A_{11}, B_{11}$)+
               SQUARE-MATRIX-MULTIPLY($A_{12}, B_{21}$)
 7      $C_{12}$ = SQUARE-MATRIX-MULTIPLY($A_{11}, B_{12}$)+
               SQUARE-MATRIX-MULTIPLY($A_{12}, B_{22}$)
 8      $C_{21}$ = SQUARE-MATRIX-MULTIPLY($A_{21}, B_{11}$)+
               SQUARE-MATRIX-MULTIPLY($A_{22}, B_{21}$)
 9      $C_{22}$ = SQUARE-MATRIX-MULTIPLY($A_{21}, B_{12}$)+
               SQUARE-MATRIX-MULTIPLY($A_{22}, B_{22}$)
10  return $C$
```

However, this particular implementation of the recursive approach still has a $O(n^3)$ running time, as demonstrated by solving the recurrence by Masters Theorem.

2.3 Strassen's Algorithm

Strassen's algorithm is a rather peculiar algorithm that multiplies two square matrices using $O(n^{\log_2 7}) = O(n^{2.81})$ scalar operations. It is debatable how practical asymptotically faster matrix multiplication algorithms (including Strassen's algorithm and Coppersmith-Winograd algorithm) are given that their crossover point (the threshold on input size beyond which such algorithms become actually faster) is often quite large for them to be useful in practical applications.

Strassen's algorithm as described in CLRS works as follows:

1. Divide the input matrices A and B and output matrix C into $n/2 \times n/2$ submatrices. This takes $\Theta(1)$ operations.
2. Create 10 submatrices $S_1, S_2, \dots, S_{10}$, each of which is $n/2 \times n/2$ and is the sum or difference of two matrices created in Step 1. This can be done in $\Theta(n^2)$.
3. Using the submatrices in Step 1 and the 10 matrices in Step 2, recursively compute seven matrix products $P_1, P_2, \dots, P_7$. Each matrix P_i is $n/2 \times n/2$.
4. Compute the desired $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C by adding and subtracting various combinations of the P_i matrices. This can also be done in $\Theta(n^2)$ time.

More specifically, in Step 2,

$$\begin{aligned}
S_1 &= B_{12} - B_{22}\\
S_2 &= A_{11} + A_{12}\\
S_3 &= A_{21} + A_{22}\\
S_4 &= B_{21} - B_{11}\\
S_5 &= A_{11} + A_{22}\\
S_6 &= B_{11} + B_{22}\\
S_7 &= A_{12} - A_{22}\\
S_8 &= B_{21} + B_{22}\\
S_9 &= A_{11} - B_{21}\\
S_{10} &= B_{11} + B_{12},
\end{aligned}$$

and in Step 3,

$$\begin{aligned}
P_1 &= A_{11} \cdot S_1 && = A_{11} \cdot B_{12} - A_{11} \cdot B_{22}\\
P_2 &= S_2 \cdot B_{22} && = A_{11} \cdot B_{22} + A_{12} \cdot B_{22}\\
P_3 &= S_3 \cdot B_{11} && = A_{21} \cdot B_{11} + A_{22} \cdot B_{11}\\
P_4 &= A_{22} \cdot S_4 && = A_{22} \cdot B_{21} - A_{22} \cdot B_{11}\\
P_5 &= S_5 \cdot S_6 && = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}\\
P_6 &= S_7 \cdot S_8 && = A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}\\
P_7 &= S_9 \cdot S_{10} && = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}.
\end{aligned}$$

In Step 4,

$$\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6\\
C_{12} &= P_1 + P_2\\
C_{21} &= P_3 + P_4\\
C_{22} &= P_5 + P_1 - P_3 - P_7
\end{aligned}$$

Chapter 3 Fast Fourier Transform

II

Greedy Algorithms

III Dynamic Programming

IV Network Flow

V

Complexity

VI Linear Programming

VII Approximation Algorithms

VIII Randomized Algorithms

Appendix

Commonly Used Axioms & Theorems

Rules of Inference

Axiom 1 — Modus Ponens. $(P \land (P \text{ IMPLIES } Q)) \text{ IMPLIES } Q$

Axiom 2 — Modus Tollens. $(\neg Q \land (P \text{ IMPLIES } Q)) \text{ IMPLIES } \neg P$

Axiom 3 — Hypothetical Syllogism (transitivity).

$((P \text{ IMPLIES } Q) \land (Q \text{ IMPLIES } R)) \text{ IMPLIES } (P \text{ IMPLIES } R)$

Axiom 4 — Disjunctive Syllogism. $((P \lor Q) \land \neg P) \text{ IMPLIES } Q$

Axiom 5 — Addition. $P \text{ IMPLIES } (P \lor Q)$

Axiom 6 — Simplification. $(P \land Q) \text{ IMPLIES } P$

Axiom 7 — Conjunction. $((P) \land (Q)) \text{ IMPLIES } (P \land Q)$

Axiom 8 — Resolution. $((P \lor Q) \land (\neg P \lor R))$ IMPLIES $(Q \lor R)$

Laws of Logic

Axiom 9 — Implication Law. $(P \text{ IMPLIES } Q) \equiv (\neg P \lor Q)$

Axiom 10 — Distributive Law.

$$(P \land (Q \lor R)) \equiv ((P \land Q) \lor (P \land R))$$
$$(P \lor (Q \land R)) \equiv ((P \lor Q) \land (P \lor R))$$

Axiom 11 — De Morgan's Law.

$$\neg(P \land Q) \equiv (\neg P \lor \neg Q)$$
$$\neg(P \lor Q) \equiv (\neg P \land \neg Q)$$

Axiom 12 — Absorption Law.

$$(P \lor (P \land Q)) \equiv P$$
$$(P \land (P \lor Q)) \equiv P$$

Axiom 13 — Commutativity of AND. $A \land B \equiv B \land A$

Axiom 14 — Associativity of AND. $(A \land B) \land C \equiv A \land (B \land C)$

Axiom 15 — Identity of AND. $\mathbf{T} \land A \equiv A$

Axiom 16 — Zero of AND. $\mathbf{F} \land A \equiv \mathbf{F}$

Axiom 17 — Idempotence for AND. $A \land A \equiv A$

Axiom 18 — Contradiction for AND. $A \land \neg A \equiv \mathbf{F}$

Axiom 19 — Double Negation. $\neg(\neg A) \equiv A$

Axiom 20 — Validity for OR. $A \vee \neg A \equiv \mathbf{T}$

Induction

Axiom 21 — Well Ordering Principle. Every nonempty set of nonnegative integers has a smallest element. i.e., For any $A \subset \mathbb{N}$ such that $A \neq \emptyset$, there is some $a \in A$ such that $\forall a' \in A. a \leq a'$.

Recurrences

Theorem 22 — The Master Theorem. Suppose that for $n \in \mathbb{Z}^+$.

$$T(n) = \begin{cases} c & \text{if } n < B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + dn^i & \text{if } n \geq B \end{cases}$$

where $a_1, a_2, B, b \in \mathbb{N}$.

Let $a = a_1 + a_2 \geq 1$, $b > 1$, and $c, d, i \in \mathbb{R} \cup \{0\}$. Then,

$$T(n) \in \begin{cases} O(n^i \log n) & \text{if } a = b^i \\ O(n^i) & \text{if } a < b^i \\ O(n^{\log_b a}) & \text{if } a > b^i \end{cases}$$

Linear Recurrences:

A linear recurrence is an equation

$$f(n) = \underbrace{a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)}_{\text{homogeneous part}} + \underbrace{g(n)}_{\text{inhomogeneous part}}$$

along with some boundary conditions.

The procedure for solving linear recurrences are as follows:

1. Find the roots of the characteristic equation. Linear recurrences usually have exponential solutions (such as x^n). Such solution is called the **homogeneous solution**.

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{k-1} x + a_k$$

2. Write down the homogeneous solution. Each root generates one term and the homogeneous solution is their sum. A non-repeated root r generates the term cr^n, where c is a constant to be determined later. A root with r with multiplicity k generates the terms

$$d_1 r^n \quad d_2 n r^n \quad d_3 n^2 r^n \quad \cdots \quad d_k n^{k-1} r^n$$

where $d_1, \cdots, d_k$ are constants to be determined later.

3. Find a **particular solution** for the full recurrence including the inhomogeneous part, but without considering the boundary conditions.
   If $g(n)$ is a constant or a polynomial, try a polynomial of the same degree, then of one higher degree, then two higher. If $g(n)$ is exponential in the form $g(n) = k^n$, then try $f(n) = ck^n$, then $f(n) = (bn+c)k^n$, then $f(n) = (an^2+bn+c)k^n$, etc.
4. Write the **general solution**, which is the sum of homogeneous solution and particular solution.
5. Substitute the boundary condition into the general solution. Each boundary condition gives a linear equation. Solve such system of linear equations for the values of the constants to make the solution consistent with the boundary conditions.

Basic Prerequisite Mathematics

Basic Prerequisite Mathematics

SET THEORY

Common Sets

- $\mathbb{N} = \{0, 1, 2, \ldots\}$: the natural numbers, or non-negative integers. The convention in computer science is to include 0 in the natural numbers.
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$: the integers
- $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$: the positive integers
- $\mathbb{Z}^- = \{-1, -2, -3, \ldots\}$: the negative integers
- $\mathbb{Q}$ the rational numbers, $\mathbb{Q}^+$ the positive rationals, and $\mathbb{Q}^-$ the negative rationals.
- $\mathbb{R}$ the real numbers, $\mathbb{R}^+$ the positive reals, and $\mathbb{R}^-$ the negative reals.

Notation

For any sets A and B, we will use the following standard notation.

- $x \in A$: "x is an element of A" or "A contains x"
- $A \subseteq B$: "A is a subset of B" or "A is included in B"
- $A = B$: "A equals B" (Note that $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.)
- $A \subsetneq B$: "A is a proper subset of B"
  (Note that $A \subsetneq B$ if and only if $A \subseteq B$ and $A \neq B$.)
- $A \cup B$: "A union B"
- $A \cap B$: "A intersection B"
- $A - B$: "A minus B" (*set* difference)
- $|A|$: "cardinality of A" (the number of elements of A)
- $\varnothing$ or $\{\}$: "the empty set"
- $\mathcal{P}(A)$ or 2^A: "powerset of A" (the set of all subsets of A)
  If $A = \{a, 34, \triangle\}$, then $\mathcal{P}(A) = \big\{\{\}, \{a\}, \{34\}, \{\triangle\}, \{a, 34\}, \{a, \triangle\}, \{34, \triangle\}, \{a, 34, \triangle\}\big\}$.
  $S \in \mathcal{P}(A)$ means the same as $S \subseteq A$.
- $\{x \in A \mid P(x)\}$: "the set of elements x in A for which $P(x)$ is true"
  For example, $\{x \in \mathbb{Z} \mid \cos(\pi x) > 0\}$ represents the set of integers x for which $\cos(\pi x)$ is greater than zero, *i.e.*, it is equal to $\{\ldots, -4, -2, 0, 2, 4, \ldots\} = \{x \in \mathbb{Z} \mid x \text{ is even}\}$.

- $A \times B$: "the cross product or Cartesian product of A and B"
  $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$.
  If $A = \{1, 2, 3\}$ and $B = \{5, 6\}$, then $A \times B = \{(1, 5), (1, 6), (2, 5), (2, 6), (3, 5), (3, 6)\}$.)

- A^n: "the cross product of n copies of A"
  This is set of all sequences of $n \geq 1$ elements, each of which is in A.

- B^A or $A \to B$: "the set of all functions from A to B."

- $f : A \to B$ or $f \in B^A$: "f is a function from A to B"
  f associates one element $f(x) \in B$ to every element $x \in A$.

NUMBER THEORY

For any two natural numbers a and b, we say that a *divides* b if there exists a natural number c such that $b = ac$. In such a case, we say that a is a *divisor* of b (*e.g.*, 3 is a divisor of 12 but 3 is not a divisor of 16). Note that any natural number is a divisor of 0 and 1 is a divisor of any natural number. A number a is *even* if 2 divides a and is *odd* if 2 does not divide a.

A natural number p is *prime* if it has exactly two positive divisors (*e.g.*, 2 is prime since its positive divisors are 1 and 2 but 1 is **not** prime since it only has one positive divisor: 1). There are an infinite number of prime numbers and any integer greater than one can be expressed in a unique way as a finite product of prime numbers (*e.g.*, $8 = 2^3$, $77 = 7 \times 11$, $3 = 3$).

Inequalities

For any integers m and n, $m < n$ if and only if $m + 1 \leq n$ and $m > n$ if and only if $m \geq n + 1$. For any real numbers w, x, y, and z, the following properties always hold (they also hold when $<$ and $\leq$ are exchanged throughout with $>$ and $\geq$, respectively).

- if $x < y$ and $w \leq z$, then $x + w < y + z$

- if $x < y$, then $\begin{cases} xz < yz & \text{if } z > 0 \\ xz = yz & \text{if } z = 0 \\ xz > yz & \text{if } z < 0 \end{cases}$

- if $x \leq y$ and $y < z$ (or if $x < y$ and $y \leq z$), then $x < z$

Functions

Here are some common number-theoretic functions together with their definitions and properties of them. (Unless noted otherwise, in this section, x and y represent arbitrary real numbers and k, m, and n represent arbitrary positive integers.)

- $\min\{x, y\}$: "minimum of x and y" (the smallest of x or y)

  Properties: $\min\{x, y\} \leq x$
  $\min\{x, y\} \leq y$

- $\max\{x, y\}$: "maximum of x and y" (the largest of x or y)

  Properties: $x \le \max\{x, y\}$
  $y \le \max\{x, y\}$

- $\lfloor x \rfloor$: "floor of x" (the greatest integer less than or equal to x, *e.g.*, $\lfloor 5.67 \rfloor = 5$, $\lfloor -2.01 \rfloor = -3$)

  Properties: $x - 1 < \lfloor x \rfloor \le x$
  $\lfloor -x \rfloor = -\lceil x \rceil$
  $\lfloor x + k \rfloor = \lfloor x \rfloor + k$
  $\lfloor \lfloor k/m \rfloor / n \rfloor = \lfloor k/mn \rfloor$
  $(k - m + 1)/m \le \lfloor k/m \rfloor$

- $\lceil x \rceil$: "ceiling of x" (the least integer greater than or equal to x, *e.g.*, $\lceil 5.67 \rceil = 6$, $\lceil -2.01 \rceil = -2$)

  Properties: $x \le \lceil x \rceil < x + 1$
  $\lceil -x \rceil = -\lfloor x \rfloor$
  $\lceil x + k \rceil = \lceil x \rceil + k$
  $\lceil \lceil k/m \rceil / n \rceil = \lceil k/mn \rceil$
  $\lceil k/m \rceil \le (k + m - 1)/m$

  Additional property of $\lfloor\ \rfloor$ and $\lceil\ \rceil$: $\lfloor k/2 \rfloor + \lceil k/2 \rceil = k$.

- $|x|$: "absolute value of x" ($|x| = x$ if $x \ge 0$; $-x$ if $x < 0$, *e.g.*, $|5.67| = 5.67$, $|-2.01| = 2.01$)

  Beware! The same notation is used to represent the cardinality $|A|$ of a set A and the absolute value $|x|$ of a number x so be sure you are aware of the context in which it is used.

- $m \operatorname{div} n$: "the quotient of m divided by n" (integer division of m by n, *e.g.*, $5 \operatorname{div} 6 = 0$, $27 \operatorname{div} 4 = 6$, $-27 \operatorname{div} 4 = -6$)

  Properties: If $m, n > 0$, then $m \operatorname{div} n = \lfloor m/n \rfloor$
  $(-m) \operatorname{div} n = -(m \operatorname{div} n) = m \operatorname{div} (-n)$

- $m \operatorname{rem} n$: "the remainder of m divided by n" (*e.g.*, $5 \operatorname{rem} 6 = 5$, $27 \operatorname{rem} 4 = 3$, $-27 \operatorname{rem} 4 = -3$)

  Properties: $m = (m \operatorname{div} n) \cdot n + m \operatorname{rem} n$
  $(-m) \operatorname{rem} n = -(m \operatorname{rem} n) = m \operatorname{rem} (-n)$

- $m \bmod n$: "m modulo n" (*e.g.*, $5 \bmod 6 = 5$, $27 \bmod 4 = 3$ $-27 \bmod 4 = 1$)

  Properties: $0 \le m \bmod n < n$
  n divides $m - (m \bmod n)$.

- $\gcd(m, n)$: "greatest common divisor of m and n" (the largest positive integer that divides both m and n)

  For example, $\gcd(3, 4) = 1$, $\gcd(12, 20) = 4$, $\gcd(3, 6) = 3$

- $\operatorname{lcm}(m, n)$: "least common multiple of m and n" (the smallest positive integer that m and n both divide)

  For example, $\operatorname{lcm}(3, 4) = 12$, $\operatorname{lcm}(12, 20) = 60$, $\operatorname{lcm}(3, 6) = 6$

  Properties: $\gcd(m, n) \cdot \operatorname{lcm}(m, n) = m \cdot n$.

CALCULUS

Limits and Sums

An infinite sequence of real numbers $\{a_n\} = a_1, a_2, \ldots, a_n, \ldots$ *converges* to a limit $L \in \mathbb{R}$ if, for every $\varepsilon > 0$, there exists $n_0 \geq 0$ such that $|a_n - L| < \varepsilon$ for every $n \geq n_0$. In this case, we write $\lim_{n\to\infty} a_n = L$ Otherwise, we say that the sequence *diverges*.

If $\{a_n\}$ and $\{b_n\}$ are two sequences of real numbers such that $\lim_{n\to\infty} a_n = L_1$ and $\lim_{n\to\infty} b_n = L_2$, then

$$\lim_{n\to\infty}(a_n + b_n) = L_1 + L_2 \quad \text{and} \quad \lim_{n\to\infty}(a_n \cdot b_n) = L_1 \cdot L_2.$$

In particular, if c is any real number, then

$$\lim_{n\to\infty}(c \cdot a_n) = c \cdot L_1.$$

The sum $a_1 + a_2 + \cdots + a_n$ and product $a_1 \cdot a_2 \cdots a_n$ of the finite sequence $a_1, a_2, \ldots, a_n$ are denoted by

$$\sum_{i=1}^{n} a_i \text{ and } \prod_{i=1}^{n} a_i.$$

If the elements of the sequence are all different and $S = \{a_1, a_2, \ldots, a_n\}$ is the set of elements in the sequence, these can also be denoted by

$$\sum_{a\in S} a \text{ and } \prod_{a\in S} a.$$

Examples:

- For any $a \in \mathbb{R}$ such that $-1 < a < 1$, $\lim_{n\to\infty} a^n = 0$.
- For any $a \in \mathbb{R}^+$, $\lim_{n\to\infty} a^{1/n} = 1$.
- For any $a \in \mathbb{R}^+$, $\lim_{n\to\infty} (1/n)^a = 0$.
- $\lim_{n\to\infty} (1 + 1/n)^n = e = 2.71828182845904523536\ldots$
- For any $a, b \in \mathbb{R}$, the *arithmetic* sum is given by:

$$\sum_{i=0}^{n}(a + ib) = (a) + (a + b) + (a + 2b) + \cdots + (a + nb) = \frac{1}{2}(n + 1)(2a + nb).$$

- For any $a, b \in \mathbb{R}^+$, the *geometric* sum is given by:

$$\sum_{i=0}^{n}(ab^i) = a + ab + ab^2 + \cdots + ab^n = \frac{a(1 - b^{n+1})}{1 - b}.$$

EXPONENTS AND LOGARITHMS

Definition: For any $a, b, c \in \mathbb{R}^+$, $a = \log_b c$ if and only if $b^a = c$.

Notation: For any $x \in \mathbb{R}^+$, $\ln x = \log_e x$ and $\lg x = \log_2 x$.

For any $a, b, c \in \mathbb{R}^+$ and any $n \in \mathbb{Z}^+$, the following properties always hold.

- $\sqrt[n]{b} = b^{1/n}$
- $b^a b^c = b^{a+c}$
- $(b^a)^c = b^{ac}$
- $b^a / b^c = b^{a-c}$
- $b^0 = 1$
- $a^b c^b = (ac)^b$
- $b^{\log_b a} = a = \log_b b^a$
- $a^{\log_b c} = c^{\log_b a}$
- $\log_b(ac) = \log_b a + \log_b c$
- $\log_b(a^c) = c \cdot \log_b a$
- $\log_b(a/c) = \log_b a - \log_b c$
- $\log_b 1 = 0$
- $\log_b a = \log_c a / \log_c b$

BINARY NOTATION

A *binary number* is a sequence of bits $a_k \cdots a_1 a_0$ where each bit a_i is equal to 0 or 1. Every binary number represents a natural number in the following way:

$$(a_k \cdots a_1 a_0)_2 = \sum_{i=0}^{k} a_i 2^i = a_k 2^k + \cdots + a_1 2 + a_0.$$

For example, $(1001)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 1 = 9$, $(01110)_2 = 8 + 4 + 2 = 14$.

Properties:

- If $a = (a_k \cdots a_1 a_0)_2$, then $2a = (a_k \cdots a_1 a_0 0)_2$, *e.g.*, $9 = (1001)_2$ so $18 = (10010)_2$.
- If $a = (a_k \cdots a_1 a_0)_2$, then $\lfloor a/2 \rfloor = (a_k \cdots a_1)_2$, *e.g.*, $9 = (1001)_2$ so $4 = (100)_2$.
- The smallest number of bits required to represent the positive integer n in binary is called the *length* of n and is equal to $\lceil \log_2(n+1) \rceil$.

Make sure you know how to add and multiply two binary numbers. For example, $(1111)_2 + (101)_2 = (10100)_2$ and $(1111)_2 \times 101)_2 = (1001011)_2$.

Proof Templates

Proof Outlines

LINE NUMBERS: Only lines that are referred to have labels (for example, L1) in this document. For a formal proof, all lines are numbered. Line numbers appear at the beginning of a line. You can indent line numbers together with the lines they are numbering or all line numbers can be unindented, provided you are consistent.

INDENTATION: Indent when you make an assumption or define a variable. Unindent when this assumption or variable is no longer being used.

1. **Implication**: Direct proof of A IMPLIES B.

```
    L1. Assume A.
    ⋮
    L2. B
A IMPLIES B; direct proof: L1, L2
```

2. **Implication**: Indirect proof of A IMPLIES B.

```
    L1. Assume NOT(B).
    ⋮
    L2. NOT(A)
A IMPLIES B; indirect proof: L1, L2
```

3. **Equivalence**: Proof of A IFF B.

```
    L1. Assume A.
    ⋮
    L2. B
L3. A IMPLIES B; direct proof: L1, L2
    L4. Assume B.
    ⋮
    L5. A
L6. B IMPLIES A; direct proof: L4, L5
A IFF B; equivalence: L3, L6
```

4. **Proof by contradiction** of A.

```
    L1. To obtain a contradiction, assume NOT(A).
    ⋮
        L2. B
    ⋮
        L3. NOT(B)
    L4. This is a contradiction: L2, L3
Therefore A; proof by contradiction: L1, L4
```

5. **Modus Ponens**.

   ⋮
   L1. A
   ⋮
   L2. A IMPLIES B
   B; modus ponens: L1, L2

6. **Conjunction:** Proof of A AND B:

   ⋮
   L1. A
   ⋮
   L2. B
   A AND B; proof of conjunction; L1, 2

7. **Use of Conjunction**:

   ⋮
   L1. A AND B
   A; use of conjunction: L1
   B; use of conjunction: L1

8. **Implication with Conjunction**: Proof of (A_1 AND A_2) IMPLIES B.

      L1. Assume A_1 AND A_2.
      A_1; use of conjunction, L1
      A_2; use of conjunction, L1
      ⋮
      L2. B
   (A_1 AND A_2) IMPLIES B; direct proof, L1, L2

9. **Implication with Conjunction**: Proof of A IMPLIES (B_1 AND B_2).

      L1. Assume A.
      ⋮
      L2. B_1
      ⋮
      L3. B_2
      L4. B_1 AND B_2; proof of conjunction: L2, L3
   A IMPLIES (B_1 AND B_2); direct proof: L1, L4

10. **Disjunction**: Proof of A OR B and B OR A.

    ⋮
    L1. A
    A OR B; proof of disjunction: L1
    B OR A; proof of disjunction: L1

11. **Proof by cases**.

    L1. C OR $NOT(C)$ tautology
    L2. Case 1: Assume C.
    $\vdots$
    L3. A
    L4. C IMPLIES A; direct proof: L2, L3
    L5. Case 2: Assume $NOT(C)$.
    $\vdots$
    L6. A
    L7. $NOT(C)$ IMPLIES A; direct proof: L5, L6
    A proof by cases: L1, L4, L7

12. **Proof by cases** of A OR B.

    L1. C OR $NOT(C)$ tautology
    L2. Case 1: Assume C.
    $\vdots$
    L3. A
    L4. A OR B; proof of disjunction, L3
    L5. C IMPLIES (A OR B); direct proof, L2, L4
    L6. Case 2: Assume $NOT(C)$.
    $\vdots$
    L7. B
    L8. A OR B; proof of disjunction, L7
    L9. $NOT(C)$ IMPLIES (A OR B); direct proof: L6, L8
    A OR B; proof by cases: L1, L5, L9

13. **Implication with Disjunction**: Proof by cases of (A_1 OR A_2) IMPLIES B.

    L1. Case 1: Assume A_1.
    $\vdots$
    L2. B
    L3. A_1 IMPLIES B; direct proof: L1,L2
    L4. Case 2: Assume A_2.
    $\vdots$
    L5. B
    L6. A_2 IMPLIES B; direct proof: L4, L5
    (A_1 OR A_2) IMPLIES B; proof by cases: L3, L6

14. **Implication with Disjunction**: Proof by cases of A IMPLIES (B_1 OR B_2).

```
    L1. Assume $A$.
    L2. $C$ OR NOT($C$) tautology
    L3. Case 1: Assume $C$.
                ⋮
                L4. $B_1$
                L5. $B_1$ OR $B_2$; disjunction: L4
    L6. $C$ IMPLIES ($B_1$ OR $B_2$); direct proof: L3, L5
    L7. Case 2: AssumeNOT($C$).
                ⋮
                L8. $B_2$
                L9. $B_1$ OR $B_2$; disjunction: L8
    L10. NOT($C$) IMPLIES ($B_1$ OR $B_2$); direct proof: L7, L9
    L11. $B_1$ OR $B_2$; proof by cases: L2, L6, L10
$A$ IMPLIES ($B_1$ OR $B_2$): direct proof. L1, L11
```

15. **Substitution of a Variable in a Tautology**:
Suppose P is a propositional variable, Q is a formula, and R' is obtained from R by replacing *every* occurrence of P by (Q).

L1. R tautology
R'; substitution of all P by Q: L1

16. **Substitution of a Formula by a Logically Equivalent Formula**:
Suppose S is a subformula of R and R' is obtained from R by replacing *some* occurrence of S by S'.

L1. R
L2. S IFF S'
L3. R'; substitution of an occurrence of S by S': L1, L2

17. **Specialization**:

L1. $c \in D$
L2. $\forall x \in D.P(x)$
$P(c)$; specialization: L1, L2

18. **Generalization**: Proof of $\forall x \in D.P(x)$.

```
    L1. Let $x$ be an arbitrary element of $D$.
    ⋮
    L2. $P(x)$
Since $x$ is an arbitrary element of $D$,
$\forall x \in D.P(x)$; generalization: L1, L2
```

19. **Universal Quantification with Implication**: Proof of $\forall x \in D.(P(x)$ IMPLIES $Q(x))$.

```
    L1. Let $x$ be an arbitrary element of $D$.
        L2. Assume $P(x)$
        ⋮
        L3. $Q(x)$
    L4. $P(x)$ IMPLIES $Q(x)$; direct proof: L2, L3
Since $x$ is an arbitrary element of $D$,
$\forall x \in D.(P(x)$ IMPLIES $Q(x))$; generalization: L1, L4
```

20. **Implication with Universal Quantification**: Proof of $(\forall x \in D.P(x))$ IMPLIES A.

```
    L1. Assume $\forall x \in D.P(x)$.
    ⋮
    L2. $a \in D$
    $P(a)$; specialization: L1, L2
    ⋮
    L3. $A$
Therefore $(\forall x \in D.P(x))$ IMPLIES $A$; direct proof: L1, L3
```

21. **Implication with Universal Quantification**: Proof of A IMPLIES $(\forall x \in D.P(x))$.

```
    L1. Assume $A$.
        L2. Let $x$ be an arbitrary element of $D$.
        ⋮
        L3. $P(x)$
    Since $x$ is an arbitrary element of $D$,
    L4. $\forall x \in D.P(x)$; generalization, L2, L3
$A$ IMPLIES $(\forall x \in D.P(x))$; direct proof: L1, L4
```

22. **Instantiation**:

```
L1. $\exists x \in D.P(x)$
    Let $c \in D$ be such that $P(c)$; instantiation: L1
    ⋮
```

23. **Construction**: Proof of $\exists x \in D.P(x)$.

```
    L1. Let $a = \cdots$
    ⋮
    L2. $a \in D$
    ⋮
    L3. $P(a)$
$\exists x \in D.P(x)$; construction: L1, L2, L3
```

24. **Existential Quantification with Implication**: Proof of $\exists x \in D.(P(x)$ IMPLIES $Q(x))$.

```
    L1. Let $a = \cdots$
    ⋮
    L2. $a \in D$
        L3. Suppose $P(a)$.
        ⋮
        L4. $Q(a)$
    L5. $P(a)$ IMPLIES $Q(a)$; direct proof: L3, L4
$\exists x \in D.(P(x)$ IMPLIES $Q(x))$; construction: L1, L2, L5
```

25. **Implication with Existential Quantification**: Proof of $(\exists x \in D.P(x))$ IMPLIES A.

```
    L1. Assume $\exists x \in D.P(x)$.
        Let $a \in D$ be such that $P(a)$; instantiation: L1
        ⋮
        L2. $A$
$(\exists x \in D.P(x))$ IMPLIES $A$; direct proof: L1, L2
```

26. **Implication with Existential Quantification**: Proof of A IMPLIES $(\exists x \in D.P(x))$.

```
    L1. Assume $A$.
        L2. Let $a = \cdots$
        ⋮
        L3. $a \in D$
        ⋮
        L4. $P(a)$
    L5. $\exists x \in D.P(x)$; construction: L2, L3, L4
$A$ IMPLIES $(\exists x \in D.P(x))$; direct proof: L1, L5
```

27. **Subset**: Proof of $A \subseteq B$.

```
    L1. Let $x \in A$ be arbitrary.
    ⋮
    L2. $x \in B$
    The following line is optional:
    L3. $x \in A$ IMPLIES $x \in B$; direct proof: L1, L2
$A \subseteq B$; definition of subset: L3 (or L1, L2, if the optional line is missing)
```

28. **Weak Induction**: Proof of $\forall n \in N.P(n)$

    Base Case:

    $\vdots$

    L1. $P(0)$

        L2. Let $n \in N$ be arbitrary.

            L3. Assume $P(n)$.

            $\vdots$

            L4. $P(n+1)$

        The following two lines are optional:

        L5. $P(n)$ IMPLIES $(P(n+1)$; direct proof of implication: L3, L4

    L6. $\forall n \in N.(P(n)$ IMPLIES $P(n+1))$; generalization L2, L5

    $\forall n \in N.P(n)$ induction; L1, L6 (or L1, L2, L3, L4, if the optional lines are missing)

29. **Strong Induction**: Proof of $\forall n \in N.P(n)$

        L1. Let $n \in N$ be arbitrary.

            L2. Assume $\forall j \in N.(j < n$ IMPLIES $P(j))$

            $\vdots$

            L3. $P(n)$

        The following two lines are optional:

        L4. $\forall j \in N.(j < n$ IMPLIES $P(j))$ IMPLIES $P(n)$; direct proof of implication: L2, L3

    L5. $\forall n \in N.[\forall j \in N.(j < n$ IMPLIES $P(j))$ IMPLIES $P(n)]$; generalization: L1, L4

    $\forall n \in N.P(n)$; strong induction: L5 (or L1, L2, L3, if the optional lines are missing)

30. **Structural Induction**: Proof of $\forall e \in S.P(e)$, where S is a recursively defined set

    Base case(s):

        L1. For each base case e in the definition of S

        L2. $P(e)$.

    Constructor case(s):

        L3. For each constructor case e of the definition of S,

            L4. assume $P(e')$ for all components e' of e.

            $\vdots$

            L5. $P(e)$

    $\forall e \in S.P(e)$; structural induction: L1, L2, L3, L4, L5

31. **Well Ordering Principle**: Proof of $\forall e \in S.P(e)$, where S is a well ordered set, i.e. every nonempty subset of S has a smallest element.

```
    L1. To obtain a contradiction, suppose that $\forall e \in S.P(e)$ is false.
    L2. Let $C = \{e \in S \mid P(e) \text{ is false}\}$ be the set of counterexamples to $P$.
    L3. $C \neq \phi$; definition: L1, L2
        L4. Let $e$ be the smallest element of $C$; well ordering principle: L2, L3
            Let $e' = \cdots$
            $\vdots$
            L5. $e' \in C$
            $\vdots$
            L6. $e' < e$.
    L7. This is a contradiction: L4, L5, L6
$\forall e \in S.P(e)$; proof by contradiction: L1, L7
```

Index

Bibliography

Courses

[2] Erik Demaine and Srini Devadas. *6.006 Introduction to Algorithms*. Massachusetts Institute of Technology. Fall 2011.

[3] Erik Demaine, Srini Devadas, and Nancy Lynch. *6.046J Design and Analysis of Algorithms*. Massachusetts Institute of Technology. Spring 2015.

[4] Faith Ellen. *CSC240S1 Winter 2021*. University of Toronto. 2021.

[5] Faith Ellen. *CSC265F1 Fall 2021*. University of Toronto. 2021.

[10] Mauricio Karchmer, Anand Natarajan, and Nir Shavit. *6.006 Introduction to Algorithms*. Massachusetts Institute of Technology. Spring 2021.

Books

[1] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844.

[6] Jeff Erickson. *Algorithms*. 1st. 2019. ISBN: 978-1792644832.

[9] Vassos Hadzilacos. *Course notes for CSC B36/236/240 Introduction to Theory of Computation*. University of Toronto. 2007.

[11] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. 8th edition. New York: McGraw-Hill Education, 2019.

Journal Articles

[7] Michael Fredman, Janos Komlos, and Endre Szemeredi. “Storing a Sparse Table with O(1) Worst Case Access Time”. In: *Journal of the Association for Computing Machinery* 31.3 (July 1984), pages 538–544.

[8] Igal Galperin and Ronald L. Rivest. “Scapegoat Trees”. In: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (Jan. 1993), pages 165–174.