QUICK MATH

This reference material is organized into three main parts:

a) Fundamental theory across probability, analysis, number theory, geometry, and linear algebra.

b) Key techniques and implementation patterns for string processing, graph theory, and complexity classes.

c) A curated selection of challenging problems, including several personal favorites, with hints to aid recall.

Mathematics

Probability

Probability Space. A measure space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the sample space, $\mathcal{F}$ is the σ-algebra of events E, and $\mathbb{P} : \mathcal{F} \to [0,1]$ is the probability measure.

Fundamental Identities.

a) Independence $\mathbb{P}(E \cap F) = \mathbb{P}(E)\,\mathbb{P}(F)$

b) Conditional Probability $\mathbb{P}(E \mid F) = \frac{\mathbb{P}(E \cap F)}{\mathbb{P}(F)}$

c) Total Probability $\mathbb{P}(B) = \sum_i \mathbb{P}(B \mid E_i)\,\mathbb{P}(E_i)$

d) Bayes' Law $\mathbb{P}(E_i \mid B) = \frac{\mathbb{P}(B \mid E_i)\,\mathbb{P}(E_i)}{\sum_j \mathbb{P}(B \mid E_j)\,\mathbb{P}(E_j)}$

Linearity of Expectation.

$$\mathbb{E}\left[\sum X_i\right] = \sum \mathbb{E}[X_i]$$

Jensen's Inequality. *If f is convex, then* $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$.

Dist	PMF/PDF	Mean	Var
$\mathrm{Bern}(p)$	$p^x(1-p)^{1-x}$	p	pq
$\mathrm{Geom}(p)$	$q^{k-1}p$	$1/p$	q/p^2
$\mathrm{Bin}(n,p)$	$\binom{n}{k}p^k q^{n-k}$	np	npq
$\mathrm{Pois}(\mu)$	$e^{-\mu}\mu^k/k!$	μ	μ
$\mathrm{U}(a,b)$	$\frac{1}{b-a}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
$\mathrm{Exp}(\lambda)$	$\lambda e^{-\lambda x}$	$1/\lambda$	$1/\lambda^2$
$\mathrm{N}(\mu,\sigma^2)$	$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
$\mathrm{Cauchy}(\alpha,\beta)$	$\frac{1}{\pi\beta}\left[1+\left(\frac{x-\alpha}{\beta}\right)^2\right]^{-1}$	Undefined	Undefined
T_k	$\frac{\Gamma(\frac{k+1}{2})}{\sqrt{k\pi}\Gamma(\frac{k}{2})}\left(1+\frac{t^2}{k}\right)^{-\frac{k+1}{2}}$	0	$\frac{k}{k-2}$ for $k>2$
$\gamma(\alpha,\beta)$	$\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$
$\mathrm{Beta}(\alpha,\beta)$	$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$
$\chi^2(k)$	$\frac{x^{k/2-1}e^{-x/2}}{2^{k/2}\Gamma(k/2)}$	k	$2k$

Analysis

Metric Space. A pair (M,d) where $d : M \times M \to [0,\infty)$ satisfies:

a) $d(x,y) = 0 \iff x = y$

b) $d(x,y) = d(y,x)$

c) $d(x,z) \leq d(x,y) + d(y,z)$

Mean Value Theorem. *If f is continuous on $[a,b]$ and differentiable on (a,b), then $\exists \xi \in (a,b)$ such that:*

$$f'(\xi) = \frac{f(b)-f(a)}{b-a}$$

Integral MVT. *If f is continuous, $\exists z \in [a,b]$ such that:*

$$\int_a^b f(x)\,dx = f(z)(b-a)$$

Taylor's Theorem. $f(x) = T_n(x) + R_n(x)$ *where*

$$T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k.$$

Lagrange Remainder.

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$$

Newton's Method: To solve $f(x) = 0$, start with x_0 and iterate:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Converges quadratically if $f'(x^*) \neq 0$.

Number Theory

Fermat's Little Theorem. *If p is prime:*

$$a^{p-1} \equiv 1 \pmod p$$

Euler's Totient Function. $\phi(n)$ counts the positive integers up to n that are relatively prime to n.

$$\phi(n) = n\prod_{p \mid n}\left(1-\frac{1}{p}\right)$$

Euler's Theorem. *If* $\gcd(a,n) = 1$*:*

$$a^{\phi(n)} \equiv 1 \pmod n$$

Extended Euclidean Algorithm. *For integers a, b, there exist integers x, y such that:*

$$ax + by = \gcd(a,b)$$

$\hookrightarrow$ **Time:** $O(\log N)$, **Space:** $O(\log N)$

```
Pair euclidean(int a, int b) {
  if (b == 0) return { 1, 0 };
  int k = a / b;
  auto [x, y] = euclidean(b, a - (k*b));
  return { y, x - y*k };
}
```

Geometry

Orientation Check: To determine the orientation of the path from point p to q to r, check the sign of the cross product $(q-p) \times (r-q)$:

a) > 0: Left turn (Counter-clockwise)

b) < 0: Right turn (Clockwise)

c) $= 0$: Collinear

Polygon Area (Shoelace Formula). *The area of a polygon with vertices $(x_1,y_1)\ldots(x_n,y_n)$ is given by:*

$$A = \frac{1}{2}\left|\sum_{i=1}^{n}(x_i y_{i+1} - x_{i+1} y_i)\right|$$

where the indices wrap around ($x_{n+1} = x_1$, $y_{n+1} = y_1$).

Matrix Operations

LU Decomposition. For an invertible matrix A, factorization into a lower triangular matrix L and an upper triangular matrix U.

$$A = LU$$

Used for solving linear systems $Ax = b$ and computing determinants.

QR Decomposition. Factorization of A into an orthogonal matrix Q and an upper triangular matrix R.

$$A = QR$$

Often computed using Gram-Schmidt orthogonalization. Primarily used for solving linear least squares problems and as the basis for the QR algorithm for finding eigenvalues.

Eigen-Decomposition. *If an $n \times n$ matrix A is diagonalizable (has n linearly independent eigenvectors), it can be decomposed:*

$$A = V\Lambda V^{-1}$$

where V is the matrix of eigenvectors and Λ is a diagonal matrix containing the corresponding eigenvalues.
If A is symmetric, then V is an orthogonal matrix.

Singular Value Decomposition (SVD). *Any $m \times n$ matrix A can be decomposed as:*

$$A = U\Sigma V^T$$

where U is $m \times m$ orthogonal, V is $n \times n$ orthogonal, and Σ is $m \times n$ diagonal with non-negative real values (singular values σ_i).
Used for dimensionality reduction (PCA), low-rank approximation, and finding pseudo-inverses.

Algorithms

Strings

KMP Algorithm: Prefix function $\pi[i]$ is the length of the longest proper prefix of $S[0\dots i]$ that is also a suffix of $S[0\dots i]$.

↪ **Time:** $O(N)$, **Space:** $O(N)$

```
std::vector<int> kmp(std::string_view string) {
  int n = string.size();
  std::vector<int> prefix(n);
  for (int index = 1; index < n; ++index) {
    int p = prefix[index - 1];
    while (p > 0 && string[p] != string[index]) p
        = prefix[p - 1];
    if (string[p] == string[index]) p++;
    prefix[index] = p;
  }
  return prefix;
}
```

Z-Algorithm: $Z[i]$ is the length of the longest common prefix between S and the suffix starting at $S[i]$.

↪ **Time:** $O(N)$

```
std::vector<int> zFunction(std::string_view
    string) {
  int n = string.size();
  std::vector<int> z(n);
  int left = 0;
  int right = 0;
  for (int index = 1; index < n; ++index) {
    if (index <= right) {
      z[index] = std::min(right - index + 1,
          z[index - left]);
    }
    while (index + z[index] < n &&
        string[z[index]] == string[index +
        z[index]]) {
      z[index]++;
    }
    if (index + z[index] > right) {
      left = index, right = index + z[index] - 1;
    }
  }
  return z;
}
```

Graph Theory

Handshaking Lemma. *Sum of degrees is twice the edges:* $\sum_{v \in V} \deg(v) = 2|E|$.

Max-Flow Min-Cut. *The max flow from s to t equals the min capacity of an $s - t$ cut.*

Laplacian Matrix. $L = D - A$. Properties:

- $x^T L x = \sum_{(u,v)\in E}(x_u - x_v)^2$
- Number of connected components = $\dim(\ker(L))$

Union-Find (DSU): Maintains disjoint sets. Operations: `find` (path compression) and `unite` (by size).

↪ **Time:** $\alpha(N)$

```
class UnionFind {
  public:
    UnionFind(int n): parent(n), size(n, 1) {
      std::iota(parent.begin(), parent.end(), 0);
    }

    int find(int node) {
      if (parent[node] == node) return node;
      return find(parent[node]);
    }

    void join(int left, int right) {
      left = find(left);
      right = find(right);
      if (left == right) return;
      if (size[left] < size[right]) {
        std::swap(left, right);
      }
      parent[right] = left;
      size[left] += size[right];
    }

  private:
    std::vector<int> parent;
    std::vector<int> size;
};
```

Dijkstra's Algorithm: Finds shortest paths from source in graph with non-negative weights.

↪ **Time:** $O(E \log V)$

```
std::vector<int>
    dijkstra(std::vector<std::vector<Pair>>
    adjacencyList, int start) {
  int n = adjacencyList.size();
  std::vector<int> distance(n,
      std::numeric_limits<int>::max());
  distance[0] = 0;
  std::priority_queue<Pair, std::vector<Pair>,
      std::greater<Pair>> queue;
  queue.emplace(0, start);

  while (!queue.empty()) {
    auto [d, node] = queue.top();
    queue.pop();

    if (d > distance[node]) continue;
    for (auto [next, weight]: adjacencyList[node])
        {
      if (weight + d >= distance[next]) continue;
      distance[next] = weight + d;
      queue.emplace(distance[next], next);
    }
  }

  return distance;
}
```

Approximation

α-approximation. An algorithm A is an α-approximation for minimization if $A(I) \leq \alpha \cdot OPT(I)$ for all I.

APX. Class of problems admitting a constant-factor approximation (e.g., Vertex Cover, Metric TSP).

PTAS (Polynomial Time Approx Scheme). Algorithm A_ϵ produces $(1+\epsilon)$-solution in time $O(n^{f(1/\epsilon)})$. Example: Euclidean TSP.

FPTAS (Fully PTAS). Produces $(1+\epsilon)$-solution in time polynomial in both n and $1/\epsilon$. Example: Knapsack.

Inapproximability. Some problems cannot be approximated within factor ρ unless P=NP (e.g., General TSP, Set Cover within $(1-\epsilon)\ln n$).

NP-Hardness

Vertex Cover: Given graph $G = (V, E)$, find min-size $S \subset V$ such that $\forall (u, v) \in E$, $u \in S$ or $v \in S$.

2-Approximation: Find maximal matching M. Output all endpoints of M. $|M| \leq 2 \cdot OPT$.

Set Cover: Universe U, collection $\mathcal{S}$ of subsets. Find min sub-collection covering U.

Greedy Strategy: Repeatedly pick set covering most uncovered elements. $\alpha = H_n \approx \ln n$.

Metric TSP: Complete graph with triangle inequality $c_{ij} \leq c_{ik} + c_{kj}$.Find min cost Hamiltonian cycle.

Christofides (3/2-Approx):

a) Compute MST T.

b) Compute min-weight perfect matching M on odd-degree vertices of T.

c) Compute Eulerian tour on multigraph $T \cup M$.

d) Shortcut the tour to skip repeated vertices.

Knapsack: Items with weight w_i, value v_i. Maximize $\sum v_i$ subject to $\sum w_i \leq W$.

FPTAS: Scale values $v_i' = \lfloor v_i/K \rfloor$ where $K = \epsilon \max v_i/n$. DP on scaled values. $O(n^3/\epsilon)$.

Puzzles

Permutation Cycles **(Hint: Linearity of Expectation)**

A permutation on $[1, n]$ decomposes into disjoint cycles. What is the expected number of cycles in a random permutation?

Lattice Random Walk **(Hint: Independent axes, sum diverges)**

Classify simple random walks on $\mathbb{Z}^2$ and $\mathbb{Z}^3$ as transient or recurrent.

---

k-Suppliers **(Hint: Triangle inequality, dominating set)**

Given a set of suppliers F and customers D, find a subset $S \subseteq F$ with size $|S| \leq k$ that minimizes the maximum distance from any customer in D to the nearest supplier in S.

a) Give a 3-approximation algorithm for this problem.

b) Prove that there is no better approximation algorithm unless $\mathsf{P} = \mathsf{NP}$.

Ant on Rubber Rope **(Hint: Harmonic series diverges)**

Ant crawls at 1 cm/s. Rope stretches by 1 m/s every second. Does the ant reach the end?
