Complete Quantitative Finance & HFT Mastery Guide

Self-Study Roadmap with Free Resources, Projects, and Career Strategy

Executive Summary

This comprehensive guide provides a **complete, actionable roadmap** for mastering quantitative finance, high-frequency trading (HFT) systems programming, and competitive programming—all using **free, open-access resources**. The curriculum is designed for self-directed learners targeting roles as quantitative developers, HFT C++ engineers, or algorithmic traders at top-tier firms.

Key Outcomes:

- Master quantitative finance fundamentals (time series, portfolio theory, derivatives, ML trading)
- Develop production-grade C++ skills for low-latency systems
- Achieve competitive programming proficiency (LeetCode 2000+, HackerRank 1895+)
- Build a portfolio of 10+ impressive projects
- Prepare for technical interviews at elite quant/HFT firms

Total Study Duration: 6-9 months intensive (20-30 hours/week)

Phase 1: Foundations (Weeks 1-14)

Module 1.1: Mathematics & Statistics (4 weeks)

Core Topics:

- Probability theory and distributions
- Linear algebra (matrices, eigenvalues, decompositions)
- Calculus (derivatives, integrals, Taylor series)
- Statistical inference and hypothesis testing

Free Resources:

1. **MIT OpenCourseWare** - 18.S096 Topics in Mathematics with Applications in Finance [1] [2] [3]
    - Full video lectures on stochastic processes, Itō calculus, Black-Scholes derivation
    - Problem sets with solutions
    - Covers linear algebra, probability, regression, and portfolio theory
2. **Khan Academy** - Statistics and Probability (complete track)

3. **3Blue1Brown YouTube** - Linear Algebra series (visual intuition)
4. **SciPy/NumPy Documentation** - Statistical functions reference

Projects:

- Implement Monte Carlo simulations for option pricing
- Build correlation matrix visualizations for stock portfolios
- Code statistical hypothesis tests from scratch

Module 1.2: Python Programming for Finance (3 weeks)

Core Libraries:

- NumPy, Pandas, Matplotlib, SciPy
- yfinance, pandas-datareader (data acquisition)
- scikit-learn (machine learning)
- statsmodels (econometrics)

Free Resources:

1. **Python for Data Analysis** (McKinney) - free chapters online
2. **Quantitative Economics with Python** (QuantEcon lectures)
3. **Real Python** tutorials on financial data analysis

Projects:

- Download and clean 10+ years of S&P 500 data
- Calculate returns, volatility, and correlation matrices
- Implement basic technical indicators (SMA, EMA, RSI, Bollinger Bands)

Module 1.3: C++ Fundamentals (4 weeks)

Core Topics:

- Modern C++ (C++11/14/17/20 features)
- STL containers and algorithms
- Object-oriented design patterns
- Memory management basics

Free Resources:

1. CPPReference.com - comprehensive C++ reference [4] [5]
2. LearnCpp.com - complete C++ tutorial
3. **Compiler Explorer (**godbolt.org**)** - examine assembly output
4. **C++ Weekly** (YouTube) - Jason Turner's educational videos

Projects:

- Implement std::vector and std::unique_ptr from scratch
- Build a custom hash table with collision handling
- Create a simple trading order management system

Module 1.4: Financial Basics (3 weeks)

Core Concepts:

- Financial instruments (stocks, bonds, derivatives)
- Returns calculation (simple vs log returns)
- Risk metrics (volatility, VaR, Sharpe ratio)
- Market microstructure basics

Free Resources:

1. **Investopedia** - comprehensive financial encyclopedia
2. **"Quantitative Trading" by Ernest Chan** - summaries available online [6] [7]
3. **Yahoo Finance** - real-time market data

Projects:

- Analyze historical returns and volatility clustering
- Calculate alpha and beta vs market benchmark
- Build a dashboard showing portfolio risk metrics

Phase 2: Quantitative Finance Core (Weeks 15-28)

Module 2.1: Time Series Analysis & Forecasting (4 weeks)

Core Topics:

- Stationarity testing (ADF, KPSS)
- Autocorrelation (ACF) and Partial Autocorrelation (PACF)
- ARIMA models (AR, MA, ARIMA, SARIMAX)
- Exponential smoothing (Holt-Winters)
- GARCH models for volatility

Free Resources:

1. **"Time Series Forecasting with Python" ArXiv paper** [8] [9]
2. **statsmodels documentation** - ARIMA and GARCH tutorials [10]
3. **Machine Learning Mastery** - ARIMA implementation guide [11]
4. **arch package documentation** - GARCH modeling in Python [12]

Code Examples:

```
from statsmodels.tsa.arima.model import ARIMA
from arch import arch_model

# ARIMA forecasting
model = ARIMA(data, order=(5,1,0))
fitted = model.fit()
forecast = fitted.forecast(steps=10)

# GARCH volatility modeling
garch = arch_model(returns, vol='GARCH', p=1, q=1)
result = garch.fit()
```

Projects:

1. Forecast S&P 500 prices using ARIMA with walk-forward validation [13]
2. Model volatility clustering with GJR-GARCH [14]
3. Compare forecasting methods (naive, SMA, ARIMA) on multiple assets
4. Build automated ARIMA parameter selection pipeline

Module 2.2: Portfolio Optimization (3 weeks)

Core Topics:

- Modern Portfolio Theory (Markowitz)
- Mean-variance optimization
- Efficient frontier construction
- Sharpe ratio maximization
- Capital Asset Pricing Model (CAPM)

Free Resources:

1. **PyPortfolioOpt documentation** - Python implementation guide [15] [16]
2. **MIT OCW Finance Theory** - CAPM and APT lectures [17] [18]
3. **"Modern Portfolio Theory" YouTube video** - BioniFinance [19]

Code Examples:

```
from pypfopt import EfficientFrontier, risk_models, expected_returns

# Calculate expected returns and covariance
mu = expected_returns.mean_historical_return(prices)
S = risk_models.sample_cov(prices)

# Optimize for maximum Sharpe ratio
ef = EfficientFrontier(mu, S)
weights = ef.max_sharpe()
```

Projects:

1. Build mean-variance optimizer for 10-stock portfolio [20]
2. Plot efficient frontier with 1000+ random portfolios
3. Compare GMV, max Sharpe, and equal-weight portfolios
4. Implement portfolio with L2 regularization constraints

Module 2.3: Derivatives & Options Pricing (4 weeks)

Core Topics:

- Option fundamentals (calls, puts, Greeks)
- Black-Scholes model
- Binomial option pricing
- Monte Carlo pricing methods
- Implied volatility

Free Resources:

1. **MIT OCW** - Black-Scholes derivation lectures [21]
2. **Black-Scholes Python Implementation** tutorial [22]
3. **Options pricing notebooks** - GitHub repositories

Code Examples:

```
from scipy.stats import norm
import numpy as np

def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma*np.sqrt(T)
    return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)
```

Projects:

1. Implement Black-Scholes pricing and Greeks calculation [23]
2. Build Monte Carlo option pricer with variance reduction
3. Calculate implied volatility using Newton-Raphson
4. Visualize option payoff diagrams and Greek sensitivities

Module 2.4: Algorithmic Trading Strategies (3 weeks)

Core Topics:

- Trend-following strategies
- Mean reversion and pairs trading
- Statistical arbitrage

- Machine learning-based strategies
- Backtesting frameworks

Free Resources:

1. **OpenAlgo** - open-source trading platform [24] [25]
2. **Statistical Arbitrage GitHub repo** (rzhadev1/statarb) [26] [27]
3. **Pairs Trading Tutorial** - Medium/Databento [28]
4. **Backtrader documentation** - Python backtesting library

Projects:

1. Implement moving average crossover strategy with backtesting
2. Build pairs trading system with cointegration testing [29]
3. Create ML-based classifier for buy/sell signals [30]
4. Develop reinforcement learning trading agent [31] [32]

Phase 3: HFT & Low-Latency Systems (Weeks 29-46)

Module 3.1: Advanced C++ & Multithreading (6 weeks)

Core Topics:

- Templates and metaprogramming (CRTP, SFINAE)
- Move semantics and perfect forwarding
- Smart pointers (unique_ptr, shared_ptr)
- Multithreading (std::thread, std::async)
- Thread synchronization (mutex, condition_variable)
- Lock-free programming (std::atomic, memory_order)

Free Resources:

1. **CPPNuts YouTube** - Complete Threading Course [33] [34] [35]
    - Mutex, lock_guard, unique_lock
    - Producer-consumer patterns
    - Lock-free data structures
2. **"C++ Design Patterns for HFT" ArXiv paper** [36]
3. **Effective Modern C++** - online summaries and notes
4. **CppCon talks** - YouTube channel (threading, optimization)

Code Examples:

```
// Lock-free SPSC queue
template&lt;typename T&gt;
class LockFreeQueue {
    std::atomic&lt;size_t&gt; read_idx{0};
    std::atomic&lt;size_t&gt; write_idx{0};
    std::vector&lt;T&gt; buffer;
public:
    bool push(const T&amp; item) {
        size_t current_write = write_idx.load(std::memory_order_relaxed);
        size_t next_write = (current_write + 1) % buffer.size();
        if(next_write == read_idx.load(std::memory_order_acquire))
            return false;
        buffer[current_write] = item;
        write_idx.store(next_write, std::memory_order_release);
        return true;
    }
};
```

Projects:

1. Implement lock-free SPSC queue [37] [38]
2. Build thread pool with work-stealing scheduler
3. Create producer-consumer system with multiple threads [39]
4. Develop custom memory pool allocator

Module 3.2: Operating Systems & Networking (4 weeks)

Core Topics:

- Linux kernel basics (scheduling, virtual memory)
- Process and thread management
- TCP/IP and UDP protocols
- Socket programming
- Kernel bypass techniques (DPDK)
- NUMA awareness

Free Resources:

1. **"User Space Network Drivers" ArXiv paper** [40]
2. **DPDK Getting Started Guide** [41]
3. **Kernel Bypass blog post** - CloudFlare [42]
4. **Beej's Guide to Network Programming** - free online book
5. **Linux kernel development tutorials**

Projects:

1. Build TCP echo server with epoll [43]

2. Implement UDP multicast publisher/subscriber
3. Create packet sniffer using raw sockets
4. Develop high-throughput market data simulator

Module 3.3: Competitive Programming (Ongoing)

Target Metrics:

- **LeetCode Rating:** ≥2000
- **HackerRank:** ≥1895
- **CodeChef:** ≥1800 (Division 1)

Topic Progression:

1. **Beginner (1200-1400):** Arrays, strings, basic STL, sorting
2. **Intermediate (1500-1700):** DP, graphs, segment trees
3. **Advanced (1800-2000+):** Advanced DP, flow algorithms, geometry

Free Resources:

1. **Competitive Programming Roadmap** (Scribd/QuantNet) [44] [45]
2. **CSES Problem Set** - 300 curated problems
3. **Codeforces** - regular contests (Div 1/2/3)
4. **LeetCode** - daily challenges and contests
5. **Errichto YouTube** - algorithm explanations

Weekly Practice Schedule:

- 3-4 LeetCode problems daily
- 2 Codeforces contests per week
- 1 long CodeChef contest monthly
- Upsolve all problems not solved during contests

Resume-Worthy Achievements:

- LeetCode 2000+ rating with profile badge [46]
- Top 10% in CodeChef Long Challenge
- Division 1 promotion on Codeforces
- Global rank <1000 in any major contest

Module 3.4: Low-Latency Optimization (4 weeks)

Core Topics:

- Cache optimization and false sharing
- SIMD and vectorization (AVX/AVX2)
- Memory alignment and padding
- Branch prediction optimization
- Profiling and benchmarking

Free Resources:

1. **Intel Optimization Manual** - free PDF
2. **Agner Fog's optimization guides** - comprehensive C++ optimization
3. **"SIMD and Vectorization using AVX" tutorial** [47] [48]
4. **Intel Intrinsics Guide** - online reference [49]

Code Examples:

```
// AVX vectorized dot product
#include &lt;immintrin.h&gt;

float dot_product_avx(const float* a, const float* b, size_t n) {
    __m256 sum = _mm256_setzero_ps();
    for(size_t i = 0; i &lt; n; i += 8) {
        __m256 va = _mm256_loadu_ps(&amp;a[i]);
        __m256 vb = _mm256_loadu_ps(&amp;b[i]);
        sum = _mm256_fmadd_ps(va, vb, sum);
    }
    // Horizontal sum reduction
    __m128 lo = _mm256_castps256_ps128(sum);
    __m128 hi = _mm256_extractf128_ps(sum, 1);
    __m128 result = _mm_add_ps(lo, hi);
    // ... continue reduction
}
```

Projects:

1. Implement cache-aware matrix multiplication [50]
2. Vectorize array operations using AVX intrinsics [51]
3. Build benchmarking framework for latency measurement
4. Optimize order book operations for L1 cache

Phase 4: Portfolio Development & Career Preparation (Ongoing)

Essential Projects Portfolio

Build **10 production-grade projects** covering different domains:

Quantitative Finance Projects:

1. Multi-Factor Statistical Arbitrage System

- Technologies: Python, NumPy, Pandas, scikit-learn
- Features: Pairs selection via cointegration, Z-score trading, PnL tracking
- GitHub: Similar to rzhadev1/statarb [52]

2. GARCH Volatility Forecasting Dashboard

- Technologies: Python, arch, Plotly, Streamlit
- Features: GJR-GARCH modeling, rolling forecasts, asymmetry analysis [53]

3. Mean-Variance Portfolio Optimizer

- Technologies: Python, PyPortfolioOpt, cvxpy
- Features: Efficient frontier, constrained optimization, backtesting [54]

4. Reinforcement Learning Trading Agent

- Technologies: Python, Stable-Baselines3, Gym
- Features: Custom trading environment, Q-learning, DQN, performance metrics [55]

HFT/Low-Latency Projects:

5. Lock-Free Order Book in C++

- Technologies: C++17, std::atomic, Red-Black trees
- Features: FIFO matching, nanosecond timestamps, 450K+ ops/sec [56] [57]

6. Low-Latency Market Data Handler

- Technologies: C++, Multithreading, UDP multicast
- Features: FIX protocol parsing, lock-free queues, tick-to-trade <10µs

7. Matching Engine Simulator

- Technologies: C++, Order matching algorithms
- Features: Price-time priority, trade reporting, performance profiling

8. Custom Memory Pool Allocator

- Technologies: C++, Memory management
- Features: NUMA-aware allocation, object pooling, zero fragmentation

Systems Programming Projects:

9. High-Performance TCP/UDP Server

- Technologies: C++, epoll, Non-blocking I/O
- Features: Concurrent connections, zero-copy, >100K requests/sec

10. Vectorized Numerical Library (AVX2)

- Technologies: C++, SIMD intrinsics
- Features: Matrix operations, dot products, 10x+ speedup vs scalar [58]

Resume Optimization for HFT Roles

Structure:

```
[YOUR NAME]
Computer Science | Quantitative Finance | Low-Latency Systems
Email | GitHub | LinkedIn | LeetCode/HackerRank profiles

COMPETITIVE PROGRAMMING ACHIEVEMENTS
• LeetCode: 2050 rating (Top 5%), 500+ problems solved
• HackerRank: 1920 rating (6-star Gold badge)
• CodeChef: Division 1 (1850 rating), Long Challenge Top 100

TECHNICAL SKILLS
Languages: C++ (17/20), Python, SQL, R
HFT/Systems: Multithreading, Lock-free queues, SIMD (AVX2), DPDK, Linux kernel
Quant Finance: Time series (ARIMA/GARCH), Portfolio optimization, Derivatives pricing
Libraries: STL, Boost, NumPy, Pandas, PyPortfolioOpt, statsmodels, arch

LOW-LATENCY PROJECTS
Lock-Free Order Book Engine | C++17, std::atomic, Red-Black trees
• Implemented FIFO matching with 450K+ operations/second throughput
• Optimized memory layout achieving &lt;100ns average latency per operation
• Reduced cache misses by 40% through alignment and false sharing elimination

Market Data Handler with DPDK | C++, Kernel bypass, UDP multicast
• Built zero-copy packet processor handling 10M+ messages/second
• Achieved &lt;5µs tick-to-trade latency using lock-free ring buffers
• Implemented FIX protocol parser with SIMD optimizations

QUANTITATIVE FINANCE PROJECTS
Statistical Arbitrage Trading System | Python, Cointegration, Backtesting
• Developed pairs trading strategy with 1.8 Sharpe ratio over 3-year backtest
• Implemented Kalman filter for dynamic beta estimation
• Achieved 15% annual return with &lt;8% maximum drawdown

GARCH Volatility Forecasting | Python, arch, GJR-GARCH
• Modeled S&amp;P 500 volatility with 0.74 correlation to realized volatility
• Captured asymmetric response to negative shocks (leverage effect)
• Built real-time dashboard for VaR calculation and risk monitoring
```

Key Resume Tips:

- **Quantify everything:** Latency numbers, throughput, Sharpe ratios, ratings
- **Highlight optimizations:** "Reduced latency by X μs via Y technique"
- **Show depth:** Mention specific technologies (AVX2, DPDK, lock-free, NUMA)
- **Link GitHub:** Every project should have documented code

Interview Preparation Strategy

Technical Interview Components:

1. LeetCode-Style Coding (1-2 rounds)

- Focus: Graph algorithms, DP, segment trees
- Preparation: 500+ problems, emphasize medium/hard [59]
- Time: 45-60 min per round

2. C++ Deep Dive (2-3 rounds)

- Implement: unique_ptr, shared_ptr, std::vector, lock-free queue [60]
- Discuss: Virtual functions, CRTP, move semantics, template specialization
- Optimization: SIMD, cache, memory ordering [61]

3. OS/Networking/Architecture (2-3 rounds)

- Topics: Threading, NUMA, TCP/UDP, kernel bypass, CPU cache [62]
- Questions: "Explain false sharing," "How does DPDK work?"
- Hands-on: Debug multithreading issues, analyze cache misses

4. Probability & Quant Puzzles (1-2 rounds)

- Resources:
  - "Heard on the Street" by Timothy Crack [63] [64]
  - "Quant Interview Questions" by Mark Joshi [65]
  - Jane Street puzzles [66] [67]
- Topics: Expected value, game theory, Bayesian inference
- Practice: 100+ probability problems

Free Interview Prep Resources:

1. **QuantNet study programme** - Mark Joshi questions, Jane Street guide [68]
2. **Glassdoor** - Company-specific interview experiences [69]
3. **r/cpp Reddit threads** - "HFT interview questions" [70]
4. **Jane Street blog** - Probability puzzles and solutions [71]

HFT Firms in India (Referral Targets):

1. **Quantbox Research** - Bangalore [72]
   - Hiring: C++ developers, Quant researchers
   - Focus: Market-making, systematic trading
2. **Graviton Research Capital** - Gurgaon
   - Roles: HFT C++ developer, Low-latency engineer
3. **Orbis Financial** - Mumbai
   - Focus: Derivatives trading, Options market-making
4. **Tower Research Capital** - Gurgaon
   - Global HFT firm with India operations
5. **AlphaGrep** - Bangalore
   - Quant trading and technology roles

Referral Strategy:

- Build strong GitHub presence with HFT-relevant projects
- Contribute to open-source financial libraries
- Network on LinkedIn with keywords: "HFT," "Low-latency," "Quant dev"
- Attend virtual quant finance meetups
- Reach out respectfully with project portfolio (not cold resume spam)
- Leverage alumni networks from IIT/top engineering colleges

Study Schedule & Progress Tracking

Recommended Weekly Schedule (30 hours):

Day	Time	Activity
Mon-Fri	2 hours/day	Quant finance coursework + projects
Mon-Fri	1 hour/day	C++ practice + HFT concepts
Mon-Fri	1 hour/day	Competitive programming (LeetCode)
Sat	4 hours	Long coding session (project work)
Sat	2 hours	CodeChef/Codeforces contest
Sun	4 hours	Study new concepts, review week
Sun	2 hours	Interview prep (puzzles, system design)

Monthly Milestones:

- Month 1: Python setup, basic stats, 50 LeetCode problems
- Month 2: ARIMA implementation, C++ STL mastery, 100 problems

- Month 3: Portfolio optimization, threading basics, 150 problems
- Month 4: Options pricing, lock-free queues, LeetCode 1600+
- Month 5: RL trading, SIMD optimization, LeetCode 1800+
- Month 6: First 3 projects completed, LeetCode 2000+
- Month 7-9: Remaining projects, interview prep, applications

Progress Tracker (use spreadsheet):

- Topics studied: Date completed, confidence level (1-5)
- Problems solved: Platform, difficulty, success rate
- Projects: GitHub link, completion %, documented?
- Interview prep: Mock interviews done, weak areas

Key Free Resource Summary

Textbooks & Papers (Open Access):

1. Applied Quantitative Finance - Hardle et al. (free PDF) [73]
2. Quantitative Trading - Ernest Chan (summaries available) [74]
3. Time Series Forecasting - ArXiv tutorials [75]
4. C++ Design Patterns for HFT - ArXiv paper [76]

Online Courses (Free):

1. MIT OpenCourseWare - Finance Theory, Stochastic Calculus [77] [78]
2. Reinforcement Learning for Trading - CognitiveClass [79]
3. Python for Finance - QuantEcon lectures

Video Tutorials:

1. CPPNuts - Complete Threading in C++ [80]
2. Bionic Turtle - CAPM and Portfolio Theory [81]
3. 3Blue1Brown - Linear Algebra (visual)
4. Dataquest - Stock Prediction with ML [82]

Code Repositories (GitHub):

1. PyPortfolioOpt - Portfolio optimization [83]
2. OpenAlgo - Trading platform [84]
3. rzhadev1/statarb - Statistical arbitrage [85]
4. timothewt/OrderBook - Matching engine [86]

Practice Platforms:

1. LeetCode - Competitive programming
2. Codeforces - Algorithmic contests
3. HackerRank - Skill certification
4. CodeChef - Long challenges
5. CSES - Curated problem set

Interview Resources:

1. Heard on the Street - Timothy Crack [87]
2. Mark Joshi quant questions [88]
3. Jane Street puzzles [89]
4. Glassdoor interview experiences

Common Pitfalls & How to Avoid Them

1. Tutorial Hell: Don't endlessly watch tutorials

- **Solution:** 70% coding practice, 30% learning theory

2. Neglecting Fundamentals: Jumping to advanced topics too fast

- **Solution:** Master probability, linear algebra, C++ STL first

3. No Portfolio: Learning without building

- **Solution:** Start projects immediately, iterate on them

4. Isolated Learning: Not networking or sharing work

- **Solution:** GitHub commits, LinkedIn posts, community engagement

5. Over-Optimization: Premature performance tuning

- **Solution:** Correctness first, then measure and optimize

6. Ignoring Soft Skills: Only technical preparation

- **Solution:** Practice explaining complex topics simply

Conclusion & Next Steps

This guide provides everything needed to transition into quantitative finance and HFT roles through structured self-study with free resources. The key differentiators for success are:

1. **Consistent daily practice** (competitive programming)
2. **Production-quality projects** (not tutorials)
3. **Deep technical knowledge** (C++, OS, optimization)

4. **Quantifiable achievements** (ratings, metrics, performance)

Immediate Action Items:

1. Download all CSV study guides (attached)
2. Set up development environment (Python, C++, GitHub)
3. Create LeetCode/HackerRank/CodeChef accounts
4. Start Day 1: Python + NumPy tutorial + 1 LeetCode problem
5. Build weekly study schedule in calendar with reminders

Remember: Elite quant/HFT firms hire based on demonstrated ability—your GitHub, competitive programming ratings, and project quality speak louder than credentials. Focus on building undeniable proof of skill through the projects outlined in this guide.

Good luck on your quantitative finance journey!

Appendix: Quick Reference Cheat Sheets

Financial Formulas:

Sharpe Ratio:

$$\text{Sharpe} = \frac{E[R_p - R_f]}{\sigma_p}$$

CAPM:

$$E[R_i] = R_f + \beta_i(E[R_m] - R_f)$$

Black-Scholes Call:

$$C = S_0N(d_1) - Ke^{-rT}N(d_2)$$

where $d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$, $d_2 = d_1 - \sigma\sqrt{T}$

GARCH(1,1):

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

C++ Performance Tips:

1. **Cache alignment:** `alignas(64)` for cache line size
2. **False sharing:** Separate frequently-accessed data by 64 bytes
3. **Lock-free:** Use `std::memory_order_acquire/release` for synchronization
4. **SIMD:** `__m256` for 8 floats, `_mm256_fmadd_ps` for FMA

Interview Question Patterns:

Probability: Expected value calculations, conditional probability
C++: Implement smart pointers, explain virtual dispatch cost
Systems: Cache coherence, TCP congestion control, NUMA
Quant: Option pricing, portfolio optimization, time series stationarity

This guide is continuously updated. Check GitHub for latest version and community contributions.

[90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108]

⁂

1. https://arxiv.org/pdf/1807.03890.pdf
2. https://arxiv.org/pdf/2209.08867.pdf
3. https://arxiv.org/pdf/2203.11972.pdf
4. http://arxiv.org/pdf/2406.20027.pdf
5. https://arxiv.org/pdf/2311.06621.pdf
6. https://arxiv.org/pdf/2305.10239.pdf
7. https://arxiv.org/pdf/2212.13815.pdf
8. https://dash.harvard.edu/bitstream/1/11878778/3/CFABlyth2.pdf
9. https://snap.berkeley.edu/project/11166188
10. http://rlaexp.com/studio/see/all-years.html
11. https://www.akwi.de/documents/AKWI_Tagungsband2022.pdf
12. https://raw.githubusercontent.com/meetDeveloper/freeDictionaryAPI/master/meta/wordList/english.txt
13. https://nlp.biu.ac.il/~ravfogs/resources/embeddings-alignment/glove_vocab.250k.txt
14. https://freecomputerbooks.com/Applied-Quantitative-Finance.html
15. https://github.com/topics/algorithmic-trading
16. https://zorro-project.com
17. https://content.e-bookshelf.de/media/reading/L-7846948-7ddd68d39c.pdf
18. https://github.com/marketcalls/openalgo
19. https://arxiv.org/pdf/2205.10941.pdf
20. http://thesai.org/Downloads/Volume14No1/Paper_33-Time_Series_Forecasting_using_LSTM_and_ARIMA.pdf
21. https://www.ccsenet.org/journal/index.php/ijsp/article/download/0/0/40839/42762
22. https://www.preprints.org/manuscript/202103.0269/v1/download
23. http://arxiv.org/pdf/2410.09567.pdf
24. http://conference.scipy.org/proceedings/scipy2011/pdfs/statsmodels.pdf
25. https://arxiv.org/pdf/2207.03517.pdf
26. https://arxiv.org/ftp/arxiv/papers/1803/1803.06386.pdf
27. https://www.machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
28. https://github.com/ishujaswani/Mean-Variance-Optimiser
29. https://cognitiveclass.ai/courses/course-v1:IBMSkillsNetwork+GPXX0SQEN+v1

30. https://www.educative.io/answers/how-to-perform-time-series-forecasting-using-arima-in-python
31. https://github.com/Yassine19HALLAL/Portfolio_optimization_using_pyportfolioopt_package
32. http://arxiv.org/pdf/2401.10834.pdf
33. https://arxiv.org/pdf/2201.02179.pdf
34. https://arxiv.org/pdf/2305.07229.pdf
35. https://arxiv.org/pdf/2309.04259.pdf
36. https://arxiv.org/pdf/2105.10486.pdf
37. http://arxiv.org/pdf/2407.15805.pdf
38. http://arxiv.org/pdf/2402.17674.pdf
39. https://arxiv.org/pdf/1810.13029.pdf
40. https://github.com/SourenaMOOSAVI
41. https://www.youtube.com/watch?v=eZ8yKZo-PGw
42. https://www.scribd.com/document/659486942/Competitive-Programming-Roadmap
43. https://stacygaudreau.com/blog/cpp/low-latency-cpp-for-hft-part2/
44. https://www.youtube.com/watch?v=zUy66Bats5c
45. https://in.talent.com/view?id=d7fe1c1af7ec
46. https://cdn.bookey.app/files/pdf/book/en/quantitative-trading.pdf
47. https://github.com/deekshasharma/inmemorydb
48. https://www.instahyre.com/job-336716-c-plus-plus-developer-at-hft-bangalore/
49. https://pdfdrive.com.co/quantitative-trading-strategies-pdf/
50. https://www.mdpi.com/2673-8392/1/3/70/pdf
51. https://arxiv.org/html/2504.01239v1
52. https://journal.alt.ac.uk/index.php/rlt/article/download/1033/1283
53. https://pmc.ncbi.nlm.nih.gov/articles/PMC509291/
54. https://arxiv.org/pdf/2211.07212.pdf
55. https://arxiv.org/pdf/2312.16448.pdf
56. https://infedu.vu.lt/journal/INFEDU/article/464/file/pdf
57. https://aemps.ewapublishing.org/media/5d0044ee17f94de6ba74e893fd2da3a8.marked_HomfAFQ.pdf
58. https://www.youtube.com/watch?v=UiLKoRKppY8
59. https://www.youtube.com/watch?v=1O_BenficgE
60. https://github.com/timothewt/OrderBook
61. https://ocw.mit.edu/courses/15-401-finance-theory-i-fall-2008/pages/video-lectures-and-slides/the-capm-and-apt/
62. https://www.youtube.com/watch?v=QIUxPv5PJOY
63. http://arxiv.org/pdf/2501.10138.pdf
64. https://arxiv.org/pdf/2301.09470.pdf
65. http://arxiv.org/pdf/2403.03377.pdf
66. https://arxiv.org/pdf/2303.04404.pdf
67. https://arxiv.org/pdf/1901.10664.pdf

68. http://arxiv.org/pdf/2404.11107.pdf
69. http://arxiv.org/pdf/2402.10513v1.pdf
70. https://arxiv.org/pdf/2308.01983.pdf
71. https://blog.cloudflare.com/kernel-bypass/
72. https://www.hireitpeople.com/resume-database/64-java-developers-architects-resumes/116063-senior-quantitative-c-developer-resume
73. https://www.askpython.com/python/examples/black-scholes-model
74. https://www.youtube.com/watch?v=MpjlWt7fvrw
75. https://www.mintresume.com/resumes/quant-developer
76. https://arxiv.org/pdf/2310.08284.pdf
77. https://arxiv.org/pdf/2403.12180.pdf
78. https://arxiv.org/pdf/2206.14932.pdf
79. http://arxiv.org/pdf/1608.03636.pdf
80. https://joss.theoj.org/papers/10.21105/joss.01591.pdf
81. https://arxiv.org/pdf/1405.2384.pdf
82. https://arxiv.org/pdf/2406.06706.pdf
83. https://downloads.hindawi.com/archive/2006/073803.pdf
84. https://github.com/rzhadev1/statarb
85. https://www.reddit.com/r/cpp/comments/1i8zct5/interview_questions_at_hft_firms_for_c_roles/
86. https://www.scribd.com/document/848067830/Quantitative-Analyst-Roadmap
87. https://medium.databento.com/build-a-pairs-trading-strategy-in-python-a-step-by-step-guide-dcee006e1a50
88. https://www.youtube.com/watch?v=tWCsaSMphP4
89. https://www.mdpi.com/2227-9091/4/1/5/pdf
90. https://www.youtube.com/watch?v=AT5nuQQO96o
91. https://quantum-journal.org/papers/q-2023-11-29-1191/pdf/
92. https://quantum-journal.org/papers/q-2022-07-20-770/pdf/
93. https://www.aclweb.org/anthology/2021.naacl-main.160.pdf
94. https://github.com/Aniruddha-Deb/quant-prep
95. https://opencw.aprende.org/courses/mathematics/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/this-course-at-mit/
96. https://www.quantboxresearch.com/jobs
97. https://quantnet.com/threads/jane-street-interview-questions.3039/
98. https://ocw.mit.edu/courses/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/video_galleries/video-lectures/
99. http://arxiv.org/pdf/2201.11070.pdf
100. https://arxiv.org/pdf/2202.11309.pdf
101. http://arxiv.org/pdf/1907.09415.pdf
102. http://arxiv.org/pdf/1904.05808.pdf
103. http://arxiv.org/pdf/2011.04672.pdf

104. https://arxiv.org/abs/2301.04020
105. https://quantnet.com/threads/study-programme-for-quant-researcher-interviews.50152/
106. https://stackoverflow.blog/2020/07/08/improving-performance-with-simd-intrinsics-in-three-use-cases/
107. https://blog.quantinsti.com/garch-gjr-garch-volatility-forecasting-python/
108. https://www.scribd.com/document/906041667/Heard-on-the-Street-PDF