FINANCIAL ENGINEERING
AND
COMPUTATION:

PRINCIPLES, MATHEMATICS, ALGORITHMS

Yuh-Dauh Lyuu
National Taiwan University
Taipei, Taiwan

Acknowledgments

Many people contributed to the writing of the book: George E. Andrews (Pennsylvania State University), Nelson H.F. Beebe (University of Utah), Alesandro Bianchi (Caboto Holding Sim, Milan), Tomas Björk (Stockholm School of Economics), Peter Carr (Morgan Stanley Dean Witter), Jack S.K. Chang (the National University of Singapore and California State University), Paul Wei-Kuang Chen (National Cheng-Chi University), Ren-Raw Chen (Rutgers University and Lehman Brothers), Shu-Heng Chen (National Cheng-Chi University), Oren Cheyette (BARRA), Mark Fisher (Federal Reserve Board, D.C.), Mau-Wei Hung (National Taiwan University), Ming-Yang Kao (Yale), Gow-Hsing King (Academia Sinica), Philip Liang (Dow Jones), Steven C. Lin (Franklin Portfolio Associates), Andrew W. Lo (MIT), László Lovász (Yale), Robert Lum (Intrepid Technology), Chris McLean (Dynamic Data Access), Michael O. Rabin (Harvard and the Hebrew University), Douglas G. Rogers (Alabama State University), Masako Sato (University of Osaka), Erik Schlögl (University of Bonn), and James A. Tilley (Morgan Stanley Dean Witter).

Many ex-colleagues at Citicorp Securities deserve my deep thanks for the intellectual stimuli: Mark Bourzutschky, Michael Chu, Burlie Jeng, Andy Liao, Ben Lis, James Liu, Andy Sparks, and Frank Feikeh Sung. In particular, Sparks and Liao taught me a lot about the market.

Many students helped improve the quality of the book: Ronald Yan-Cheng Chang (chief developer of the Web software for the book); Kun-Yuan Chao (implementer of the binomial double barrier option algorithm [153]); Wei-Jui Chen (author of the differential tree program for the Black-Derman-Toy model [161]); Jing-Hong Chou; Tian-Shyr Dai (author of the geometric Asian option program [211]); Hau-Ren Fang; Jia-Hau Guo (implementor of the option-adjusted-spread model of mortgage-backed securities [359]); Yon-Yi Hsiao; How-Ming Hwang of Taiwan's Securities and Futures Commission (for his unbounded interest in financial computation); Kuang-Shieng Hwang (for his ability to come up with unexpected solutions; author of the binomial tree program for options used in the book); Heng-Yi Liu; Yu-Hong Liu (implementer of the binomial barrier option algorithm [527]); Ruo-Ming Sung; Eric Chen-Leh Wang of Citibank, Taiwan (my first graduate student [771]); Huang-Wen Wang (whose study of Monte Carlo, variance reduction, and quasi-Monte Carlo methods in [772] continues to amaze me with its thoroughness and honesty); Hsing-Kuo Wong; Chao-Sheng Wu.

My research has been partially supported by grants from the National Science Council in the Humanities in the period 1996–1999 and from the Tzong Tsuo-Zhang Foundation in

1997. The support of the Department of Computer Science and Information Engineering (CSIE), National Taiwan University is most indispensable.

Finally, I want to thank my wife, Chih-Lan, and my son, Raymond, for their being with me during the five-year-long project. This book, I hope, puts to rest their dreadful question, "When are you going to finish the book?"

Chapter 1

Introduction

Contents

> *But the age of chivalry is gone. That of sophisters,*
> *oeconomists, and calculators, has succeeded;*
> *and the glory of Europe is extinguished for ever.*
> —Edmund Burke (1729–1797),
> *Reflections on the Revolution in France* [123, p. 170]

1.1 An Overview

Today, the wide varieties of financial instruments dazzle even the knowledgeable. Brokerage firms offer accounts that are almost indistinguishable from bank accounts. Besides stocks and bonds, individuals can trade options, futures, stock index options, among countless others. When it comes to diversification, one has thousands of mutual funds to choose from categories in equity, bond, real estate, sector, international, money market, closed-end, open-end, and so on. Life insurance policies can be either term or whole-life, and new innovations are coming to the market constantly. Corporations and local governments increasingly use complex derivative instruments to manage their financial risks or even speculate. All of these are the results of **financial engineering**, which means structuring financial instruments to target investor preferences or take advantage of arbitrage opportunities [557].

The benefits of such wide ranges of innovations will not be confined only to industrialized nations. As developing countries become more prosperous and their markets more interwined with the global economy, they will increasingly look to the global financial market for their investment choices and capital. In so doing, they will rely on similar financial instruments to manage risks.

The innovations in the financial markets are paralleled by equally explosive progress in computer technology. In fact, one can not think of doing modern finance without it. Use of computers makes possible automated trading, efficient bookkeeping, timely clearing and settlements, real-time data feed, huge databases, tracking and monitoring of market conditions, sophisticated modeling of key economic indicators, and fast security pricing [557, 559, 769]. Harnessing computing power is essential to incessant innovations in the financial world; instruments have to be priced quickly and consistently, and sophisticated pricing models and new strategies need to be tested thoroughly. With the dependence of the financial industry on computers comes the recognition that **financial computation** is a distinct field worthy of study. The subject matter centers around applying computers to model the behavior of financial securities and key indicators, price financial instruments, and find combination of financial assets to achieve results consistent with risk exposures.

Understanding the ideas underlying the computation is essential to future innovations and improvements in the use of computers. We are not studying computer science or mathematics for its own sake. Rather, we are concerned with computation in the context of financial analytics and engineering. As a result, the fundamental principles of finance will be adequately covered. As the United States has the world's most efficient financial markets and is at the forefront of financial innovations, we will allude to those markets most of the time unless stated otherwise.

1.2 Idea of the Market

In a market economy, individuals pursue their own economic interests rather than being commanded. They make their own choices in buying cars, selecting careers, or investing to achieve some subjective level of satisfaction. Since the choices of many individuals are made in a decentralized fashion, their interests are bound to conflict with one another, the supply of economic goods being limited. Hence, a problem of co-ordination arises [379]. Such competing interests are reconciled and co-ordinated through the so-called **price mechanism**, which communicates information as people compete for the goods [381]. Any price change ripples through the whole economy, affecting the calculation of any person who needs such information.

The beauty of the system lies in that the pursuit of private interests of individuals often promote the welfare of the society as a whole without the use of coercion. It is as if, to use the immortal words of Adam Smith, they were led by an "invisible hand" [708]. Moreover, social welfare can often be achieved better this way than a more overt approach.[1] That a price mechanism can impose order upon the behavior of economic agents without any centralized control was recognized only some three hundred years ago, and Adam Smith, often called the father of economics, was among the first to fully grasp its implications [84].

---

[1] Adam Smith remarked that [709], "I have never known much good done by those who affected to trade for the public good."

Market economy works better than a command economy because no single person or government agency can be trusted to have all the information and act upon it fast enough without putting its own interest first. Individuals with sufficient knowledge of their particular circumstances are almost always better situated to adapt to them and exploit relevant information communicated by the price mechanism. Such information is the result of the competition among individuals who respond to the price signal and whose actions in turn add to it. As one philosopher so aptly put it [630], "A grocer who opens a new shop is conducting a social experiment." Evidently, the price mechanism will fulfill its function of transmitting information only under competition; without it, there would not be meaningful prices.

Competition is not a process that starts with all the knowledge given to all. Rather, it is a **discovery process**. It is a process whereby people obtain, respond to, and communicate information[2] [384]. In its fullest expression under modern capitalist societies, competition from new technology and organization periodically brings about fundamental changes in the market and society [687]. A capitalist society is thus in a permanent state of revolution, destruction, and creation. Capital, remarked Marx [558], "can be understood only as motion, not as a thing at rest."

1.3 Financial Markets

A society improves its welfare through investments. In a developed nation, the business owners need outside capital for investments because even projects of moderate sizes are beyond the reach of most wealthy individuals. Governments also need funds for public investments. Much of that money is channeled through the financial markets from savers to borrowers. In so doing, the financial markets provide a link between saving and investment.[3] As a consequence, savers can earn higher returns from their savings instead of hoarding them, borrowers can execute their investment plans to earn future profits, and both are better off. The economy also benefits by acquiring better productive capabilities as a result. Financial markets, therefore, facilitate real investments by acting as the sources of information.

A financial market typically takes its name from the borrower's side of the market: the government bond market, the municipal bond market, the mortgage market, the corporate bond market, the stock market, the commodity market, the foreign exchange (forex or FX) market,[4] the futures market, and so on [88, 653]. Within financial markets, there are two basic types of financial instruments: **debt** and **equity**. Debt instruments are loans with

---

[2] Frederich von A. Hayek (1899–1992) should be credited with his emphasis on the role of information in a market economy and competition as a discovery process.

[3] Distinction is often made between **real** investment and **financial** investment. What economists mean by investment is the sort that produces real capital formation such as plants, land, and machinery [678]. Investments in this book will be of the financial kind as opposed to the real kind mentioned above. They involve only papers such as stocks and bonds [653, 697].

[4] The foreign exchange market is the world's largest financial market where as much as US$1 trillion were traded daily as of early 1990s [588]. Players are the major commercial and investment banks with their traders connected by computers, telephones, and other telecommunication equipments [666].

a promise to repay the funds with interest, while equity securities are shares of stock in a company. Figure 1.1 quantifies the U.S. markets of debt securities. Financial markets are also often divided into two. **Money markets** concentrate on short-term debt instruments, and **capital markets** trade in long-term debt (bond) and equity instruments (stocks) [245, 309, 666, 699, 731].

Outstanding U.S. debt market securities (U.S. $ billions)

Year	Municipal	Treasury	Agency MBSs	U.S. corporate	Fed agencies	Money market	Asset–backed	Total
1985	859.5	1,360.2	372.1	719.8	293.9	847.0	2.4	4,454.9
1986	920.4	1,564.3	534.4	952.6	307.4	877.0	3.3	5,159.4
1987	1,010.4	1,724.7	672.1	1,061.9	341.4	979.8	5.1	5,795.4
1988	1,082.3	1,821.3	749.9	1,181.2	381.5	1,108.5	6.8	6,331.5
1989	1,135.2	1,945.4	876.3	1,277.1	411.8	1,192.3	59.5	6,897.6
1990	1,184.4	2,195.8	1,024.4	1,333.7	434.7	1,156.8	102.2	7,432.0
1991	1,272.2	2,471.6	1,160.5	1,440.0	442.8	1,054.3	133.6	7,975.0
1992	1,302.8	2,754.1	1,273.5	1,542.7	484.0	994.2	156.9	8,508.2
1993	1,377.5	2,989.5	1,349.6	1,662.1	570.7	971.8	179.0	9,100.2
1994	1,341.7	3,126.0	1,441.9	1,746.6	738.9	1,034.7	205.0	9,634.8
1995	1,293.5	3,307.2	1,570.4	1,912.6	844.6	1,177.2	297.9	10,403.5
1996	1,296.0	3,459.0	1,715.0	2,055.9	925.8	1,393.8	390.5	11,235.0
1997	1,366.2	3,456.8	1,827.0	2,213.6	1000.0	1,690.8	516.7	12,071.1

Figure 1.1: U.S. DEBT MARKETS 1985–1997. The Bond Market Association estimates. Sources: Federal Home Loan Mortgage Corporation, Federal National Mortgage Association, Federal Reserve System, Government National Mortgage Association, Securities Data Company, and U.S. Treasury.

It is possible that borrowers and savers can trade directly with each other; newspapers are full of classified advertisements seeking direct loans. However, in many cases, minimum size requirements, transactions costs, and costly evaluation of the security in question often prohibit direct trades. For example, many individuals cannot afford the ten thousand U.S. dollar minimum requirement to bid for the U.S. Treasury bills. Such impediments are remedied by **financial intermediaries**. They are financial institutions that act as middlemen to transfer funds from lenders to borrowers. Unlike most firms, they hold only financial assets [572]. Banks, savings banks, savings and loan associations, credit unions, pension funds, insurance companies, mutual funds, and money market funds are prominent examples. Financial intermediaries can lower the minimum investment as well as other costs for savers. Funds from many savers are pooled, for example, to buy debt instruments and equities from governments and corporations. Debts can also be combined to tailor to the clients [746].

The financial markets can be divided further into the primary market and the secondary market. The **primary market** is often merely a fiction, not a physical location. Governments and corporations initially sell securities—debt or equity—in the so-called primary market. Such sales can be done via either public offerings or private placements. A syndicate of investment banks **underwrite** the debt and equity by buying them from the issuing entities and then reselling them to the public. Sometimes, underwriting is applied to cases where the investment banker works on the **best effort** basis to avoid the risk of not being able to sell all the securities. Subsequently, people trade those instruments in the **secondary markets** such as the New York Stock Exchange. Existing securities are exchanged in the secondary market.

The existence of the secondary market makes securities more attractive to investors. It adds marketability as securities can be traded after their initial purchases. For example, the same idea is behind the creation of a secondary market in mortgages since 1970, which is achieved by asset securitization [46]. **Securitization** converts assets into traded securities with the assets pledged as collaterals. Such assets then can often be removed from the balance sheet of the bank. By making mortgages more attractive to investors, the secondary market makes them more affordable to home buyers. Here, we see financial intermediaries transform illiquid assets into liquid liabilities [746]. Securitization is also a means to hedging interest rate risk [394]. Besides mortgages, auto loans, credit card receivables, senior bank loans, and leases have all been securitized [288]. Securitization has fundamentally changed the credit market by making the capital market a major supplier of credit, a role traditionally held by the banking system.

1.4 Quantitative Analysis in Finance

Advantage used to be gained by getting data faster than competitors [270]. It has been supplemented now by superior models and programs. The availability of computers and the use of sophisticated mathematical models in finance have made analysis extremely quantitative [53]. Computers nowadays are routinely employed to generate various statistics and to corroborate proposed models. Their ability to price a security fast means quotes can be returned faster, and the decision to buy or sell are made sooner. More importantly, the confidence in such models leads to more financial innovations and deeper markets [571].

Still, judgement is called for. One has to remember that any computed results are based upon inputs and assumptions made by the models. Inputs might not be accurate enough or complete, and the assumptions are at best approximations. Computer programs are also subject to errors ("bugs"). These factors easily defeat any computation. For example, most of the theories about the stock market behavior considered in this book assume price changes satisfy the lognormal distribution. The moment that assumption is relaxed, theories based on it are at best approximations. Not only our targets are moving, but our hands are shaky as well.[5]

Although quantitative analysis is no substitute for sound judgement, it does force us to think more clearly. In trying to formulate the problems on mathematical grounds, the analyst must abstract out the important factors and sort out or hypothesize their relationships. The end result, even if not completely correct, may provide a first-order approximation to the solution. In many cases, it is not the exact number that is important but its direction

---

[5] Two Nobel laureates in economics, Merton and Scholes, helped found the hedge fund company, Long-Term Capital Management (LTCM). The firm's tools were "computers and powerful mathematics, not intuition nor inside information" [773]. The company underwent a US$3.6-billion forced bailout by 14 commercial and investment banks in September 1998.

of change. Furthermore, for complex instruments, the numerical approach is often the only option left. The insights provided by a model are ultimately what count.

1.5 Evolution of the Computer Technology

Computer hardware has been progressing at an exponential rate. Measured by the widely accepted integer SPEC benchmarks, the workstations improved their performance by 54% per year between 1987 and 1992. The memory technology is equally impressive. The **DRAM** (**Dynamic Random Access Memory**) has quadrupled its capacity every three years since 1977. Relative performance per unit cost of technologies from vacuum tube to transistor to integrated circuit to very large scale integrated circuit is a factor of 400,000 between 1951 and 1990 [622].

Some milestones in the industry include the IBM/360[6] **mainframe**, followed by Digital's **minicomputers**. (Digital was acquired by Compaq in 1998.) The year 1963 saw the first **supercomputer**, built by Seymour Cray (1926–1996) at the Control Data Corporation (CDC). Apple II of 1977 is generally considered to be the first **personal computer**. It was overtaken by the IBM Personal Computer in 1981 powered by Intel microprocessors and Microsoft's Disk Operating System (DOS) [550, 622]. The 1980s also witnessed the emergence of the so-called **massively parallel computers**, some of which sported more than 65,000 single-bit processors [440]. Parallel computers have also been applied to database applications [214, 224] and pricing complex financial instruments [465, 696, 791]. Since commodity components offer the best performance/cost ratio, personal computers connected by fast networks have the potential to uproot niche parallel machines from most of their traditional markets [18, 171].

On the software side, high-level programming languages dominate. Although they are easier to program with than low-level languages, it remains difficult to design and maintain complex software systems. In fact, in the 1960s, the software cost of the IBM/360 system already dominated its hardware cost [775]. The trend is fast moving toward **object-oriented principles** to encapsulate as much information as possible into the so-called **objects** [91, 418]. Object-oriented software development systems are widely available [20, 356].

The revolution fostered by **graphical user interface** (**GUI**) brings computer to the masses. The omnipotence of personal computers armed with easy-to-use interfaces enables employees to have access to information and to bypass several layers of management [124]. It also paves the way for the **client/server systems** [642].

Client/server systems consist of components that are logically distributed rather than centralized (see Fig. 1.2). Separate components therefore can be optimized based on their functionalities, boosting the overall performance/cost ratio. For instance, the three-tier client/server architecture contains three parts: user interface, computing (application) server, and data server [272]. Since the user interface demands fewer resources, it can run on lightly configured computers. Best of all, it can potentially be made **platform-independent**, thus offering maximum availability of the server applications, thanks to

---

[6] The IBM/360 was named "after the 360 degrees in a circle—because we intended it to encompass every need of every user in the business and the scientific worlds" [775, p. 369].

Internet-induced developments in the mid-1990s. The server machines, on the other hand, can be powerful **multiprocessors** for computing servers and machines with high disk throughputs for data servers. The object-oriented methodology and client/server architecture can be profitably combined for financial computation [538, 771].

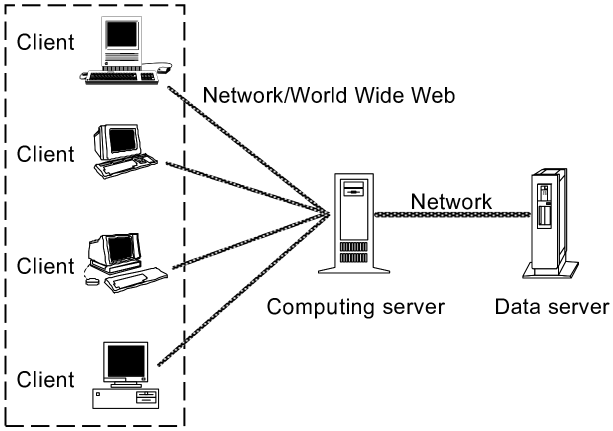

Figure 1.2: CLIENT/SERVER ARCHITECTURE. In a typical three-tier client/server architecture, client machines are connected to the computing server, which in turn is connected to the data server. As the bulk of the computation is with the computing server and the bulk of the data access is with the data server, the client computer can be lightly equipped.

Relational database was invented by Codd at IBM in the 1970s and came to dominate the database scenery with products from such companies as Oracle, Informix, Sybase, Computer Associates, and Microsoft. The incessant advancement in the capability of low-cost personal computers together with the release of truly multi-tasking operating systems for them, particularly IBM's OS/2 and Microsoft's Windows NT, brings client/server database systems to the masses [155, 188, 582, 595].

In relational database, data are organized as two-dimensional tables. Consider the following table for storing values for each day of the month, `d1`, `d2`, ... , `d31`, with the type indicated by `value_type`.

Attribute	Null?	Type
`key`	`not null`	`char(20)`
`year_month`	`not null`	`number(4)`
`value_type`	`not null`	`char(2)`
`d1`	—	`number(15,8)`
`d2`	—	`number(15,8)`
⋮	⋱	⋮
`d31`	—	`number(15,8)`

Name the table `prices`. The SQL[7] statement below can be used to retrieve the two-year U.S. Treasury yield as of December 1, 1994,

```
select d1 from prices where key='2YR'
    and year_month='9412' and value_type='3';
```

Above, `3` designates the value returned to be a yield; other numbers might be used to denote, say, prices. SQL can also be **embedded** into general-purpose programming languages such

---

[7] For **S**tructured **Q**uery **L**anguage [275], the most widely used database language. SQL is derived from SEQUEL for **S**tructured **E**nglish **QUE**ry **L**anguage, which was designed and implemented at IBM.

as **C**. By 1996, the relational database market started to feel the impact of the Internet momentum [273].

Prototyped in 1991 by Berners-Lee, the **World Wide Web** (**WWW**) is a global information system providing easy access to Internet resources [54]. It quickly sparked a revolution in the use of the Internet for communications, information, and businesses [566]. A personal computer with access to the WWW—typically through a graphical browser from Microsoft or Netscape (now part of America Online)—opens up a window to a world that can only be described as awesome: shopping information, online versions of major business and financial newspapers and magazines, stock and bond quotes, online stock trading, up-to-date and historical financial data, financial analysis software, academic research results, journal archives and preprints, to mention just a few. The WWW can also form the information network *within* corporations, or **intranet** [639]. The surge of the WWW was one of the major reasons behind the Internet's growing from fewer than 500,000 hosts to more than 10 million between 1990 and 1996 [54, 566], and in 1998, 100 million people were using the Internet [756]. Even software development strategies are fundamentally changed [441]. The release of the **Java** programming language by Sun in May of 1995 further promised genuine platform-independent software systems [310, 368, 410, 419]. Most importantly, these amazing developments are reshaping the business and financial worlds [52, 270, 415, 469, 736, 756, 762, 777].

1.6 Ethics

A market economy is no anarchy. Definite institutional arrangements must exist for it to fully function. Property rights, stable money, environment for competition, and rule of law easily come to mind.

Due to the prominent role played by financial intermediaries in market economies, the integrity of the professionals working in them are essential to the credibility of the market. In fact, without regulations to ensure the integrity of the stock markets, they would be marginal institutions [62]. As we have seen before, a functioning financial market improves the well-being of the whole society. The society, therefore, has an interest in assuring that those markets promote, rather than subvert, public goods.

Adam Smith once remarked thus [710, p. 318],

> When people seldom deal with one another, we find that they are somewhat disposed to cheat, because they can gain more by a smart trick than they can lose by the injury which it does their character.

There seems to be truth in that sociological observation. As a society becomes commercialized and people interact more, the same rationality might be applied to infer that integrity would be accepted as a better policy based on self-interest. This explains why financial firms guard their reputation most jealously, partly because they are trusted with other people's money. Warren Buffet was quoted as saying, after being appointed chairman of Salomon Brothers (now part of of Travelers Group) [241], "[I]f I hear of any employee losing Salomon one shred of reputation, I'll be ruthless." Reputable financial institutions usually have stern policies regarding their employees' activities in the markets.

These practices are "ethical" not because they achieve some absolute ideal in ethics. Rather, it is in the best interest of the firm to do so. Such behavior, like many other behaviors in a market economy, is the result of market rationality, a by-product of competition and self-interest.

1.7 Remarks on Methodologies

Some might doubt the universality of the methodologies developed by modern finance. Of course, some principles developed in this book do not seem controversial. Assuming an economic agent would prefer more profits to less, it is reasonable to infer that the payoff of, say, an option is the same in any culture; that is, it will be exercised only if it is profitable to do so. In this respect, social sciences seem easier than the physical science as there is no a priori reason a falling stone should follow $gt^2/2$ without resorting to first principles in Newtonian mechanics.

On the other hand, one might legitimately question, for example, the relevancy of modeling the stock market as a random walk to nations with less developed financial systems. One pragmatic answer may be that, as the markets become more closely linked because of globalization, the methodologies will converge regardless [268]. Such convergence can be brought about by the presence of foreign investors in the local markets, local companies trying to tap into the global capital market, and so on. In summary, theories mentioned in this book should at least become more relevant as an economy progresses.

Admittedly, economic theories often sound speculative. Many economists take the position that the goals of economics are predictive, not explanatory. They are prepared to accept a theory as long as it makes reliable predictions for the phenomena they are interested in even if its assumptions are unrealistic and it makes false predictions about other phenomena [378]. Friedman in his most influential 1953 paper [320] on the methodology in economics argued that, "Truly important and significant hypotheses will be found to have 'assumptions' that are wildly inaccurate descriptive representations of reality." Sharpe was quoted as saying [185], "The proper test of a theory is not the realism of its assumptions but the acceptability of its implications." Popper (1902–1994) was of the opinion that, to predict is to explain [630].

We shall take a pragmatic attitude. One can never do experiments in social sciences as the physicists do theirs. Neither can we watch the whole process unfold or predict its precise course and consequences [383]. Despite these difficulties, we will take comforts in the thought that, with computers, the situation might be remedied a bit because of their capability to compute with finer details than before and try out ever more scenarios. Still, this should not blind us to the fact that, deep down, finance is an "inexact" discipline. Good senses are valuable here as elsewhere.

Chapter 2

Analysis of Algorithms: a Primer

Contents

> *In computer science there is no history of critical experiments*
> *that decide between the validity of various theories,*
> *as there are in physical sciences.*
> —Juris Hartmanis [374, p. 10]

Algorithms are precise procedures that can be translated into computer programs. A classical example is Euclid's algorithm: It specifies the exact steps toward computing the greatest common divisor. Problems such as the greatest common divisor are therefore said to be **computable**. Some problems do not admit algorithms at all; they are **uncomputable**. Others may have complexity so high that no efficient algorithms exist; they are **intractable** for all practical purposes. One goal of this chapter is about ways to measure the complexity of algorithms.

Instead of using a programming language, algorithms in this book will be expressed in an easy-to-read style that can be easily transformed into programs. Our purpose is to convey the idea without getting tied up in syntax. The rules for expressing algorithms will be sketched in this chapter.

2.1 Measures of Complexity

To precisely predict the performance of a program is difficult. It depends on such diverse factors as the machine it runs on, the programming language it is written in, the compiler used to generate the machine code, the workload of the computer it runs on, and so on. Although actual running time is the only valid criterion for picking one program over the other [622], one still needs measures of complexity that are sufficiently machine-independent in order to have a grip on the *expected* performance of an algorithm.

We start with a set of basic operations which will be assumed to take one unit of time. Logical comparisons ($\leq$, $=$, $\geq$, and so on) and arithmetic operations of finite precision ($+$, $-$, $\times$, $/$, exponentiation, logarithm, and so on) are assumed to be unit-time operations. The total number of these operations is then used as a measure of the total work done by an algorithm, or its **computational complexity**. Similarly, the **space complexity** is the amount of memory space demanded by an algorithm. Since algorithms in this book do not require tremendous amount of space, we will use the term "complexity" to refer exclusively to computational complexity unless stated otherwise.

Different algorithms are compared based on their complexities. The purpose of this exercise is to concentrate on the abstract complexity of an algorithm instead of its detailed implementation, simply because implementation involves so many details that can never be fully taken into account. Although computational complexity is never a substitute for the actual running time, it should serve as a good guide to an algorithm's real-world performance.

The complexity of an algorithm is expressed as a function of the size of the input. Consider the search algorithm in Fig. 2.1. It looks for a given element by comparing it sequentially to every element in an array of length n. Apparently, the worst-case complexity is n comparisons, which occurs when the matching element is the last element of the array or when there is no match. There are other operations to be sure. The `for`-loop, for example, uses a loop variable, k, which has to be incremented by one for each execution of the loop and compared against the loop bound, n. Shall we count them too? We shall not, because it is the **order** of the complexity that is interesting. That is, we do not care much about the exact number of operations, which may be quite involved and not worthwhile to devote efforts to—its effects on real-world performance cannot be pinpointed anyway. This accepted, the complexity from maintaining the loop is subsumed by the complexity of the body of the loop.

```
Algorithm for searching an element:

input:   $x$, $n$, $A_i$ $(1 \leq i \leq n)$;
integer  $k$;
for $k = 1$ to $n$
        if [$x = A_k$] return $k$;
return not-found;
```

Figure 2.1: SEQUENTIAL SEARCH ALGORITHM.

We remark that performance is not the only criterion on which choice of algorithm is based. An efficient algorithm may incur high costs in other aspects. Take software maintainability. Simpler algorithms are often preferred to complex ones in a first attempt to make a system work even if they run more slowly [693].

2.2 Analysis of Algorithms

We are interested in **worst-case** measures. This measure is easier to obtain than other measures such as average-case measures. Of course, worst case may not occur in practice, but to assume a distribution on the input in order to perform the average-case analysis seems arbitrary unless the distribution has been certified. To suppress unnecessary details, we are concerned only with the rate of growth of the complexity required for the algorithm as the input gets larger, ignoring constant factors and small inputs. The focus is on the asymptotic growth rate, the order [31, 194]. Figure 2.2 illustrates the importance of order of growth.

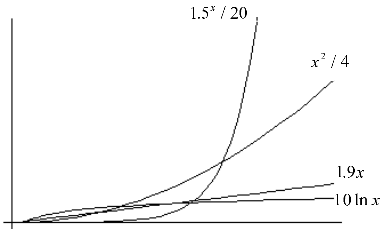

Figure 2.2: ORDER OF GROWTH OF FUNCTIONS. Plotted are four functions: $10 \times \ln x$, $1.9 \times x$, $x^2/4$, and $(1.5)^x/20$. Observe how each grows as x increases.

Let $\boldsymbol{R}$ denote the set of real numbers, $\boldsymbol{R}^+$ the set of positive real numbers, and $\boldsymbol{N} = \{0, 1, 2, \ldots\}$. The following definition lays out the notation needed to formulate complexity.

Definition 2.2.1 We say $g = O(f)$ if $\lim_{n \to \infty} g(n)/f(n) = c$ for some non-negative $c \in \boldsymbol{R}^+$, where $f, g : \boldsymbol{N} \to \boldsymbol{R}^+$. Equivalently, we say $g = O(f)$ if $g(n) \leq cf(n)$ for some non-negative c and sufficiently large n. □

Note that the base of logarithm is not important for asymptotic analysis since

$$\log_a x = \frac{\log_e x}{\log_e a} = O(\log_e x),$$

where $e = 2.71828\ldots$. We will abbreviate $\log_e x$ as $\ln x$.

Example 2.2.2 Let $f(n) = n^3$ and $g(n) = 3.5 \times n^2 + \ln n + \sin n$. Clearly, $g = O(f)$ because $g(n)$ is less than n^3 for sufficiently large n. But, $f \neq O(g)$. □

Denote the input size by N. The sequential search algorithm in Fig. 2.1 has a complexity of $O(N)$ as it has $N = n+2$ inputs and carries out $O(n)$ operations. There are several commonly accepted complexity classes. An algorithm runs in **logarithmic** time if its complexity is $O(\log N)$. Such a complexity class admits only extremely efficient algorithms since $\log N$ grows slowly. An algorithm runs in **linear** time if its complexity is $O(N)$. A complexity of $O(N \log N)$ typifies sorting and various divide-and-conquer types of algorithms. An algorithm runs in **quadratic** time if its complexity is $O\left(N^2\right)$. Many elementary matrix computations like matrix-vector multiplication have this complexity. An algorithm runs in **cubic** time if its complexity is $O\left(N^3\right)$. Matrix-matrix multiplication and solving simultaneous linear equations belong to this class. An algorithm runs in **exponential** time if its complexity is $O\left(2^N\right)$. Problems that *require* exponential time are intractable. It is possible for exponential-time algorithms to perform well on "typical" inputs.

Definition 2.2.3 We say $g = \Omega(f)$ if $\lim_{n \to \infty} \frac{g(n)}{f(n)} = \infty$, where $f, g : \boldsymbol{N} \to \boldsymbol{R}^+$. An equivalent definition can be stated as follows. We say $g = \Omega(f)$ if $g(n) \geq cf(n)$ for any non-negative c and sufficiently large n. We say $g = \Theta(f)$ if $g = O(f)$ and $g = \Omega(f)$. □

Foundations for computational complexity were laid in the 1960s [614]. Since then, several meaningful complexity classes have been identified. The difficulty of evaluating certain financial instruments, such as average-rate (Asian) options, may be due to their inherent computational complexity [148].

Exercise 2.2.1 Show that $f + g = O(f)$ if $g = O(f)$. ◇

Exercise 2.2.2 Prove the following relationships. (1) $\sum_{i=1}^{n} i = O\left(n^2\right)$; (2) $\sum_{i=1}^{n} i^2 = O\left(n^3\right)$; (3) $\sum_{i=0}^{\log_2 n} 2^i = O(n)$; (4) $\sum_{i=0}^{\alpha \log_2 n} 2^i = O\left(n^\alpha\right)$; (5) $n \sum_{i=0}^{n} \frac{1}{i} = O(n \ln n)$. ◇

Exercise 2.2.3 Let C_n satisfy the relation $C_n = C_{n/2} + n$, where n is some positive power of two and $C_1 = 1$. Show that $C_n = \Omega(n)$. ◇

2.3 Description of Algorithms

> *Probably only a person with some mathematical*
> *knowledge would think of beginning with 0*
> *instead of with 1.*
> —Bertrand Russell (1872–1970)
> *Introduction to Mathematical Philosophy*

Universally accepted mathematical symbols will be respected. So, $+$, $-$, $\times$, $/$, $<$, $>$, $\leq$, $\geq$, and $=$ mean what they signify in practice. The symbol "$:=$" denotes assignment. For example, "$a := b$" assigns the value of b to the variable a. The statement "`return` a" says a's value is returned by the program, which then exits.

The construct

```
for $i = a$ to $b$ {
    ...
}
```

means the statements within $\{\cdots\}$ will be executed $b - a + 1$ times with i equal to $a, a+1, \ldots, b$, in that order. The construct

```
for $i = a$ down to $b$ {
    ...
}
```

means the statements within $\{\cdots\}$ will be executed $b - a + 1$ times with i equal to $a, a-1, \ldots, b$, in that order. The construct

```
while [$S$] {
    ...
}
```

executes the statements within $\{\cdots\}$ until the condition, S, is violated. For example,

```
while [$a = b$] { ... }
```

runs until a is not equal to b. The construct

```
if [$S$] {
        $T_1$
} else {
        $T_2$
}
```

executes T_1 if the expression S is true and T_2 if the expression S is false. In all the above cases, the enclosing brackets "{" and "}" can be dropped if there is only a single statement within.

The construct

$$a[\,n\,][\,m\,]$$

allocates an $n \times m$ array with indices starting from *zero* (not *one*!) and arranged as

$$\boxed{\begin{array}{ccc} a[\,0\,][\,0\,] & \cdots & a[\,0\,][\,m-1\,] \\ \vdots & \ddots & \vdots \\ a[\,n-1\,][\,0\,] & \cdots & a[\,n-1\,][\,m-1\,] \end{array}}$$

The zero-based indexing scheme is more convenient in many cases. However, in other times, the one-based indexing scheme is more intuitive. So we add the construct

$$a[\,1..n\,][\,1..m\,]$$

meaning array a contains the following $n \times m$ elements,

$$a[\,1\,][\,1\,], a[\,1\,][\,2\,], \ldots, a[\,n\,][\,m-1\,], a[\,n\,][\,m\,]$$

Finally, anything following "//" shall be treated as comments without any effects.

2.4 Software Implementation

Implementation turns an algorithm into a computer program on specific computer platforms. Design, coding, debugging, and module testing are all integral parts of implementation. Software errors can be costly. They were responsible for the crash of the maiden flight of the Ariane 5 launcher on June 4, 1996 at a cost of half a billion U.S. dollars [523].

A key to a productive software project is the **reuse** of code, either from previous projects or commercial products [562]. Though a tall order, current trend toward object-oriented principles and standardization promises to further promote software reuse.

Algorithms in real software projects often have to be viewed within the context of a larger system. The overall system design might limit the choice of algorithms to only a few alternatives [693]. This constraint may arise from the requirements of other parts of the system. More often, it simply reflects the fact that most pieces of code are written for an existing system, and one cannot do much about it short of overhauling the whole system [618]. Picking an algorithm that runs counter to the global design should therefore only be attempted when the benefits sufficiently outweigh the costs.