بسم الله الرحمن الرحیم

آشنایی با الگوریتم‌ها
با رویکرد خلاقانه
(جلد اول)

دکتر رضا مرتضوی
(عضو هیئت علمی دانشگاه دامغان)
مهدی محمدی
(برنز کشوری المپیاد ریاضی ۱۳۷۹)

سرشناسه	: منبر، یودی Manber, Udi
عنوان و نام پدیدآور	: آشنایی با الگوریتم‌ها با رویکردی خلاقانه/ [تالیف یودی منبر]؛ ترجمه رضا مرتضوی، مهدی محمدی.
مشخصات نشر	: تهران: دانش‌پژوهان جوان، ۱۳۹۶.
مشخصات ظاهری	: ۲ ج.: مصور، جدول، نمودار.
شابک	: دوره: ۶-۰۷۹-۲۸۸-۶۰۰-۹۷۸ ؛ ج.۱: ۰-۰۳۹-۲۸۸-۶۰۰-۹۷۸ ؛ ج.۲: ۲-۰۷۷-۲۸۸-۶۰۰-۹۷۸
وضعیت فهرست‌نویسی	: فیپا
یادداشت	: عنوان اصلی: Introduction to algorithms : a creative approach, c1989.
یادداشت	: چاپ قبلی: احمد صادقی‌صفت، ۱۳۸۷.
موضوع	: ساختار داده‌ها
موضوع	: Data structures (Computer science)
موضوع	: الگوریتم‌های کامپیوتری
موضوع	: Computer algorithms
شناسه افزوده	: مرتضوی، رضا، ۱۳۶۲ -، مترجم
شناسه افزوده	: محمدی، مهدی، ۱۳۶۱ - ، مترجم
رده بندی کنگره	: ۱۳۹۶/QA۷۶ ۹/ ۲م۷۵س
رده بندی دیویی	: ۰۰۵/۷۳
شماره کتابشناسی ملی	: ۲۱۵۴۱۹۹

آشنایی با الگوریتم‌ها با رویکرد خلاقانه (جلد اول)

مترجمان: دکتر رضا مرتضوی -مهدی محمدی

صفحه‌آرایی: فاطمه قلی‌نژاد

چاپ اول: ۱۳۹۶

قطع: وزیری

تیراژ: ۲۴۰۰ نسخه

شابک جلد اول: ۰-۰۳۹-۲۸۸-۶۰۰-۹۷۸

شابک دوره: ۶-۰۷۹-۲۸۸-۶۰۰-۹۷۸

قیمت : ۲۱۰۰۰ تومان

ناشر کتابهای المپیاد

آدرس: میدان انقلاب، خیابان کارگر جنوبی، خیابان شهید نظری، بین خیابان منیری جاوید و ۱۲فروردین، پلاک ۱۰۵، طبقه ۳، واحد ۱۱ تلفن: ۶۶۴۹۶۳۶۳-۶۶۴۹۸۹۹۸ ۶۶۴۸۵۲۳۴ دورنگار: ۶۶۴۱۰۷۸۴

کدپستی: ۱۳۱۴۶۷۵۹۳۴ کانال ما در تلگرام: @dpjpub www.irolympiad.com

پیشگفتار مترجمین

گسترش دانش در دنیای جدید در گرو پژوهش با روش و متد علمی است. اینکه دانستن روش علم‌آموزی شرط کافی برای پژوهش علمی و بارور شدن درخت علم در هر شرایطی نیست، چندان بدیهی به نظر نمی‌آید. اما در هر حال شاید بتوان از ماده‌ی این علوم جدید برای بنای طرحی نو استفاده نمود. بی شک نهضت ترجمه که در دوره‌ای از تاریخ، مبدأ حرکتی عظیم در تاریخ علم بوده است، می‌تواند دوباره نقطه‌ی شروع دیگری برای رسیدن به افق آینده باشد. جامعه‌ی ما نیز که به هر روی مقصد خود را پیشرفت علمی و درنوردیدن قله‌های علمی قرار داده است، ناگزیر از آموختن قواعد و ابعاد مقصد خویش است. آموزش خلاق در نظام‌های آموزشی همواره دغدغه‌ی مسئولان آموزشی بوده و هست. همیشه آموزشی مورد ستایش است که علاوه‌بر آموخته‌ها، آموختن را هم به دانش‌آموزان بیاموزد. این آموزش خلاق تنها راه استقلال و مواجه با مسائل و مشکلات پیش‌رو است.

این کتاب نیز در همین زمینه گام برمی‌دارد. دغدغه‌ی نویسنده تنها آموزش انواع مختلف الگوریتم در زمینه‌های مختلف نبوده است. بلکه با یک رویکرد جدید سعی نموده به الگوی درست و روشمندی بپردازد که در طراحی همه الگوریتم‌ها استفاده می‌شود و در این راه خواننده را با خود همراه می‌کند تا او خود به کشف راه‌حل مسئله برسد. خواننده ناخودآگاه با نویسنده همراه شده و همین‌طور که پله‌پله دانش خود را می‌افزاید، قوه‌ی خلاقیت خویش را در مواجه با مشکلات و ناهمواری‌ها به‌کار می‌بندد. به این ترتیب درک او از الگوریتم نهایی قوی‌تر شده و آموزش کامل‌تری صورت می‌گیرد و همین‌طور مواجه با مسائل جدید و بدون سابقه‌ی قبلی را پیدا می‌کند. این شیوه‌ی منحصر به فرد آموزشی موجب شد که تعداد زیادی از دانش‌جویان و دانش‌آموزان المپیادی این کتاب را به عنوان مرجع اصلی خود استفاده نمایند. از همین روی ترجمه‌ی آن را لازم دیدیم. توضیح مناسب در مورد محتوای کتاب در پیش‌گفتار نویسنده آورده شده است. اما برخی نکات را در مورد ترجمه یادآور می‌شویم:

- واژه‌گزینی یکی از موارد مهم در ترجمه بود که سعی شد نزدیک‌ترین جایگزین‌ها با مشورت اهل فن و به‌کارگیری همه‌ی منابع انتخاب شود.

- سعی شده که ترجمه آزاد و تا حد ممکن روان و قابل فهم برای خواننده باشد و از ترجمه‌ی لغت به لغت پرهیز نموده‌ایم.
- بخش منابع در انتهای کتاب و بخش ارجاعات و مطالعه‌ی بیشتر در انتهای هر فصل شامل ارجاعاتی بود که اغلب آن‌ها یا منابع قدیمی بودند و یا اینکه در دسترس دانش‌آموزان و دانش‌جویان قرار نداشت. به منظور کاهش حجم کتاب و پایین آمدن هزینه‌ی آن برای خوانندگان، این بخش‌ها حذف شد.

افراد زیادی ما را در آماده‌سازی این مجموعه یاری کردند. آقایان فرشاد محتشمی، سعید و بهروز مرتضوی، وفا خلیقی و وحید دامن‌افشان و نیز ناشر محترم آقای مرتضی محمدآبادی در آماده‌سازی کتاب نقش مهمی داشتند. از ایشان و سایر عزیزانی که ذکر نام آن‌ها در این مجال نمی‌گنجد، تشکر می‌کنیم. گرچه آماده‌سازی این کتاب به منظور افزایش کیفیت آن زمان زیادی به طول انجامید، با وجود این، نتیجه خالی از خطا نیست. از خوانندگان می‌خواهیم ما را از نظراتشان محروم نفرموده و آن‌ها را به آدرس رایانامه‌ی ناشر ارسال نمایند، تا به امید خدا در ویرایش‌های بعدی لحاظ شود. همچنین با عضویت در گروه اینترنتی زیر می‌توانید از آخرین تغییرات احتمالی مطلع شوید.

http://groups.google.com/group/creative-approach

رضا مرتضوی، مهدی محمدی

پاییز ۹۶

پیشگفتار نویسنده

زمانی که به دنبال راهی برای تشریح واضح عملکرد الگوریتم‌ها برای دانش‌جویان بودم، اندیشه‌ی نگارش این کتاب به ذهنم خطور کرد. همانند بسیاری مدرسان دیگر من به این موضوع پی برده بودم که برای دانش‌جویان نه تنها حل مسائلی که به نظر من ساده است، مشکل می‌نماید، بلکه حتی برای آنان فهم راه‌حل ارائه شده برای آن مسئله هم مشکل است! من معتقدم که این دو بخش یعنی خلاقیت و تشریح به‌هم مربوطند و نباید از هم جدا در نظر گرفته شوند. دنبال کردن مراحلی که به یک راه‌حل ختم می‌شوند برای فهم کامل آن، یک اصل به نظر می‌رسد و نباید تنها به نتیجه‌ی نهایی نگریست.

این کتاب بر جنبه‌ی خلاقانه‌ی طراحی الگوریتم تأکید می‌کند و مهم‌ترین هدف آن این است که به خواننده، نحوه‌ی طراحی الگوریتم‌ها را نشان دهد. الگوریتم‌ها به شیوه‌ی «مسئله‌ی X، الگوریتم A، الگوریتم A'، برنامه‌ی P، برنامه‌ی P'» ارائه نشده‌اند. بلکه سعی شده است روش بیشتر شبیه به «مسئله X، یک الگوریتم ساده، موانع آن و سختی‌های غلبه بر این موانع، تلاش اولیه برای ارائه یک الگوریتم بهتر (که گاهی شامل برخی اشتباهات نیز می‌باشد)، بهینه‌سازی‌ها، تجزیه و تحلیل، ارتباط با سایر شیوه‌ها و الگوریتم‌ها و غیره» باشد. هدف آن نیست که الگوریتمی را ارائه دهیم که ترجمه‌ی آن برای برنامه‌نویس به یک برنامه راحت‌تر باشد، بلکه سعی شده الگوریتم‌ها به نحوی باشند که اصول پایه‌ای آن‌ها بهتر درک شوند. بنابراین الگوریتم‌ها طی یک روند کاملاً پویا مورد بررسی و توسعه قرار گرفته‌اند.

هدف ما فقط تدریس حل مسائلی خاص نیست، بلکه علاوه‌بر آن چگونگی حل مسائل جدیدی که احتمالاً در آینده بروز می‌کنند نیز مورد توجه است. تدریس چگونگی فکرکردن هنگام طراحی یک الگوریتم به اندازه‌ی تدریس جزئیات عملکرد یک راه‌حل اهمیت دارد.

برای کمک بیشتر به فرآیند تفکر که هنگام خلق یک الگوریتم لازم است، شیوه‌ی «قدیم‌ـ جدید» در طراحی الگوریتم‌های این کتاب به‌کار گرفته شده است. این شیوه علاوه‌بر اینکه بسیاری از روش‌های شناخته شده برای طراحی الگوریتم‌ها را در بر می‌گیرد، یک چارچوب فکری دقیق برای تشریح چگونگی طراحی الگوریتم را فراهم می‌آورد. به هر حال همهٔ روش‌های ممکن برای

طراحی الگوریتم را پوشش نمی‌دهد و ما هم تنها این روش را به‌کار نخواهیم برد. اساس این شیوه بر پایه‌ی مقایسه‌ی بین فرآیند عقلانی اثبات یک قضیه‌ی ریاضی با استقرا، با طراحی الگوریتم‌های ترکیبیاتی[1] است. گرچه این دو فرآیند مقاصد متفاوتی را دنبال می‌کنند و نتایج متفاوتی را به دنبال دارند، اما آن‌ها بیشتر از آنکه در نگاه اول به نظر می‌رسد، به‌هم شبیه هستند و بسیاری از مردم به این تشابه پی برده‌اند. تازگی این کتاب به میزان بهره‌گیری از این تشابه بستگی دارد. نشان می‌دهیم که این تشابه بسیاری از شیوه‌های شناخته شده را در بر گرفته و به فرآیند پویای طراحی الگوریتم کمک می‌کند. این شیوه به اختصار در فصل ۱ مورد بحث قرار گرفته است و به‌صورت مفصل‌تری در فصل ۵ مطرح می‌شود.

مثال زیر را در نظر بگیرید. فرض کنید که وارد یک شهر غریب می‌شوید، یک ماشین کرایه می‌کنید و مسیرهای منتهی به هتلتان را می‌خواهید. مطمئناً مطالبی در مورد تاریخچه‌ی شهر، شکل کلی شهر، الگوی ترافیکی و نظیر آن مورد توجه شما نیست. شما ترجیح می‌دهید راهنمایی‌هایی به‌صورت «مستقیماً به اندازه ۲ ساختمان جلو بروید، به راست بپیچید، سه کیلومتر مستقیماً به جلو بروید و مانند آن»، به شما ارائه شود. اما اگر تصمیم گرفته باشید که مدتی طولانی در آن شهر زندگی کنید، قضیه فرق خواهد کرد. گرچه می‌توانید مدتی کوتاه با آگاهی از مسیرها به فرم دوم، دوری بزنید، اما نهایتاً به اطلاعات بیشتری در مورد شهر نیاز خواهید داشت. به‌طور مشابه این کتاب نیز به‌صورت مجموعه‌ای از راهنمایی‌های ابتدایی نیست. این کتاب شامل توضیحاتی در مورد چگونگی حل بسیاری از مسائل مخصوص است، اما تأکید بر اصول و شیوه‌های کلی است. درنتیجه کتاب حالت مباحثه‌ای دارد و نیاز به درگیرشدن با مسئله و فکر دارد. من معتقدم که این کار کاملاً با ارزش است.

طراحی الگوریتم‌های کارای غیر جبری به‌صورت یک امر مهم در بسیاری از رشته‌ها از جمله ریاضیات، آمار، زیست‌شناسی ملکولی و مهندسی درآمده است. این کتاب به‌طور کلی الگوریتم‌ها و محاسبات غیر عددی را معرفی می‌کند. بسیاری از حرفه‌ای‌ها و حتی دانشمندانی که خیلی دقیق با کامپیوتر سر و کار ندارند، معتقدند که برنامه‌نویسی چیزی جز یک کار خسته کننده و غیر عُقلائی نیست. گرچه گاهی این‌طور است، اما پیامد چنین عقیده‌ای راه‌حل‌هایی ابتدایی، پیش پا افتاده، غیر کارآمد و در عین وجود راه‌حل‌هایی کاراتر و دقیق‌تر برای حل مسائل خواهد بود.

[1] combinatorial

یکی از اهداف این کتاب این است که خوانندهی خود را متقاعد کند که طراحی الگوریتمها یک رشتهی ظریف و دقیق و در عین حال مهم است.

این کتاب بهصورت خودآموز میباشد. مطالب ارائه شده عموماً بهصورت شهودی بوده و نکات تکنیکی، یا در سطح کمی قرار گرفتهاند یا جدا از بحث اصلی مطرح شدهاند. به ویژه جزئیات پیادهسازیها تا حد امکان از ایدههای طراحی الگوریتم جدا نگاه داشته شدهاند. الگوریتمهای زیادی برای مثال آورده شدهاند که هدف از طراحی آنها تأکید بر مطالب طرح شده بوده است. موضوع این کتاب ارائه مطالبی برای یادگیری و حفظ نیست، بلکه بهصورت یک سری از ایدهها، مثالهای متنوع، مثالنقضها، اصلاحات و بهینهسازیها در الگوریتمهاست. شبهکدهای مربوط به بیشتر الگوریتمها همراه با توضیحات ارائه شدهاند. تمرینهای شمارهدار و مباحثهای برای مطالعهی بیشتر در انتهای هر فصل آورده شده است. تمرینهای بیشتر فصلها به دو دستهی تقسیم شدهاند، تمرینهای مروری و تمرینهای خلاقانه. تمرینهای مروری به منظور سنجش میزان درک خواننده از مثالها و الگوریتمهای مشخصی که در آن فصل ارائه شدهاند، میباشد. تمرینهای خلاقانه هم به منظور آزمودن توانایی استفاده خواننده از روشهای شکل گرفته در ذهن وی، به علاوهی الگوریتمهای خاصی برای حل مسائل جدید، میباشد. خلاصهای از حل تمرینهای منتخب (که زیر شمارهی آنها خط کشیده شده است.)، درانتهای کتاب آورده شده است. به علاوه هر فصل شامل خلاصهای از ایدههای اساسی مطرح شده در آن نیز میباشد.

سازماندهی مطالب کتاب بهصورت زیر است. فصلهای ۱ تا ۴ موضوعات مقدماتی را ارائه میدهند. فصل ۲ به معرفی استقرای ریاضی میپردازد. استقرای ریاضی همانطور که خواهیم دید، در طراحی الگوریتمها بسیار مهم میباشد. بنابراین تجربهی کار با اثباتهای استقرایی میتواند بسیار مفید باشد. متاسفانه تعداد کمی از دانشجویان علوم کامپیوتری با اثباتهای استقرایی آشنایی کافی دارند. فصل ۲ برای برخی دانشجویان نسبتاً مشکل است. پیشنهاد میکنیم که در صورت لزوم، در مرتبهی اول مطالعه، از مثالهای مشکلتر میتوانید صرفنظر کرده و بعداً به آنها مراجعه کنید. فصل ۳ به معرفی تجزیه و تحلیل الگوریتمها میپردازد. این فصل روند کلی تحلیل الگوریتمها را تشریح کرده و ابزارهایی ابتدایی، که یک فرد برای تحلیل سادهی الگوریتم ارائه شده در این کتاب نیاز دارد، در اختیار وی میگذارد. فصل ۴ شامل معرفی

مختصری از ساختمان‌داده‌هاست. افرادی که با ساختمان‌داده‌ها اصلی آشنایی دارند و افرادی که زمینه‌ی ریاضیات خوبی دارند، می‌توانند مستقیماً از فصل ۵ شروع کنند (گرچه خواندن فصول قبلی نیز توصیه می‌شود). فصل ۵ ایده‌ی اصلی مقایسه‌ی شباهت طراحی الگوریتم با اثبات‌های استقرایی را مورد بحث قرار می‌دهد. در این فصل چندین مثال ساده از الگوریتم‌ها آورده شده و نحوه‌ی خلق آن‌ها توضیح داده شده است. اگر قرار است شما فقط یک فصل از این کتاب را بخوانید خواندن فصل ۵ را به شما توصیه می‌کنیم.

دو روش پایه برای سازماندهی یک کتاب در مورد الگوریتم‌ها وجود دارد. یک راه تقسیم کتاب بر اساس موضوع الگوریتم‌ها، مثلاً الگوریتم‌های نظریه گراف، الگوریتم‌های هندسی و غیره می‌باشد. روش دیگر تقسیم کتاب بر اساس روش‌های طراحی می‌باشد. گرچه تأکید این کتاب بر روش‌های طراحی است اما من سازماندهی کتاب را بر اساس روش اول انجام داده‌ام! فصل‌های ۶ تا ۹ الگوریتم‌هایی را در ۴ زمینه ارائه می‌دهند. الگوریتم‌های مربوط به دنباله‌ها و مجموعه‌ها (مثلاً مرتب‌سازی، مقایسه‌ی دنباله‌ها، فشرده‌سازی اطلاعات)، الگوریتم‌های گراف (مثلاً درخت‌های پوشا، کوتاهترین مسیرها، تطابق)، الگوریتم‌های هندسی (مثلاً پوشش محدب، مسئله‌ی اشتراک) و الگوریتم‌های عددی و جبری (مثلاً ضرب ماتریس‌ها، تبدیل سری فوریه). به نظر من این نوع سازماندهی برای دنبال کردن، واضح‌تر و ساده‌تر است.

فصل ۱۰ به موضوع «کاهش[۲]» اختصاص دارد. گرچه مثال‌هایی از کاهش در فصول قبلی مطرح گردیده، اما این موضوع به تنهایی چنان اهمیتی دارد که یک فصل را به خود اختصاص دهد. این فصل همچنین مقدمه‌ای برای فصل ۱۱ که در ارتباط با مسائل NP-کامل [۳] بحث می‌کند، می‌باشد. این جنبه از قضیه‌ی پیچیدگی به‌صورت یک بخش مهم از تئوری الگوریتم‌ها در آمده است. هر کس که قصد طراحی الگوریتم‌ها را دارد، باید در مورد NP-کامل و شگردهای اثبات این خاصیت اطلاع داشته باشد. فصل ۱۲ مقدمه‌ای بر الگوریتم‌های موازی ارائه خواهد داد. این فصل شامل چندین الگوریتم جالب، تحت شرایط متفاوتی از محاسبات موازی خواهد بود.

مطالب این کتاب بیشتر از آن است که بتوان آن را در یک دوره‌ی نیم‌سال ارائه کرد. درنتیجه مدرسین می‌توانند انتخاب‌های مختلفی کنند. یک دوره‌ی مقدماتی طراحی الگوریتم باید شامل

[2]reduction [3]NP-Complete

بخش‌هایی از فصل‌های ۳، ۵، ۶، ۷ و ۸ باشد. گرچه همه‌ی بخش‌ها ضروری به نظر نمی‌رسند. بخش‌های پیشرفته‌تر این فصل‌ها در طول فصل‌های ۹، ۱۰، ۱۱ و ۱۲ برای یک دوره‌ی مقدماتی، اختیاری به نظر می‌رسند و می‌توانند به عنوان پایه‌ای برای دوره‌های پیشرفته‌تر مطرح شوند.

فهرست اجمالی دو جلد

فهرست

عنوان | صفحه

فصل ۱

مفاهیم اولیه

و به شما از دانش، جز اندکی داده نشده است.

اسرا ـ ۱۷

۱.۱ مقدمه

در لغت‌نامه‌ی وبستر[1]، کلمه‌ی *الگوریتم* به این‌صورت تعریف شده است: «فرآیند حل یک مسئله‌ی ریاضی (به‌طور مثال یافتن بزرگ‌ترین مقسوم‌علیه مشترک)، طی مراحلی محدود، که عموماً نیازمند تکرار یک عمل است یا به‌صورت کلی: یک فرآیند گام به گام برای حل یک مسئله و یا رسیدن به یک هدف». ما بر تعریف دوم تأکید می‌کنیم. بنابراین تعریف طراحی الگوریتم یک رشته‌ی قدیمی است. مردم معمولاً تمایل دارند که شیوه‌های بهتری را برای رسیدن به اهدافشان بیابند، خواه برای روشن کردن آتش، خواه برای ساخت اهرام و یا مرتب‌سازی نامه‌ها. البته مطالعه‌ی الگوریتم‌های کامپیوتری مطلبی جدید است. برخی الگوریتم‌های کامپیوتری از شیوه‌هایی که قبل از اختراع کامپیوتر به وجود آمده بودند، استفاده می‌کنند. اما بیشتر مسائل نیازمند روش‌های جدید هستند. برای مثال گفتن این مطلب به کامپیوتر که: «به این تپه نگاه کن و هر گاه که ارتشی در حال نزدیک شدن بود، اعلام خطر کن!» کافی نیست. کامپیوتر باید معنای دقیق نگاه کردن، نحوه‌ی تشخیص یک ارتش، چگونگی به صدا درآوردن زنگ خطر و غیره را بداند. کامپیوتر دستوراتش را از طریق یک تعداد دستورهای خوش‌تعریف و مشخص ابتدایی دریافت می‌کند. این کار، فرآیند سختی است که دستورالعمل‌های معمولی را به زبانی که کامپیوتر آن را درک می‌کند، ترجمه کنیم. این روند اجباری، برنامه‌نویسی خوانده می‌شود که هم‌اکنون توسط میلیون‌ها نفر انجام می‌شود.

[1] Webster

البته برنامه‌نویسی کامپیوتر بیشتر از ترجمه‌ی دستوراتی مشخص به زبان قابل فهم برای کامپیوتر است. در اغلب موارد نیازمند شیوه‌های جدیدی برای حل مسائل هستیم. تنها یادگیری زبانی مرموز که به‌وسیله‌ی آن ما با کامپیوتر حرف می‌زنیم، باعث سختی برنامه‌نویسی نمی‌شود، بلکه سختی آن، دانستن آن چیزی است که باید به کامپیوتر بگوییم. کامپیوترها می‌توانند با سرعت باور نکردنی بسیاری از کارهایی را که در گذشته توسط انسان انجام می‌شده، انجام دهند. الگوریتم‌های قدیمی با ده یا شاید صدها و حداکثر هزار دستور سروکار دارند. اما کامپیوترها می‌توانند با میلیاردها و یا حتی تریلیون‌ها بیت اطلاعاتی کار کرده و میلیون‌ها عمل از اعمال پایه‌ای را در یک ثانیه انجام دهند. طراحی الگوریتم در این سطح موضوعی جدید و از بسیاری از جهات عجیب است. ما عادت کرده‌ایم به شیوه‌ای که چیزها را می‌بینیم و درک می‌کنیم، فکر کنیم. درنتیجه تمایل به طراحی یک الگوریتم ابتدایی برای استفاده داریم که البته در مسائل کوچک بسیار خوب جواب می‌دهد. متأسفانه الگوریتم‌هایی که برای مسائل کوچک به خوبی جواب می‌دهند، برای انواع بزرگ‌تر ممکن است نتایج وحشتناکی به دنبال داشته باشند. مشاهده‌ی افزایش پیچیدگی و عدم کارایی این الگوریتم‌ها هنگام به‌کارگیری در محاسبات با حجم زیاد کاملاً مشهود است.

این مسئله، جنبه‌ی دیگری نیز دارد. الگوریتم‌هایی که در زندگی روزمره‌ی خود به‌کار می‌بریم، خیلی پیچیده نیستند و قرار نیست آن‌ها را به دفعات انجام دهیم. بنابراین صرف تلاش زیاد برای ایجاد یک الگوریتم کامل، مقرون به‌صرفه نمی‌باشد و بازده‌ی آن بسیار کم است. برای مثال گشودن کیسه‌های بقالی را در نظر بگیرید. بدیهی است شیوه‌هایی غیر کارا و کارا با توجه به محتویات و یا سازماندهی آشپزخانه وجود دارد. مردم کمی حتی به این مسئله توجه می‌کنند و تعداد کمتری هم برای آن الگوریتمی درست می‌کنند. از سوی دیگر افرادی که در سطوح تجاری و با حجم بالا با بسته‌بندی و گشودن کیسه‌ها سر و کار دارند، باید شیوه‌های خوبی را برای این کار ایجاد کنند. شیوه‌های چمن‌زنی هم مثال دیگری است. می‌توانیم این عمل را از لحاظ تعداد دورها، کل زمان مصرفی یا میزان جابه‌جایی سطل‌های زباله بهبود ببخشیم. همانند حالت قبل کسی حاضر نیست یک ساعت از وقت خود را برای کشف شیوه‌ای بهتر که در آن یک دقیقه در چمن‌زنی صرفه‌جویی می‌شود قرار دهد مگر اینکه واقعاً از چمن‌زنی تنفر داشته باشد. اما از سویی دیگر کامپیوترها می‌توانند با وظایف پیچیده‌ای سر و کار داشته باشند و امکان دارد که مجبور

باشند آن وظایف را به دفعات زیاد تکرار کنند. اکنون به‌صرفه است که وقت زیادی را برای یافتن شیوه‌ای بهتر صرف کنیم، گرچه الگوریتم‌های به‌دست آمده برای فهم مشکل‌تر و پیچیده‌تر باشند. اکنون بازده کار بسیار بیشتر است (البته نباید برای کاهش زمان کل اجرای یک الگوریتم و در حد چند ثانیه، ساعت‌ها وقت برنامه‌نویسی خود را به هدر بدهیم).

این دو موضوع ــ یعنی نیاز به الگوریتم‌های غیر بدیهی برای استفاده‌های گسترده و پیچیدگی احتمالی این نوع الگوریتم‌ها ــ به مشکلات یادگیری در این حیطه اشاره دارد. در وهله‌ی اول باید بفهمیم که شیوه‌های ابتدایی بدیهی، همیشه بهترین نیستند. ادامه‌ی جستجو برای یافتن شیوه‌های بهتر، از آنجا مهم است که هم کارمان را انجام دهیم و هم شیوه‌های جدیدی یاد بگیریم. این کتاب شیوه‌های زیادی از طراحی الگوریتم‌ها را بررسی کرده و نشان می‌دهد. اما حتی یادگیری تعداد زیادی از این شیوه‌ها کافی نیست. همانند اینکه برای تبدیل شما به یک بازیکن خوب، حفظ کردن تعداد زیادی بازی شطرنج کافی نیست. یک فرد باید اصول پشت سر این شیوه‌ها را بفهمد، چگونگی به‌کار بردن آن‌ها را بداند و مهم‌تر آنکه بداند چه هنگام آن‌ها را به‌کار ببرد.

طراحی و پیاده‌سازی یک الگوریتم مشابه طراحی و ساخت یک ساختمان است. با مصالح اولیه‌ی مورد نیاز برای ساختمان آغاز می‌کنیم. وظیفه‌ی معمار ارائه‌ی یک نقشه است که این مصالح اولیه را مشخص می‌کند. وظیفه‌ی مهندس اطمینان از عملی بودن و درستی نقشه است (که خانه بعد از مدت کوتاهی فرو نریزد)، آنگاه وظیفه‌ی سازنده است که خانه را بر اساس این نقشه بنا کند. البته در طی تمام مراحل راه، هزینه‌های هر بخش باید تحلیل شده و به حساب آورده شوند. هر وظیفه‌ای جدا بوده، اما در عین حال با سایر وظایف مرتبط و به‌هم پیچیده است. طراحی الگوریتم نیز با شروع از ایده‌ها و شیوه‌هایی اولیه آغاز می‌شود. آنگاه یک طرح ساخته می‌شود که باید درستی آن را اثبات کرده و از مناسب بودن هزینه‌ی آن مطمئن شویم. مرحله‌ی پایانی، پیاده‌سازی الگوریتم روی یک کامپیوتر خاص می‌باشد. به بیان بسیار ساده، می‌توانیم این فرآیند را به چهار زیرمسئله تقسیم کنیم: طراحی، اثبات درستی، تجزیه و تحلیل و پیاده‌سازی. مانند بالا، همه‌ی این مراحل متفاوت و در عین حال به‌هم مرتبط هستند. یعنی هیچ‌کدام به تنهایی بدون توجه به سایرین قابل انجام نیستند. کمتر کسی این مراحل را به‌صورت خطی و پشت سر هم انجام می‌دهد. مشکلاتی در هر کدام از این مراحل ممکن است پیش بیاید که معمولاً نیازمند اصلاح در طراحی اولیّه است. درنتیجه نیازمند یک اثبات امکان‌پذیر دیگر، تنظیم هزینه‌ها و تغییر پیاده‌سازی می‌باشند.

این کتاب روی مرحله‌ی اول، یعنی طراحی الگوریتم‌ها تمرکز دارد. در مقایسه با سایر نوشته‌های مرتبط، می‌توانیم این کتاب را معماری الگوریتم‌ها بنامیم. گرچه معماری کامپیوتر معنیِ متفاوتی داشته و استفاده از این واژه ممکن است گیج‌کننده شود. البته کتاب همه‌ی بخش‌های دیگر را نادیده نمی‌گیرد. بحثی پیرامون درستی تجزیه و تحلیل و پیاده‌سازی الگوریتم‌ها در توضیحات اغلب الگوریتم‌ها گنجانده شده است، گاه به اختصار و گاه به دقت و تفصیل. اما تأکید بر شیوه‌های طراحی است.

این کافی نیست که فقط تعداد زیادی الگوریتم را فراگیریم تا یک معمار خوب شویم. آنچه لازم است بفهمیم، اصول طراحی است. ما شیوه‌ی متفاوتی را برای توضیح الگوریتم‌ها در این کتاب به‌کار می‌بریم. ابتدا سعی می‌کنیم خواننده را به جایی راهنمایی کنیم که راه‌حل را خودش پیدا کند. واقعاً معتقدیم که بهترین شیوه برای آموختن خلق چیزی، سعی در ساختن آن است. آن‌گاه شیوه‌ای را در طراحی الگوریتم‌ها دنبال می‌کنیم که به این فرآیند پویا و خلاق کمک می‌کند.

گرچه این شیوه، تمام راه‌های متصور برای طراحی الگوریتم‌ها را در بر نمی‌گیرد، اما برای اکثریت الگوریتم‌های این کتاب مفید است.

این شیوه بر پایه‌ی استقرای ریاضی بر پا شده است که در آن به شباهت فرآیند عقلانی اثبات یک قضیه‌ی ریاضی با طراحی یک الگوریتم ترکیبیاتی می‌پردازد. ایده‌ی اصلی در استقرای ریاضی این است که همه‌ی عبارات نیازی به اثبات از ابتدا ندارند، بلکه کافی است که صحت عبارت را از روی صحت عبارت مشابه، بر روی نمونه‌های کوچک‌تر و صحت یک یا چند عبارت کوچک‌تر اولیه نشان دهیم. مشابه همین در طراحی الگوریتم‌ها پیشنهاد می‌شود که با توسعه‌ی جواب برای مسائل کوچک‌تر به جواب مسائل بزرگ‌تر دست یابیم. اگر بتوانیم چگونگی حل مسئله را با استفاده از راه‌حل همان مسئله برای اطلاعات ورودی کوچک‌تر نشان دهیم، به هدف خود رسیده‌ایم. ایده‌ی اصلی توسعه‌ی یک راه، به‌جای ساختن آن از ابتدا است. همان‌طور که در فصل‌های آتی نشان خواهیم داد، راه‌های زیادی برای نیل به این هدف وجود دارند که منجر به روش‌های مختلف طراحی الگوریتم شوند.

اساساً از استقرای ریاضی به عنوان ابزاری برای تشریح و طراحی الگوریتم‌های سطح بالا استفاده می‌کنیم. تلاش کمی را صرف رسمی کردن یا بدیهی‌کردن این شیوه می‌کنیم. این امر توسط افرادی از جمله پاول[۲]، دایکسترا[۳]، مانا[۴]، گریس[۵]، درشویتز[۶] و سایرین انجام شده است.

[2]Paull [3]Dijkstra [4]Manna [5]Gries [6]Dershowitz [7]double induction

این کتاب تکمیل کننده‌ی این کتاب‌هاست. هدف اصلی ما آموزش بوده و طبیعی است که هرچه موضوعی بهتر تشریح شود، بهتر فهمیده می‌شود. روش‌های مختلف اثبات که بر آن تأکید داریم، شامل فرضیه‌ی استقرا، انتخاب خردمندانه‌ی دنباله‌ی استقرا، استقرای دوگانه[۷] و استقرای معکوس می‌شود. این شیوه دارای دو حسن است. نخست اینکه روش‌های ظاهراً متفاوت طراحی الگوریتم را زیر یک پرچم گرد آورده و دوم اینکه از روش‌های شناخته شده‌ی اثبات ریاضی در طراحی الگوریتم استفاده می‌کنیم. مورد دوم اهمیت ویژه‌ای دارد، زیرا راهی به منظور استفاده از روش‌های قدرتمندی می‌گشاید که برای مدتی طولانی در سایر رشته‌ها توسعه یافته‌اند.

یکی از نقاط ضعف این شیوه جامع نبودن آن است. همه‌ی الگوریتم‌ها نمی‌توانند و نباید به شیوه‌ی استقرا در ذهن طراحی شوند. گرچه اصول استقرا آن‌چنان در طراحی الگوریتم‌ها رایج است که متمرکز شدن روی آن با ارزش به نظر می‌رسد. البته روش‌های دیگر نیز در این کتاب نادیده گرفته نشده‌اند. نقد رایجی که به اکثر شیوه‌های جدید می‌شود این است که گرچه آن‌ها شیوه‌های جالبی را برای توضیح آنچه تاکنون ایجاد شده ارائه می‌دهند، اما کمکی به خلق آن‌ها نمی‌کنند. این یک انتقاد درست است، زیرا فقط آینده است که می‌تواند در مورد کارایی یک شیوه‌ی خاص و وسعت استفاده از آن قضاوت کند. من واقعاً معتقدم که استقرا نه تنها ابزاری برای توضیح الگوریتم‌ها می‌باشد، بلکه برای فهم آن‌ها نیز ضروری است. شخصاً حتی با اینکه تجارب خوبی در ایجاد الگوریتم‌هایی بدون پیروی از این شیوه داشتم اما آن را کاملاً مفید یافته و حداقل در دو مورد، این شیوه باعث شده الگوریتم‌هایی را سریع‌تر ایجاد کنم.

۲.۱ نشان‌های استفاده شده در تشریح الگوریتم‌ها

علاوه‌بر توضیح الگوریتم‌ها طی روند پویا و خلاقانه‌ی ایجاد و توسعه‌ی آن‌ها، شبه‌کدهایی را برای بسیاری از آن‌ها ضمیمه کرده‌ایم. هدف از این شبه‌کدها تکمیل توضیحات است. تلاش زیادی برای بهینه‌سازی این برنامه‌ها نکرده و توصیه نمی‌کنیم که مستقیماً همان را کپی کنید. به‌طور ساده می‌توانید از کپی آن‌ها استفاده کنید. در برخی موارد به‌صورت عمدی از آوردن بهترین نسخه‌ی برنامه خودداری کرده‌ایم، زیرا موجب پیچیدگی مضاعف می‌شد که ما را از ایده‌ی اصلی الگوریتم دور می‌کرد. گاهی موارد نیز به جزئیات پیاده‌سازی ایده‌های الگوریتم در یک برنامه نپرداخته‌ایم. این‌گونه پیاده‌سازی‌ها گاهی واضحند و البته گاهی نه! همان‌طور که گفته شد تأکید اصلی این کتاب بر اصول طراحی الگوریتم‌هاست.

[7] double induction

در بیشتر بخش‌ها از کد پاسکال (یا حتی خود پاسکال) به همراه توضیحات سطح بالایی داخل کد پاسکال، مانند «وارد جدول کن» یا «تست کن که آیا مجموعه تهی است یا خیر»، برای افزایش خوانایی استفاده می‌کنیم. البته یک استثناء قابل ذکر استفاده از کلمات begin و end برای مشخص کردن بلوک‌های برنامه است. از این عبارت‌ها فقط در ابتدا و انتهای برنامه‌ها استفاده می‌کنیم و در عوض از دندانه‌گذاری[۸] برای فاصله‌گذاری یا جدا کردن بلوک‌ها استفاده می‌کنیم. این قاعده علاوه‌بر حفظ فضا مانع ابهام می‌شود. معمولاً به تعریف دقیق متغیرهایی که تعریف آن‌ها مشخص است نمی‌پردازیم (مثلاً می‌گوییم که G از نوع گراف و یا T از نوع درخت است).

تمرین

راه‌حل تمرین‌هایی که زیر شماره‌ی آن‌ها خط کشیده شده در انتهای کتاب آورده شده است. تمرین‌های ستاره‌دار نیز از نظر مولف تاحدّ قابل توجهی از سایر تمرین‌ها مشکل‌ترند.

تمرین‌های این بخش نیاز به هیچ آگاهی قبلی از الگوریتم‌ها ندارند. آن‌ها مثال‌هایی از سؤال‌های ساده‌ی مشابه امّا با ورودی خاص هستند. از خواننده خواسته شده است که جواب‌ها را به‌صورت عادی و با کاغذ و قلم ارائه دهد. سختی کار هنگام مواجه شدن با احتمالات زیاد می‌باشد. به بیان دیگر یکی از اهداف این تمرین‌ها نمایش عدم کارایی شیوه‌های ابتدایی در مواجه با آن‌هاست. مسائل داده شده در جای مناسب در فصل‌های بعدی مورد بحث قرار می‌گیرند.

[8] Indentation

تمرین‌های مروری

۱.۱ هر یک از اعداد ۱ تا ۱۰۰ را جداگانه روی یک کارت نوشته و آن‌ها را با هم مخلوط کنید.سپس دوباره آن‌ها را مرتب کنید.

۲.۱ صد عدد زیر را روی یک کارت نوشته وسپس کارت‌ها را مرتب کنید. به تفاوت میان این تمرین و تمرین قبل فکر کنید.

۳۲۹۱۸ ۲۱۱۹۲ ۱۱۹۲۳ ۴۲۳۳ ۸۸۲۳۱ ۸۳۱۲ ۱۱ ۷۲ ۹۷۱ ۸۲۳۴ ۲۲۲۳۸ ۴۹۲۸۳
۳۲۹۵ ۲۹۳۴۷ ۳۱۰۲ ۳۲۸۸۳ ۲۰۹۳۸ ۲۹۳۰ ۱۶ ۸۲۳ ۹۲۳۴ ۹۲۳۶ ۲۹۳۷۲ ۲۲۱۸
۹۲۲۲ ۲۱۲۰۲ ۸۳۷۲۱ ۹۲۳۸ ۸۲۲۱ ۳۰۲۳۴ ۹۳۹۲۰ ۸۱۱۰۲ ۱۰۱۱ ۱۸۱۵۲ ۲۸۳۱
۲۹۱۳۳ ۹۲۲۹ ۱۰۰۳۹ ۹۲۳۵ ۴۸۳۹۵ ۲۸۳۲ ۳۷۹۲۷ ۷۳۴۹۲ ۸۴۰۲ ۴۸۲۰۱ ۳۸۰۲۴
۲۸۰۰ ۳۲۱۵۵ ۲۲۷۳ ۸۲۹۳۰ ۲۲۲۱ ۳۸۴۱ ۳۱۱ ۳۰۲۲ ۳۸۰۹۹ ۲۹۹۲۰ ۲۸۳۴۹
۷۴۲۱۲ ۷۰۱۱ ۱۸۲۳ ۹۰۳ ۲۹۹۱ ۹۳۳۵ ۲۹۱۲۳ ۲۸۹۱۰ ۲۹۲۸۱ ۳۷۷۲ ۲۰۰۱۲
۷۰۴۵۸ ۳۰۵۷۲ ۳۸۰۱۳ ۷۲۰۳۲ ۲۸۰۰۱ ۸۳۸۳۵ ۳۰۱۷ ۹۲۶۲۶ ۷۳۸۲۵ ۲۹۲۶۳
۲۰۱۷ ۲۶۲ ۸۳۶۲ ۷۷۳۰۲ ۸۵۹۳ ۳۸۲۶ ۹۳۷۴ ۲۰۰۱ ۸۳۲۶۱ ۴۸۴۰۲ ۴۸۴۵ ۷۹۷۹۴
۲۷۲۷۱ ۳۹۹۹۲ ۲۲۸۳۶ ۴۴۴ ۲۹۳۷ ۳۷۲۰۱ ۳۷۳۲۲ ۴۹۴۷۲ ۱۱۳۲۹ ۲۲۵۳

۳.۱ اعداد زیر را در نظر بگیرید. وظیفه‌ی شما این است که تا حد ممکن کمترین تعداد از این اعداد را چنان حذف کنید که اعداد باقی‌مانده صعودی باشند. برای مثال حذف همه‌ی اعداد به‌جز اعداد اولی و دومی، یک دنباله‌ی صعودی را باقی می‌گذارد. پاک کردن همه‌ی اعداد به‌جز اولین، سومین، ششمین و هشتمین عدد، نتیجه‌ی مشابهی را به دنبال دارد، اما اعداد کمتری را حذف کرده‌ایم.

۹ ۴۴ ۳۲ ۱۲ ۷ ۴۲ ۳۴ ۹۲ ۳۵ ۳۷ ۴۱ ۸ ۲۰ ۲۷ ۸۳ ۶۴ ۶۱ ۲۸ ۳۹ ۹۳ ۲۹
۱۷ ۱۳ ۱۴ ۵۵ ۲۱ ۶۶ ۷۲ ۲۳ ۷۳ ۹۹ ۱ ۲ ۸۸ ۷۷ ۳ ۶۵ ۸۳ ۸۴ ۶۲ ۵ ۱۱
۷۴ ۶۸ ۷۶ ۷۸ ۶۷ ۷۵ ۶۹ ۷۰ ۲۲ ۷۱ ۲۴ ۲۵ ۲۶

۴.۱ تمرین ۳.۱ را چنان حل کنید که دنباله‌ی اعداد باقی‌مانده به‌صورت نزولی مرتب شود.

۵.۱ فرض کنید که در کشوری سکه‌هایی به ارزش ۱۵، ۲۳، ۲۹، ۴۱ و ۶۷ (همگی به سِنت) وجود دارند. ترکیبی از این سکه‌ها را چنان بیابید که مجموع آن‌ها ۱۸ دلار و هشت سنت (۱۸۰۸ سنت) بشود. از هر سکه به تعداد کافی در اختیار دارید.

۶.۱ فرض کنید ورودی زیر، لیستی از اعداد صحیح به‌صورت زوجِ مرتب است. زوج مرتب (x, y) به معنای آن است که x منتظر جوابی از y است. هنگامی که x منتظر است نمی‌تواند هیچ کار دیگری انجام دهد و به ویژه به سوال دیگران که منتظر جواب او هستند، نمی‌تواند پاسخ دهد.

مسئله یافتن دنباله‌ای از زوج مرتب‌های $(x_k, x_1), (x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k)$ به‌ازای $k > 1$ است (هر kای قابل قبول است) در صورتی که چنین دنباله‌ای یافت شود ما به یک بن‌بست[9] رسیده‌ایم. چون هیچ‌کس در آنجا که منتظر جواب دیگری است نمی‌تواند کاری کند. می‌توانید از کاغذ و قلم برای محاسبات عددی خود (مانند مقایسه و ساخت جدول) استفاده کنید. اما اجازه رسم شکل را ندارید. (ممکن است شما شکلی نامربوط به این ورودی خاص بکشید که به شما کمک کند مسئله را در حالت کلی حل کنید.)

(۱, ۱۶) (۲, ۲۱) (۲, ۲۵) (۲, ۲۲) (۲۳, ۵۰) (۲۳, ۴۷) (۲۴, ۱) (۲۵, ۱۰) (۳۵, ۷)
(۳۶, ۴۵) (۳۶, ۳۷) (۳۸, ۴۲) (۳۹, ۴۱) (۱۲, ۳۷) (۱۲, ۲۳) (۱۲, ۳) (۱۲, ۲۰)
(۱۴, ۲۵) (۴۱, ۹) (۴۲, ۳) (۴۳, ۵) (۴۳, ۲۲) (۲۹, ۲) (۳۰, ۴۸) (۳۱, ۱۵) (۳۲, ۱۷)
(۶, ۴۵) (۶, ۱) (۵, ۳۵) (۵, ۲۰) (۵, ۲۸) (۵, ۱۱) (۴۸, ۴) (۴۸, ۱۰) (۴۹, ۳۲)
(۷, ۳۱) (۷, ۴) (۵, ۳۳) (۶, ۲۹) (۶, ۱۲) (۶, ۱۱) (۶, ۳) (۶, ۱۷) (۴۵, ۲۷)
(۴۷, ۳۴) (۴۸, ۲۰) (۷, ۴۰) (۷, ۳۴) (۸, ۱۱) (۹, ۱۹) (۱۱, ۳۰) (۱۱, ۴) (۱۱, ۲۲)
(۱۱, ۲۵) (۲۰, ۲۴) (۲۱, ۲۳) (۲۱, ۴۶) (۲۲, ۴۷) (۲۳, ۴۹) (۳, ۳۹) (۳, ۳۴) (۴, ۱۴)
(۴, ۳۷) (۵, ۴۲) (۵, ۸) (۱۵, ۲) (۱۵, ۵۰) (۱۵, ۴) (۱۵, ۳۷) (۱۶, ۱۳) (۱۷, ۳۸)
(۱۸, ۲۸) (۱۹, ۸) (۲۶, ۱۵) (۲۶, ۴۲) (۲۷, ۱۸) (۲۸, ۳۵) (۱۳, ۳۶) (۱۳, ۵۰)
(۱۳, ۳۴) (۱۳, ۲۲) (۲۹, ۳۴) (۲۹, ۳۸) (۲۹, ۳۰) (۲۹, ۱۶) (۴۴, ۳۳) (۴۴, ۳۶)
(۴۴, ۷) (۴۴, ۳) (۴۴, ۳۲) (۴۴, ۲۱) (۳۳, ۹) (۳۳, ۲۱) (۳۳, ۳۵) (۳۳, ۱۹) (۳۳, ۴۱)
(۲۶, ۱۰) (۲۶, ۴۴) (۲۶, ۱۶) (۲۶, ۳۹) (۲۶, ۱۷)

۷.۱ ورودی این مسئله یک جدول دو بعدی ۱۵ × ۱۵ است که در شکل ۱.۱ نشان داده شده است. به‌ازای i دلخواه سطر i اُم و ستون i اُم مربوط به یک مکان است. هر خانه از جدول فاصله‌ی مستقیم بین مکان‌های منطبق بر سطر و ستون آن خانه را نشان می‌دهد. نماد «ـ» عدم وجود راه مستقیم بین این دو مکان را نشان می‌دهد. راه مستقیم ممکن است نزدیک‌ترین مسیر بین آن دو مکان نباشد. ممکن است با استفاده از یک مکان سوم (یا چند مکان واسطه) مسیر کوتاه‌تری وجود داشته باشد. برای مثال کوتاه‌ترین مسیر بین ۱ و ۶، از ۵ و ۱۲ می‌گذرد. کوتاه‌ترین مسیر بین ۱ و ۱۵، بین ۴ و ۳ و بین ۱۵ و ۸ را بیابید.

۸.۱ جدول شکل ۱.۱ را در نظر بگیرید و کوتاهترین مسیر بین مکان ۵ و سایر مکان‌ها را به‌دست بیاورید.

۹.۱ گراف شکل ۲.۱ را در نظر بگیرید. یک مسیر بسته از یال‌ها در گراف چنان بیابید که از تمام رئوس گراف دقیقاً یک‌بار عبور کند (این مسیر به نام دور همیلتونی معروف بوده و در بخش ۱۲.۷ بیشتر بررسی می‌شود.)

[9] deadlock

۱۰.۱ این مسئله یک **maze** معمولی است. با این تفاوت که نمایش آن به‌جای حالت تصویری

	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۰	۲	۳	–	۱	۹	۶	۲	۱	۷	۴	۲	۸	۳	–
۲	۷	۰	۲	–	–	–	–	۲	۱	۶	۹	۱	۷	۲	۸
۳	۸	–	۰	۸	۹	۳	۶	۸	۵	۷	–	۸	–	۳	–
۴	–	۸	–	۰	–	۵	۴	–	–	۱	۱	۹	–	۸	–
۵	۹	–	۸	–	۰	۳	۲	۷	۵	۸	–	۱	–	۴	۲
۶	۳	۲	–	۳	۶	۰	۵	۳	۲	–	۸	۷	۲	–	۸
۷	۲	–	–	۲	۸	–	۰	۶	۲	–	۸	۸	۲	–	۴
۸	۱	۱	–	–	۲	۳	۸	۰	–	۱	۱	–	۲	۷	–
۹	۴	–	۹	–	۲	۹	–	۲	۰	۴	۹	۳	–	–	–
۱۰	–	–	–	–	۱	۸	–	۷	۱	۰	۳	–	–	–	۲
۱۱	۳	۸	۷	۱	–	–	۳	۸	–	–	۰	۲	۹	۲	۱
۱۲	۳	–	۱	۲	۸	۱	۱	–	۵	۱	۹	۰	۲	–	۹
۱۳	۷	–	۳	۱	۶	–	–	۲	–	۳	–	۹	۰	۲	–
۱۴	۲	۹	۶	–	۷	–	۹	–	۳	–	۱	۱	۹	۰	–
۱۵	۲	۹	۲	۱	–	–	۱	–	۴	۳	۶	۵	۱	–	۰

شکل ۱.۱ جدول مربوط به تمرین‌های ۷.۱ و ۸.۱

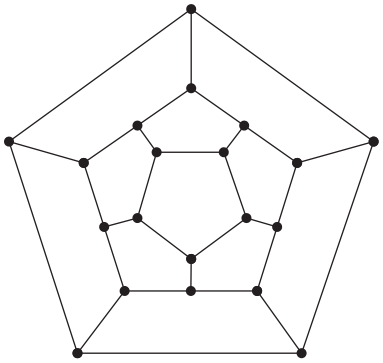
به‌صورت عددی است. یک مستطیل با ۱۱ سطر و ۱۱ ستون (از شماره‌ی ۰ تا ۱۰) شماره‌گذاری شده است. قرار است که با شروع از نقطه‌ی (۰,۰) و با حرکت‌های چپ، راست، بالا و پایین، مسیری تا نقطه‌ی (۱۰,۱۰) طی کنیم. نقاط پی‌آمد موانعی هستند که نمی‌توانید از آن‌ها عبور کنید.

(۹،۳) (۶،۳) (۱،۳) (۰،۸) (۲،۴) (۸،۴) (۵،۹) (۲،۸) (۷،۰) (۶،۶) (۳،۲)
(۶،۱۰) (۶،۲) (۱۰،۵) (۵،۶) (۴،۵) (۲،۲) (۷،۸) (۴،۲) (۳،۷) (۳،۰) (۱،۹)
(۲،۶) (۱۰،۹) (۹،۲) (۸،۷) (۴،۴) (۵،۷) (۸،۱) (۷،۹) (۷،۵) (۴،۰)

الف: مسیری را از نقطه‌ی شروع تا مقصد بیابید که از هیچ‌یک از نقاط فوق عبور نکند.

ب: سعی کنید کوتاه‌ترین مسیر را از نقطه‌ی شروع تا مقصد بیابید که از موانع فوق عبور نکند.

۱۱.۱ بزرگ‌ترین مقسوم‌علیه مشترک دو عدد ۲۲۵۲۷۷ و ۱۷۸۷۹۴ را بیابید (طبق تعریف بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح (که همگی آنها ۰ نیستند)، بزرگ‌ترین عدد مثبتی است که هر دوی آن اعداد بر آن بخش‌پذیر باشند).



شکل ۲.۱ معمای همیلتون

۱۲.۱ مقدار $۲^{۶۴}$ را محاسبه کنید. راهی برای کاهش تعداد ضرب‌ها بیابید.

۱۳.۱ لیست زیر نشان‌دهنده‌ی آراء الکترال انتخاباتی از هر کدام از ایالت‌های ایالات متحده در انتخابات ریاست جمهوری ۱۹۸۶ می‌باشد. کاندیدایی که اکثریت آراء یک ایالت را کسب کند، تمامی آراء الکترال داده شده از آن ایالت به نفع او حساب می‌شود. در انتخابات ریاست جمهوری ایالات متّحده تنها ۲ نامزد وجود دارند. در کل ۵۳۸ رأی وجود دارد. مشخص کنید آیا از نظر ریاضی ممکن است انتخابات به حالت تساوی برسد یا نه؟ (این مسئله را به اسم مسئله‌ی تقسیم‌بندی[۱۰] می‌شناسند که حالتی خاص از کوله‌پشتی[۱۱] است که در بخش ۱۰.۵ روی آن بحث خواهد شد.)

[10] partition problem [11] knapsack problem

۷	Arizona	۳	Alaska	۹	Alabama
۸	Colorado	۴۷	California	۶	Arkansas
۲۱	Florida	۳	Delaware	۸	Connecticut
۴	Idaho	۴	Hawaii	۱۲	Georgia
۸	Iowa	۱۲	Indiana	۲۴	Illinois
۱۰	Louisiana	۹	Kentucky	۷	Kansas
۱۳	Massachusetts	۱۰	Maryland	۴	Maine
۷	Mississippi	۱۰	Minnesota	۲۰	Michigan
۵	Nebraska	۴	Montana	۱۱	Missouri
۱۶	New Jersey	۴	New Hampshire	۴	Nevada
۱۳	North Carolina	۳۶	New York	۵	New Mexico
۸	Oklahoma	۲۳	Ohio	۳	North Dakota
۴	Rhode Island	۲۵	Pennsylvania	۷	Oregon
۱۱	Tennessee	۳	South Dakota	۸	South Carolina
۳	Vermont	۵	Utah	۲۹	Texas
۳	Washington, D.C.	۱۰	Washington	۱۲	Virginia
۳	Wyoming	۱۱	Wisconsin	۶	West Virginia

فصل ۲

استقرای ریاضی

به راستی که تو به عقل خود سنجیده می‌شوی، پس آن را با علم پرورش ده.

حضرت علی(ع)

۱.۲ مقدمه

در فصل‌های آتی شاهد خواهیم بود که استقرا نقش مهمی در طراحی الگوریتم‌ها بازی می‌کند. در این فصل به‌طور مختصر به معرفی استقرای ریاضی با کمک مثال می‌پردازیم. این مثال‌ها از آسان تا نسبتاً مشکل هستند. خوانندگانی که اثبات‌های استقرایی زیادی را تا به حال ندیده‌اند ممکن است این فصل را اندکی مشکل بیابند. ادعا می‌کنیم که فرایند ساخت اثبات‌ها و ساخت الگوریتم‌ها مشابهند و درنتیجه کسب تجربه در مورد اثبات‌های استقرایی بسیار مفید است.

استقرای ریاضی یک روش بسیار قدرتمند اثبات است و معمولاً به‌صورت زیر عمل می‌کند. فرض کنید T قضیه‌ای است که به دنبال اثبات آن هستیم. همچنین فرض کنید که T شامل پارامتر n است که می‌تواند هر عدد طبیعی باشد. به‌جای اثبات اینکه T برای تمام مقادیر n درست است، دو مورد زیر را اثبات می‌کنیم:

۱. T برای $n = ۱$ درست است.

۲. برای هر $n > ۱$، اگر T برای $n - ۱$ درست باشد آن‌گاه T برای n درست است.

علت کافی بودن دو شرط فوق کاملاً واضح است. از شرط ۱ و ۲ مستقیماً درستی T برای $n = ۲$ نتیجه می‌شود. اگر T برای $n = ۲$ درست باشد می‌توان از شرط ۲ نتیجه گرفت که T برای $n = ۳$ نیز درست است و به همین ترتیب تا آخر. خود استقرای ساده آن‌چنان واضح است که معمولاً اثبات نمی‌شود و معمولاً از آن به عنوان یک اصل در تعریف اعداد طبیعی یاد می‌گردد.

معمولاً شرط ۱ به‌راحتی اثبات می‌شود و اثبات شرط ۲ نیز در بسیاری از موارد ساده‌تر از اثبات قضیه به‌طور مستقیم است، زیرا می‌توانیم از این فرض که T برای $n-1$ درست است، استفاده کنیم. این فرض، فرضِ استقرا نامیده می‌شود. گاهی اوقات فرض را به‌صورت آگاهانه انتخاب می‌کنیم. (یعنی به‌جای $n-1$ مقداری کوچک‌تر را درست فرض می‌کنیم) کافی است که قضیه را «کاهش» داده و آن را برای مقادیر کوچک‌تر از n بررسی کنیم. به‌جای اثبات آن از اول، روی این کاهش تأکید می‌کنیم. اجازه دهید با یک مثال آغاز کنیم:

قضیه ۱.۲ *برای همه‌ی اعداد طبیعی x و n، x^n-1 بر $x-1$ قابل قسمت است.*

برهان. اثبات با کمک استقرا بر روی n انجام می‌شود. بدیهی است قضیه برای $n=1$ درست است. حالا فرض می‌کنیم قضیه برای $n-1$ درست است، یعنی فرض می‌کنیم که $x^{n-1}-1$ بر $x-1$ به‌ازای تمام مقادر طبیعی x قابل قسمت است. اکنون باید ثابت کنیم که x^n-1 بر $x-1$ قابل قسمت است. اکنون ایده‌ی کلی، سعی بر نوشتن عبارت x^n-1 برحسب $x^{n-1}-1$ است، که قبلاً بنا به فرض استقرا بر $x-1$ قابل قسمت بوده است. می‌توان نوشت:

$$x^n-1=x(x^{n-1}-1)+(x-1)$$

که در آن و در قسمت راست تساوی عبارت اوّل بنا به فرض استقرا بر $x-1$ بخش‌پذیر بوده و عبارت دوم نیز خود $x-1$ است. □

استقرای ساده به‌صورت زیر بیان می‌شود:

> اگر عبارت P با پارامتر n برای $n=1$ درست باشد و اگر برای هر $n>1$ درستی عبارت P برای $n-1$ نتیجه بدهد که P برای n هم درست است، آنگاه P برای تمام مقادیر طبیعی n درست است.

گاهی به‌جای به‌کار بردن $n-1$ و n، از n و $n+1$ استفاده می‌کنیم که این دو به‌طور کامل با هم معادلند.

> اگر عبارت P با پارامتر n برای $n=1$ درست باشد و اگر برای هر $n\geq 1$ درستی عبارت P برای n نتیجه بدهد که P برای $n+1$ هم درست است، آنگاه P برای تمام مقادیر طبیعی n درست است.