دانشگاه هرمزگان

اصول

ساختمان داده‌ها

سیمور لیپ شوتز / مهندس حسین ابراهیم زاده قُلزم

چاپ هشتم

قابل استفادهٔ دانشجویان مهندسی کامپیوتر

سخت‌افزار، نرم‌افزار، ریاضی کاربردی

ناشر برگزیده سال ۱۳۷۷

حاوی ۴۵۷ مساله حل شده

بسم الله الرحمن الرحیم

اصول ساختمان داده‌ها

نوشتهٔ : سیمور لیپ شوتز

ترجمهٔ : مهندس حسین ابراهیم‌زادهٔ قُلزم

لیپشوتز، سیمور Lipschutz, Seymour
اصول ساختمان داده‌ها/ نوشته سیمور لیپ شوتز؛ ترجمه حسین ابراهیم‌زاده قلزم - [بندرعباس]: دانشگاه هرمزگان، ۱۳۷۴.
(۸)، ۵۰۱ ص: مصور، جدول، نمودار.

فهرستنویسی براساس اطلاعات فیپا (فهرستنویسی پیش از انتشار).
عنوان روی جلد: اصول ساختمان داده‌ها؛ قابل استفاده دانشجویان مهندسی کامپیوتر، سخت‌افزار، ریاضی کاربردی، حاوی ۴۵۷ مسأله حل شده.
عنوان اصلی: Schaums outline of theory and problems of data structures.

واژه‌نامه.
چاپ هشتم: ۱۳۸۲
ISBN: 964 - 6426 - 35 - 2: ۳۴۰۰۰ ریال
۱. ساختار داده‌ها. ۲. ساختار داده‌ها -- مسائل، تمرینها و غیره. الف. ابراهیم‌زاده قلزم، حسین، ۱۳۴۰ -، مترجم. ب. دانشگاه هرمزگان. ج. عنوان.
QA۷۶/۹/ل۲س۹ ۰۰۵/۷۳
۱۳۷۴ م۷۵-۱۰۵۸۱

نام کتاب: اصول ساختمان داده‌ها
ناشر: دانشگاه هرمزگان
مؤلف: سیمور لیپ شوتز
مترجم: مهندس حسین ابراهیم‌زاده قلزم
لیتوگرافی: کبریا
چاپ: کبریا
نوبت چاپ: هشتم/۱۳۸۲
شمارگان: ۳۳۰۰ نسخه
قیمت: ۳۴۰۰۰ ریال

شابک: ۲ - ۳۵ - ۶۴۲۶ - ۹۶۴ ISBN: 964 - 6426 - 35 - 2

«حق چاپ محفوظ است»

بندرعباس - صندوق پستی ۳۹۹۵ - دانشگاه هرمزگان تلفن: (۵ - ۴۰۱۲۱ - ۰۷۶۱)
فروشگاه: تهران - خیابان انقلاب - مقابل دانشگاه تهران - پاساژ فروزنده - تلفن: ۶۹۵۰۵۵۳

مقدمه مترجم

سالهاست که به ترجمه و تألیف کتابهای ریاضی و کامپیوتری اشتغال دارم که کاری است پررنج، با مشقتهای فراوان. اما هیچ ترجمه و تالیفم را شیرین‌تر و جذابتر از اثر حاضر ندیده‌ام زیرا کامپیوتر را علم پرفسونی می‌دانم که معتقدم فسون آن در ساختمان داده‌ها متجلی است. ساختمان داده‌ها یکی از زیباترین مباحث و نیرومندترین ابزار در حل مسائل کامپیوتری است. این درس شیفتگان بسیار در میان دانشجویان رشتهٔ کامپیوتر دارد و نظر به اهمیت آن به عنوان درس رسمی دانشگاههای کشور پذیرفته شده است.

کتاب حاضر دربرگیرندهٔ تمامی مطالب ساختمان داده‌های رشته‌های مهندسی کامپیوتر ـ سخت‌افزار، نرم‌افزار و ریاضی کاربردی است. این کتاب در عین حال که به تئوری پرداخته اما عاری از لفاظی و توصیفهای ناضرور است. از کمتر نکته‌ای در آن چشم پوشیده شده است و آنچه در کتاب است، ساختمان داده‌ها است و بس. این کتاب را می‌توان به عنوان سرفصل چندین واحد درسی دانشجویان رشته‌های کامپیوتر و ریاضی کاربردی مورد استفاده قرار داد و می‌تواند مرجعی مناسب و مطمئن برای درسهای برنامه‌نویسی پیشرفته، ساختمان داده‌ها و الگوریتم‌ها، ساختمان و ریاضیات گسسته، ذخیره و بازیابی اطلاعات و زبانهای برنامه‌نویسی باشد.

در ترجمهٔ این کتاب بر کسی منتی نمی‌گذارم و خویشتن را سزاوار هیچگونه پاداشی نمی‌دانم زیرا به ایران و مردم ایران مدیونم و اگر بتوانم از این رهگذر اندکی از دین معنوی خود بکاهم تا حدی به آرزوی خود رسیده‌ام.

حسین ابراهیم‌زادهٔ قُلزم
دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
تهران ۲۰ ـ مهر ـ ۱۳۷۴

پیشگفتار مؤلف

مطالعه و بررسی ساختمان داده‌ها جزء اصلی و اجباری برنامهٔ هر دانشجوی لیسانس و فوق‌لیسانس علم کامپیوتر است. این کتاب که اساسی‌ترین مطالب و سرفصل‌های درس ساختمان داده‌ها را ارائه می‌دهد، می‌تواند به عنوان یک کتاب درسی برای یک درس رسمی ساختمان داده‌ها یا به عنوان یک کتاب مکمل برای تقریباً تمام کتب متعارف و موجود کامپیوتر در این زمینه مورد استفاده قرار گیرد.

فصل‌های کتاب عمدتاً براساس میزان پیچیدگی مطالب تنظیم شده‌اند. *فصل ۱*، معرفی و مرور کلی بر مطالب تمام کتاب خواهد داشت و *فصل ۲*، زمینهٔ ریاضی و نمادهای کامپیوتری را جهت ارائه و تجزیه و تحلیل الگوریتم‌ها فراهم می‌آورد. *فصل ۳*، که در مورد تطبیق الگوست، فصل مستقلی است که به صورت حاشیه‌ای با متن ارتباط دارد و از این رو در مطالعه اول کتاب می‌توان این فصل را حذف کرد یا خواندن آن را تا زمان دیگر به تعویق انداخت. *فصل‌های ۴* تا ۸ شامل مطالب اصلی درس ساختمان داده‌هاست. مخصوصاً این که در *فصل ۴*، آرایه‌ها **Arrays**، رکوردها **Records** مورد بررسی قرار می‌گیرد. *فصل ۵* راجع به لیستهای پیوندی **Linked Lists** است. *فصل ۶* حاوی مطالبی در زمینهٔ پشته‌ها **Stacks** و صفها **Queues** و همچنین زیربرنامه‌های بازگشتی **Recursion** است، *فصل ۷* در مورد درختهای دودویی **Binary Trees** و *فصل ۸* در بارهٔ گرافها **Graphs** و کاربردهای آن است. اگرچه مرتب کردن **Sorting** و جستجوی اطلاعات **Searching** از ابتدا تا انتهای کتاب در داخل مجموعه‌ای از ساختمان دادهٔ مخصوص مانند جستجوی دودویی با آرایه‌های خطی، **QuickSort** با پشته‌ها و صفها، و **HeapSort** با درختهای دودویی، مورد بحث و بررسی قرار گرفته است اما در *فصل ۹*، که آخرین فصل کتاب را نیز تشکیل می‌دهد الگوریتم‌های اضافی و تکمیلی مرتب کردن اطلاعات و جستجوی آنها نظیر ادغام و مرتب کردن **Merge - Sort** و درهم سازی **Hashing** معرفی و بحث و بررسی شده است.

الگوریتم‌ها به صورتی ارائه شده‌اند که مستقل از نوع کامپیوتر و زبان‌های برنامه‌نویسی است. علاوه

براین، این الگوریتم‌ها برای دستورهای کنترلی، عمدتاً با استفاده از قطعه برنامه‌های IF- THEN - ELSE و REPEAT - WHILE نوشته شده‌اند و همچنین جهت سهولت در خواندن و درک و فهم سادهٔ آنها، از الگوی تورفته و پله‌ای استفاده شده است. بنابراین، هر یک از الگوریتم‌های کتاب را تقریباً می‌توان به سادگی، به تمام زبانهای متعارف و موجود برنامه‌نویسی ترجمه کرد.

این کتاب با اتخاذ یک روش کاملاً ساده و مقدماتی و با ارائه مثال و نمودارهای متعدد، خوانندگان زیادی را بخود اختصاص خواهد داد و به ویژه به عنوان راهنمایی خودآموز، کاملاً مؤثر و مفید عمل خواهد کرد.

هر فصل از کتاب با بیان روشنی از تعاریف، اصول و عبارتهای توصیفی همراه است. بدنبال این مراحل، مجموعه‌های درجه‌بندی شده از مسایل حل شده و تکمیلی می‌آیند. مسأله‌های حل شده، مطالب درسی را توضیح می‌دهند و نکات ظریفی را مطرح می‌سازند و در جهت تقویت آنها است در حالی که مسایل تکمیلی مروری کامل بر مطالب درسی این سرفصل‌ها دارند.

مایلم مراتب سپاسگزاری خود را از دوستان و همکاران بیشماری که به لحاظ ارائه پیشنهادات ارزنده‌شان و بازنگری انتقادی دست‌نوشته‌ها که وقت خود را مصروف آن نموده‌اند ابراز دارم. علاوه بر این مایلم از کارکنان نشر Mc Graw - Hill Schaum به ویژه آقای Jeffrey Mccartney به خاطر همکاری سودمندشان سپاسگزاری بنمایم.

در پایان از آقای دانلد. ا. کناث Donald. E. Knuth که متن جامع اولیهٔ ساختمان داده‌ها را تدوین نموده‌اند و بدون شک تأثیر بسزایی در تألیف این کتاب داشته‌اند و همچنین موضوعات متعدد دیگر در این‌خصوص که از ناحیهٔ ایشان صورت پذیرفته است صمیمانه تشکر می‌نمایم.

سیمور لیپ شوتز
دپارتمان کامپیوتر
دانشگاه تِمپل

فهرست مندرجات

صفحه

فصل ۱ معرفی و مرور سریعِ مطالب کتاب

۱-۱ مقدمه ۱
۱-۲ اصطلاحات اصلی و سازماندهی ابتدایی داده‌ها ۱
۱-۳ ساختمان داده‌ها ۳
۱-۴ عملیات بر روی ساختمان داده‌ها ۱۲
۱-۵ الگوریتم‌ها: پیچیدگی، توازن بین زمان اجرا و حافظه ۱۴
۱-۶ مسأله‌های حل شده ۱۷

فصل ۲ مفاهیم مقدماتی

۲-۱ مقدمه ۲۷
۲-۲ نمادگذاری ریاضی و تابعهای کامپیوتری ۲۸
۲-۳ نمایش الگوریتمی ۳۳
۲-۴ دستورهای کنترلی ۳۶
۲-۵ پیچیدگی الگوریتم‌ها ۴۳
۲-۶ زیرالگوریتم‌ها ۴۷
۲-۷ متغیرها و انواع داده‌ها ۵۰
۲-۸ مسأله‌های حل شده ۵۲
۲-۹ مسأله‌های تکمیلی ۶۰

فصل ۳ پردازش رشته‌ها

۳-۱ مقدمه ۶۳

۲ـ۳ اصطلاحات پایه‌ای ۶۴
۳ـ۳ ذخیرهٔ رشته‌ها ۶۵
۴ـ۳ نوع دادهٔ کاراکتری ۷۱
۵ـ۳ عملیات بر روی رشته‌ها ۷۳
۶ـ۳ پردازش رشته ۷۶
۷ـ۳ الگوریتم‌های تطبیق الگو ۸۲
۸ـ۳ مسأله‌های حل شده ۸۹
۹ـ۳ مسأله‌های تکمیلی ۹۹

فصل ۴ آرایه‌ها، رکوردها و اشاره‌گرها

۱ـ۴ مقدمه ۱۰۳
۲ـ۴ آرایه‌های خطی ۱۰۴
۳ـ۴ نمایش آرایه‌های خطی در حافظه ۱۰۷
۴ـ۴ پیمایش آرایه‌های خطی ۱۰۹
۵ـ۴ اضافه کردن و حذف کردن عنصر ۱۱۰
۶ـ۴ مرتب کردن، مرتب کردن حبابی ۱۱۳
۷ـ۴ جستجو کردن، جستجوی خطی ۱۱۷
۸ـ۴ جستجوی دودویی ۱۲۱
۹ـ۴ آرایه‌های چند بعدی ۱۲۵
۱۰ـ۴ اشاره‌گرها، آرایه‌های نوع اشاره‌گر ۱۳۳
۱۱ـ۴ رکوردها، ساختارهای رکوردی ۱۳۹
۱۲ـ۴ نمایش رکوردها در حافظه، آرایه‌های موازی ۱۴۲
۱۳ـ۴ ماتریسها ۱۴۵
۱۴ـ۴ ماتریسهای خلوت ۱۴۹
۱۵ـ۴ مسأله‌های حل شده ۱۵۱
۱۶ـ۴ مسأله‌های تکمیلی ۱۶۶

فصل ۵ لیستهای پیوندی

۱ـ۵ مقدمه ۱۷۳
۲ـ۵ لیستهای پیوندی ۱۷۴
۳ـ۵ نمایش لیستهای پیوندی در حافظه ۱۷۶
۴ـ۵ پیمایش یک لیست پیوندی ۱۸۱

۵ـ۵ جستجو در یک لیست پیوندی ۱۸۳
۵ـ۶ تخصیص حافظه، جمع‌آوری حافظهٔ بلااستفاده ۱۸۶
۵ـ۷ اضافه کردن گره به یک لیست پیوندی ۱۹۱
۵ـ۸ حذف گره از یک لیست پیوندی ۲۰۰
۵ـ۹ لیستهای پیوندی دارای سرلیست ۲۰۸
۵ـ۱۰ لیستهای دو طرفه ۲۱۴
۵ـ۱۱ مسأله‌های حل شده ۲۲۰
۵ـ۱۲ مسأله‌های تکمیلی ۲۳۰

فصل ۶ پشته‌ها، صفها، زیربرنامه‌های بازگشتی

۶ـ۱ مقدمه ۲۳۹
۶ـ۲ پشته‌ها ۲۴۰
۶ـ۳ نمایش پشته‌ها به کمک آرایه ۲۴۳
۶ـ۴ عبارتهای محاسباتی، نمادگذاری لهستانی ۲۴۶
۶ـ۵ QuickSort ، یک کاربرد از پشته‌ها ۲۵۲
۶ـ۶ زیربرنامهٔ بازگشتی ۲۵۸
۶ـ۷ برجهای هانوی ۲۶۴
۶ـ۸ پیاده‌سازی زیربرنامه‌های بازگشتی به وسیلهٔ پشته‌ها ۲۶۸
۶ـ۹ صفها ۲۷۶
۶ـ۱۰ صفهای دو سره Deque ۲۸۲
۶ـ۱۱ صفهای اولویت ۲۸۳
۶ـ۱۲ مسأله‌های حل شده ۲۸۸
۶ـ۱۳ مسأله‌های تکمیلی ۳۰۷

فصل ۷ درختها

۷ـ۱ مقدمه ۳۱۵
۷ـ۲ درختهای دودویی ۳۱۵
۷ـ۳ نمایش درختهای دودویی در حافظه ۳۲۰
۷ـ۴ پیمایش درختهای دودویی ۳۲۵
۷ـ۵ الگوریتم‌های پیمایش به کمک پشته‌ها ۳۳۰
۷ـ۶ سرگره‌ها، گره‌های نخ‌کشی شده ۳۳۸
۷ـ۷ درختهای جستجوی دودویی ۳۴۳

۷-۸ جستجو و وارد کردن یک عنصر در درختهای جستجوی دودویی ۳۴۵
۷-۹ حذف یک عنصر از یک درخت جستجوی دودویی ۳۵۱
۷-۱۰ Heap ، HeapSort ۳۵۷
۷-۱۱ طول مسیر، الگوریتم هافمن ۳۶۵
۷-۱۲ درختهای عمومی ۳۷۳
۷-۱۳ مسأله‌های حل شده ۳۷۸
۷-۱۴ مسأله‌های تکمیلی ۳۹۲

فصل ۸ گرافها و کاربردهای آن

۸-۱ مقدمه ۴۰۳
۸-۲ چند اصطلاح نظریهٔ گراف ۴۰۳
۸-۳ نمایش ترتیبی گرافها، ماتریس مجاورت، ماتریس مسیر ۴۰۸
۸-۴ الگوریتم وارشال، کوتاهترین مسیر ۴۱۲
۸-۵ نمایش یک گراف با استفاده از لیست پیوندی ۴۱۶
۸-۶ عملیات بر روی گرافها ۴۲۰
۸-۷ پیمایش یک گراف ۴۲۷
۸-۸ مجموعه‌های جزئاً مرتب، مرتب سازی موضعی ۴۳۲
۸-۹ مسأله‌های حل شده ۴۳۸
۸-۱۰ مسأله‌های تکمیلی ۴۵۱

فصل ۹ مرتب کردن و جستجوی اطلاعات

۹-۱ مقدمه ۴۶۱
۹-۲ مرتب کردن ۴۶۲
۹-۳ مرتب کرن درجی ۴۶۷
۹-۴ مرتب کردن انتخابی ۴۷۰
۹-۵ ادغام کردن ۴۷۲
۹-۶ ادغام و مرتب کردن ۴۷۶
۹-۷ مرتب کردن مبنایی ۴۷۹
۹-۸ جستجوی اطلاعات و اصلاح داده‌ها ۴۸۲
۹-۹ درهم سازی ۴۸۳
۹-۱۰ مسأله‌های تکمیلی ۴۹۱
واژه‌نامهٔ ریاضی و کامپیوتر ۴۹۵

فصل ۱

معرفی و مرور سریعِ مطالب کتاب

۱ـ۱ مقدمه

این فصل موضوع ساختمان داده‌ها را معرفی می‌کند و مروری سریع بر مطالب کتاب خواهد داشت. اصطلاحات و مفاهیم اصلی، تعریف می‌شوند و مثالهایی در رابطه با آنها ارائه می‌شود. مرور کلی بر سازماندهی داده‌ها و ساختمان داده‌های معین به همراه بحث و بررسی عملیات مختلفی که بر روی این ساختمان داده‌ها صورت می‌گیرد از جمله مطالب کتاب را تشکیل می‌دهد. در پایان، مفهوم الگوریتم و پیچیدگی آن معرفی می‌شود و توازن‌بین زمان اجرا و حافظه راکه ممکن است در انتخاب یک الگوریتم خاص و ساختمان دادهٔ مربوط به یک مسأله معین اتفاق بیفتد مورد بررسی قرار خواهیم داد.

۱ـ۲ اصطلاحات اصلی و سازماندهی ابتدایی داده‌ها

داده‌ها، فقط مقادیر یا مجموعه‌هایی از مقادیر هستند. یک **عنصر داده‌ای**، به معنی واحد منحصر بفرد از مقادیر است. عنصرهای داده‌ای که به زیرعنصرهایی تقسیم شده باشد **عنصرهای چند قسمتی** نامیده می‌شوند و آن دسته از عناصر که چند قسمتی نیستند **عنصرهای ابتدایی** نامیده می‌شوند. برای مثال، اسم یک کارمند ممکن است از سه زیر عنصر تشکیل شده باشد، اسم اول، اسم وسط و اسم آخر، اما شمارهٔ تأمین اجتماعی **SSN** معمولاً به صورت یک عنصر منحصر بفرد مورد توجه قرار می‌گیرد.

غالباً مجموعهٔ داده‌ها در سلسله مراتب **فیلدها**، **رکوردها** و **فایلها** سازماندهی می‌شوند. برای این که مفهوم این اصطلاحات دقیقتر بیان شود، چند اصطلاح دیگر را معرفی می‌کنیم. یک **موجودیت**، چیزی است که دارای **خصیصه‌ها** یا خواص معین باشد و ممکن است مقادیری به آن نسبت داده شود. این مقادیر، ممکن است عددی یا غیرعددی باشند. برای مثال، برای موجودیتی نظیر کارمند یک سازمان معین، ممکن است خصیصه‌ها و مقادیر متناظر آنها به صورت زیر باشد :

خصیصه‌ها :	Name	Age	Sex	Social Security Number
مقادیر :	ROHLAND,GAIL	34	F	134 – 24 – 5533

موجودیت‌هایی که دارای خصیصه‌های مشابه هستند مانند تمام کارمندان یک سازمان، تشکیل یک **مجموعهٔ موجودیت** را می‌دهند. هر خصیصه از یک مجموعه موجودیت دارای یک **دامنه** از مقادیر است، این دامنه، مجموعهٔ تمام مقادیر ممکنی است که می‌توان آنها را به خصیصه‌های خاص نسبت داد. اصطلاح **اطلاعات** گاهی اوقات به جای داده‌هایی با خصیصه‌های معین بکار می‌رود؛ به عبارت دیگر، به داده‌های دارای معنی یا داده‌های پردازش شده، **اطلاعات** می‌گویند.

روشی که در آن داده‌ها به سلسله مراتب فیلد، رکورد و فایل سازماندهی می‌شوند در رابطهٔ بین خصیصه‌ها، موجودیت‌ها و مجموعه‌های موجودیت انعکاس پیدا می‌کند؛ یعنی، یک **فیلد** یک واحد ابتدایی منحصر بفرد از اطلاعات است که یک خصیصه از یک موجودیت را به نمایش می‌گذارد. یک **رکورد** مجموعه‌ای از مقادیر فیلدهای یک موجودیت معین است و یک **فایل** مجموعه‌ای از رکوردهای موجودیت در یک مجموعه از موجودیت معین است.

هر رکورد یک فایل ممکن است از چند فیلد تشکیل شود اما مقدار یک فیلد معین، می‌تواند به‌طور منحصر بفرد رکورد داخل فایل را مشخص کند. به یک چنین فیلدی که آن را **K** می‌نامیم **کلید اولیه** یا **اصلی** و به مقادیر k_1 ، k_2 ، ... این فیلد، **کلیدها** یا **مقادیر کلیدی** می‌گویند.

مثال ۱ـ ۱

(الف) فرض کنید یک بنگاه معاملات اتومبیل مشخصات اتومبیل‌ها را در یک فایل نگهداری می‌کند که هر رکورد آن شامل فیلدهای زیر است :

Serial Number شمارهٔ سریال Type نوع Year سال Price قیمت Accessories لوازم‌همراه

فیلد شمارهٔ سریال را می‌توان به عنوان کلید اولیهٔ فایل مورد استفاده قرار داد، زیرا هر اتومبیل شمارهٔ سریال منحصر بفرد دارد.

(ب) فرض کنید یک سازمان مشخصات عضوهایش را در یک فایل نگهداری می‌کند که هر رکورد آن شامل فیلدهای زیر است :

Name نام Address آدرس Telephone Number شماره تلفن Dues Owed دیون بدهکار

هرچند هر رکورد چهار عنصر داده‌ای دارد، نام و آدرس ممکن است عنصرهای چند قسمتی باشند. در اینجا فیلد نام، یک کلید اولیه است. توجه دارید که فیلدهای آدرس و شمارهٔ تلفن را نمی‌توان به عنوان کلید اولیه مورد استفاده قرار داد چون برخی از عضوها ممکن است متعلق به یک خانواده باشند و درنتیجه دارای یک آدرس و یک شماره تلفن هستند.

رکوردها را می‌توان برحسب طول آنها نیز دسته‌بندی کرد. یک فایل می‌تواند دارای رکوردهایی با طول ثابت یا رکوردهایی با طول متغیر باشد. در رکوردهای با طول ثابت، تمام رکوردها دارای فیلدهای برابر و یکسان هستند و به هر فیلد مقدار حافظهٔ مساوی اختصاص می‌یابد. در رکوردهای با طول متغیر، رکوردهای فایل ممکن است طولهای مختلف داشته باشند؛ برای مثال، رکوردهای دانشجویی معمولاً طول متغیر دارند چون دانشجویان مختلف تعداد درسهای متفاوتی را اختیار می‌کنند. معمولاً رکوردهای با طول متغیر، یک طول حداقل و یک طول حداکثر دارند.

سازماندهی داده‌ها به شکل بالا به صورت فیلد، رکورد و فایل ممکن است آنقدر پیچیده نباشد تا مجموعه‌هایی از داده‌های معین را در حافظه ذخیره کند و به صورت موثر و کارا مورد پردازش قرار دهد.به همین دلیل، داده‌ها نیز به انواع پیچیده‌تری از ساختمانها تقسیم‌بندی می‌شوند. مطالعه چنین ساختمان داده‌هایی، که موضوع اصلی این کتاب را تشکیل می‌دهد، شامل سه مرحلهٔ زیر است :

(۱) توصیف منطقی یا ریاضی ساختمان داده

(۲) پیاده‌سازی این ساختمان داده بر روی یک کامپیوتر

(۳) تجزیه و تحلیل کمی این ساختمان‌داده که شامل تعیین مقدار حافظهٔ موردنیاز برای ذخیرهٔ این ساختمان و زمان موردنیاز برای پردازش آن است.

بخش بعدی، برخی از این ساختمان داده‌ها را به شما معرفی می‌کند.

توجه کنید : مرحلهٔ دوم و سوم مطالعه ساختمان داده‌ها بستگی به این دارد که آیا داده‌ها (**الف**) در حافظهٔ اصلی (اولیه) کامپیوتر ذخیره می‌شوند یا (**ب**) در واحد حافظهٔ ثانویه (خارجی). این کتاب اساساً حالت اول را پوشش می‌دهد؛ یعنی آدرس دادن به یک خانهٔ حافظه و زمان موردنیاز برای دسترسی به محتوای خانهٔ حافظه، که بستگی به یک حافظهٔ معین یا به محتوای حافظه‌ای که قبلاً به آن دسترسی پیدا کردیم ندارد. حالت دوم، که مدیریت فایل یا مدیریت پایگاه داده‌ها نیز نامیده می‌شود مطالب مربوط به خود را شامل می‌شود که بحث و بررسی آن از محدودهٔ درس ساختمان داده‌ها خارج است.

۳ـ۱ ساختمان داده‌ها

داده‌ها می‌توانند به صورتهای متفاوتی سازماندهی شوند. مدل منطقی یا ریاضی سازماندهی داده‌ها

به یک صورت خاص، یک **ساختمان داده** نامیده می‌شود. انتخاب یک مدل خاص از داده‌ها بستگی به دو مسأله قابل توجه دارد. نخست این‌که، ازنظر ساختمان باید به اندازهٔ کافی غنی باشد تا بتواند رابطه‌های واقعی بین داده‌ها را در دنیای ما منعکس سازد، از سوی دیگر، این ساختمان باید به اندازهٔ کافی ساده باشد تا بتواند به هنگام نیاز، داده‌ها را به صورت مؤثری پردازش کند. این بخش، برخی از ساختمان داده‌هایی را که دیرتر در این کتاب به طور مشروح مورد بررسی قرار می‌گیرد، به ما معرفی می‌کند.

آرایه‌ها

یک **آرایهٔ خطی** (یا **یک بعدی**) ساده‌ترین نوع ساختمان داده‌ها است. منظور از یک آرایهٔ خطی، یک لیست متناهی با **n** عنصر داده‌ای مشابه است که به عناصر آن به ترتیب به کمک مجموعه‌ای از **n** عدد متوالی که معمولاً **1** ، **2** ، **3** ، ... **n** می‌باشد دسترسی پیدا می‌کنیم. اگر ما برای نام آرایه، **A** را اختیار کنیم، آنگاه عنصرهای آرایهٔ **A** را با نماد اندیس‌گذاری شدهٔ

$$a_1, a_2, a_3, \ldots, a_n$$

یا نماد پرانتزی

$$A(1), A(2), A(3), \ldots, A(N)$$

یا با نماد کروشه‌ای

$$A[1], A[2], A[3], \ldots, A[N]$$

نمایش می‌دهیم.

صرفنظر از نوع نمادگذاری، عدد **K** در **A[K]** یک **اندیس** و **A[K]** یک **متغیر اندیس‌دار** نامیده می‌شود.

توجه کنید : نماد پرانتزی و نماد کروشه‌ای غالباً هنگامی مورد استفاده قرار می‌گیرند که نام آرایه بیش از یک حرف داشته باشد یا نام آرایه در یک الگوریتم بکار گرفته شود. هنگام استفاده از این نمادگذاری، معمولاً برای نام آرایه و اندیس آرایه همانگونه که در بالا بیان شده است از حروف بزرگ نظیر **A** و **N** استفاده می‌کنیم. درغیر اینصورت، می‌توانیم از نمادگذاری متداول اندیسی کج برای نام و اندیسهایش و از حروف کوچک، به صورتی که در بالا با **a** و **n** بیان شده است برای اندیسها استفاده کنیم. از نمادگذاری اول اغلب درکتب چاپی ریاضی استفاده می‌شود حال آنکه در اکثر کتب کامپیوتری، نمادگذاری دوم دیده می‌شود.

مثال ۲ـ ۱

آرایهٔ خطی **STUDENT** از نام شش دانشجو، به صورتی که در شکل ۱ـ ۱ نشان داده، تشکیل شده

است. در اینجا STUDENT[1]، John Brown و STUDENT[2]، Sandra Gold و غیره را نمایش می‌دهد.

	STUDENT
1	John Brown
2	Sandra Gold
3	Tom Jones
4	June Kelly
5	Mary Reed
6	Alan Smith

شکل ۱ـ ۱

آرایه‌های خطی، **آرایه‌های یک‌بعدی** نیز نامیده می‌شوند زیرا به هر عنصر چنین آرایه‌ای تنها می‌توان از طریق یک اندیس مراجعه کرد. یک **آرایهٔ دوبعدی**، یک مجموعه از عناصر داده‌ای مشابه است که به هر عنصر آن از طریق دو اندیس مراجعه می‌کنیم. به آرایه دوبعدی در ریاضیات **ماتریس** و در امور تجاری **جدول** می‌گویند. آرایه‌های چند بعدی به‌طور مشابه تعریف می‌شوند. بحث و بررسی مشروح آرایه‌ها در فصل ۴ کتاب گنجانده شده است.

مثال ۳ـ ۱

یک فروشگاه زنجیره‌ای از ۲۸ فروشگاه تشکیل شده است که هر فروشگاه ۴ بخش دارد و می‌توان فروش هفتگی آن را (برحسب نزدیکترین عدد به دلار) مانند شکل ۲ـ ۱ ارائه داد :

Store \ Dept.	1	2	3	4
1	2872	805	3211	1560
2	2196	1223	2525	1744
3	3257	1017	3686	1951
. . .	. . .	. . .	. . .	. . .
28	2618	931	2333	982

شکل ۲ـ ۱

این نوع داده‌ها را می‌توان با استفاده از یک آرایهٔ دو بعدی در کامپیوتر ذخیره کرد که در آن اندیس اول شمارهٔ فروشگاه و اندیس دوم شمارهٔ بخش را نشان می‌دهد. اگر **SALES** نامی باشد که به این آرایه داده‌ایم، آنگاه

SALES[1, 1] = 2872, SALES[1, 2] = 805, SALES[1, 3] = 3211, . . . , SALES[28, 4] = 982

اندازهٔ این آرایه به صورت **28×4** (بخوانید ۲۸ در ۴) نمایش داده می‌شود زیرا این آرایه ۲۸ سطر (شماره‌های خطوط افقی) و ۴ ستون (شماره‌های خطوط عمودی) دارد.

لیستهای پیوندی

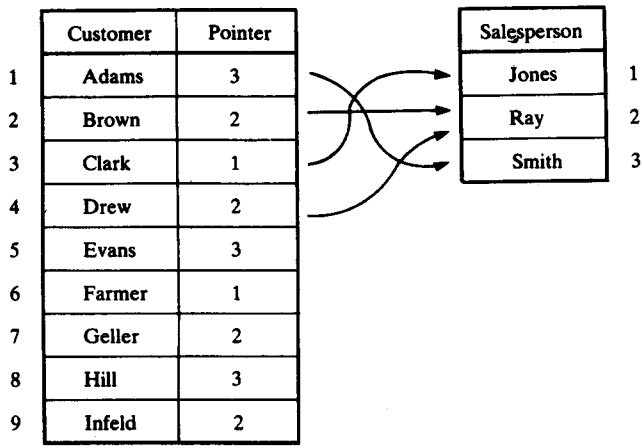
به کمک یک مثال لیستهای پیوندی را معرفی می‌کنیم. فرض کنید یک شرکت حق‌العمل کار، مشخصات مشتری‌های خود را در یک فایل نگهداری می‌کند که هر رکورد آن شامل نام مشتری **Customer** و فروشنده **Salesperson** است و فرض کنید این فایل شامل داده‌هایی است که در شکل ۳- ۱ نشان داده شده است :

	Customer	Salesperson
1	Adams	Smith
2	Brown	Ray
3	Clark	Jones
4	Drew	Ray
5	Evans	Smith
6	Farmer	Jones
7	Geller	Ray
8	Hill	Smith
9	Infeld	Ray

شکل ۳- ۱

واضح است که این فایل می‌توانست به وسیلهٔ چنین جدولی یعنی به‌وسیلهٔ یک جدول دوستونه با نُه نام در کامپیوتر ذخیره شود. با وجود این، همانگونه که بحث زیر نشان می‌دهد، این روش نمی‌تواند مفیدترین راه برای ذخیره داده‌ها باشد.

روش دیگر برای ذخیره داده‌های شکل ۳- ۱ ، آن است که یک آرایه مستقل برای فروشنده‌ها و یک مدخل موسوم به **اشاره‌گر** در فایل مشتری داشته باشیم که به مکان فروشندهٔ هر مشتری اشاره می‌کند. این روش ذخیره داده‌ها در شکل ۴- ۱ نشان داده شده است که در آن تنها تعدادی از اشاره‌گرها به وسیلهٔ



	Customer	Pointer
1	Adams	3
2	Brown	2
3	Clark	1
4	Drew	2
5	Evans	3
6	Farmer	1
7	Geller	2
8	Hill	3
9	Infeld	2

	Salesperson
1	Jones
2	Ray
3	Smith

شکل ۴ـ ۱

اما در کامپیوتر عملاً از یک عدد صحیح استفاده می‌شود. چون یک اشاره‌گر، نسبت به یک اسم به حافظهٔ کمتری نیاز دارد. از این‌رو، این نحوهٔ نمایش باعث صرفه‌جویی در حافظه می‌شود به‌ویژه این که اگر هر فروشنده چند صد مشتری داشته باشد، اهمیت صرفه‌جویی در حافظه بیشتر نمایان می‌شود.

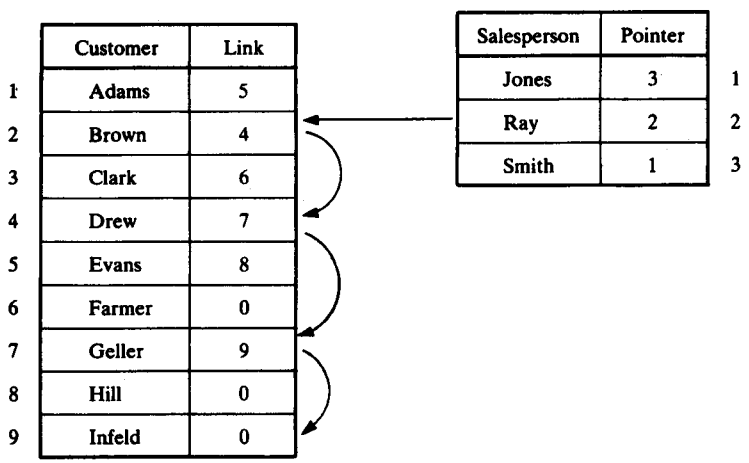
فرض کنید این شرکت بخواهد لیست تمام مشتریان یک فروشندهٔ معین را به دست آورد. با استفاده از داده‌های نشان داده‌شده در شکل ۴ـ ۱، شرکت مجبور است در میان فایل تمام مشتری‌ها، فروشندهٔ موردنظر را جستجو کند. یک راه ساده برای چنین جستجویی، داشتن پیکانهایی نظیر شکل ۴ـ ۱ است که به اطلاعات دیگر اشاره می‌کند. اکنون هر فروشنده یک مجموعه از اشاره‌گر دارد که به مکان مشتریهای وی اشاره می‌کند. شکل ۵ـ ۱ را ببینید.

	Salesperson	Pointer
1	Jones	3, 6
2	Ray	2, 4, 7, 9
3	Smith	1, 5, 8

شکل ۵ـ ۱

عیب اصلی این نمایش آن است که هر فروشنده ممکن است چند اشاره‌گر داشته باشد و با اضافه و حذف شدن مشتریها، مجموعهٔ اشاره‌گرها تغییر خواهند کرد.

یکی از راههای بسیار متداول ذخیرهٔ داده‌های شکل ۳ـ ۱ ، در شکل ۶ـ ۱ نشان داده شده است. در اینجا هر فروشنده یک اشاره‌گر دارد که به مشتری اول اشاره می‌کند که اشاره‌گر آن نیز به نوبهٔ خود به مشتری دوم اشاره می‌کند و الی آخر، که در آن فروشندهٔ مشتری آخر با یک **0** مشخص شده است. این اشاره‌گرها برای فروشنده‌ای به نام **Ray** به وسیلهٔ پیکانهایی، در شکل ۶ـ ۱ نشان داده شده است. با استفاده از این نمایش به سادگی می‌توان لیست تمام مشتریهای یک فروشندهٔ معین را به دست آورد. همانگونه که در فصل ۵ خواهید دید به‌سادگی می‌توان عمل اضافه‌کردن و حذف مشتریها را انجام داد.



شکل ۶ـ ۱

نمایش داده‌های شکل ۶ـ ۱ یک مثال از لیستهای پیوندی است. اگرچه معمولاً اصطلاحات "اشاره‌گر" و "پیوند" به یک معنی بکار می‌روند. اما سعی می‌کنیم از اصطلاح "اشاره‌گر" در مواردی استفاده کنیم که یک عنصر از یک لیست به یک عنصر در لیست دیگر اشاره می‌کند و اصطلاح "پیوند" را برای مواردی نگه می‌داریم که یک عنصر از یک لیست به عنصر دیگری از همان لیست اشاره می‌کند.

درختها

داده‌ها اغلب شامل رابطهٔ سلسله مراتبی بین عناصر مختلف هستند. ساختمان داده‌ای که این رابطه را

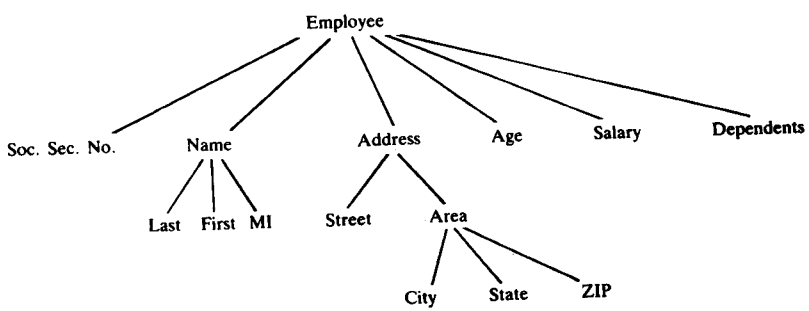
منعکس می‌کند یک **گراف درختیِ ریشه‌دار** یا فقط یک **درخت** نامیده می‌شود. درختها در فصل ۷ تعریف شده و به‌طور مفصل موردبحث و بررسی قرار می‌گیرند. در اینجا ما با ارائه دو مثال تنها برخی از خواص اصلی آنها را بیان می‌کنیم.

مثال ۴ـ ۱ : ساختمان رکورد

هرچند یک فایل را می‌توان به وسیلهٔ یک یا چند آرایه یا رکورد نگهداری کرد که هر یک از آنها، می‌توانند هم فیلدهای چندقسمتی و هم فیلدهای ابتدایی باشند. اما می‌توان آن را به وسیلهٔ یک ساختمان درختی به شکل بهتری بیان کرد. برای مثال، رکورد پرسنلی یک کارمند می‌تواند شامل فیلدهای زیر باشد :

Social Security Number, Name, Address, Age, Salary, Dependents

با وجود این، نام **Name** ممکن است یک فیلد چند قسمتی با فیلدهای کوچک نام **Name**، نام آخر، **Last** نام اول **First** و نام وسط **MI** باشد. علاوه بر این، آدرس **Address** ممکن است یک فیلد چند قسمتی با فیلدهای کوچک آدرس خیابان **Street** و آدرس منطقه **Area** باشد که در آن منطقه **Area** خود می‌تواند یک فیلد چند قسمتی با فیلدهای کوچک شهر، **City**، ایالت **State** و شماره کد پستی **ZIP** باشد. این ساختمان سلسله مراتبی در شکل ۷ـ ۱ (الف) به تصویر درآمده است.



شکل ۷ـ ۱ (الف)

روش دیگرِ به نمایش گذاشتن یک ساختمان درختی از این نوع، برحسب سطوح آن در شکل ۷ـ ۱ (ب) نشان داده شده است.

```
01  Employee
    02  Social Security Number
    02  Name
        03  Last
        03  First
        03  Middle Initial
    02  Address
        03  Street
        03  Area
            04  City
            04  State
            04  ZIP
    02  Age
    02  Salary
    02  Dependents
```

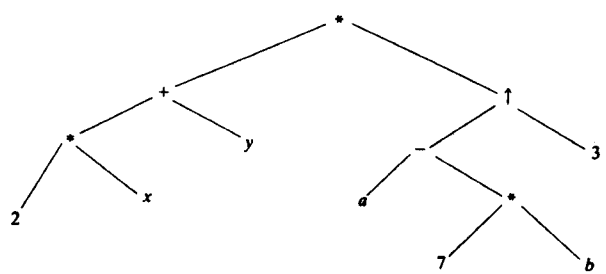
شکل ۷ـ ۱ (ب)

مثال ۵ـ ۱ : عبارتهای جبری

عبارت جبری زیر را درنظر بگیرید :

$$(2x+y)(a-7b)^3$$

با استفاده از پیکان به طرف بالای ↑ برای توان و ستاره * برای ضرب، می‌توانیم این عبارت را به صورت درخت شکل ۸ـ ۱ نمایش دهیم.



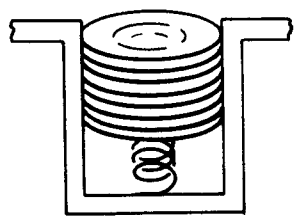
شکل ۸ـ ۱

ملاحظه می‌کنید که ترتیب اجرای عملیات در این نمودار نیز، انعکاس یافته است، عمل توان‌رساندن باید بعد از انجام عمل تفریق عبارت داخل پرانتز اجرا شود و عمل ضرب نشان داده شده در بالای درخت باید آخر اجرا شود.

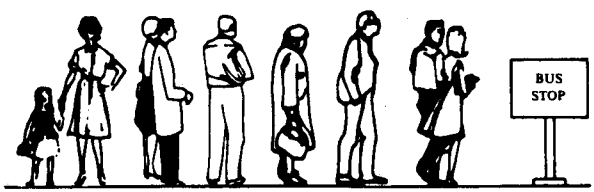
به غیر از آرایه‌ها، لیستهای پیوندی و درختها، ساختمان داده‌های دیگری وجود دارند که ما برخی از این ساختمانها راکه به اختصار در زیر شرح داده‌ایم دیرتر بطور مشروح مورد مطالعه قرار خواهند گرفت.

(الف) پشته Stack : یک پشته که به آن یک سیستم **LIFO** یا **آخرین ورودی اولین خروجی** است، نیز

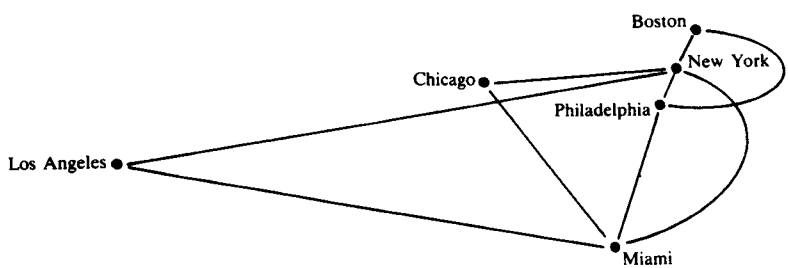
می‌گویند، یک لیست خطی است که در آن عملیات اضافه‌شدن عناصر تنها از یک انتهای آن موسوم به **بالای پشته Top** صورت می‌گیرد. این ساختمان، عملیاتی مشابه یک پشته از بشقابها بر روی یک سیستم فنری دارد که در شکل ۹ـ ۱ (الف) نشان داده شده است. توجه دارید که بشقابهای جدید را می‌توان تنها از بالای پشته به آن اضافه کرد و عمل حذف یا برداشتن بشقابها تنها از بالای پشته امکان‌پذیر است.



(الف) یک پشته از بشقابها



(ب) صف انتظار اتوبوس



(ج) خطوط هواپیمایی

شکل ۹ـ ۱

(ب) صف Queue : یک صف، که به آن یک سیستم **FIFO** یا **اولین ورودی اولین خروجی** است، نیز می‌گویند، یک لیست خطی است که در آن عملیات حذف عناصر تنها از یک انتهای لیست موسوم به **ابتدا Front** لیست و اضافه‌شدن عناصر به صف تنها از انتهای دیگر لیست، موسوم به **انتها Rear** لیست صورت می‌گیرد. این ساختمان داده به همان صورتی عمل می‌کند که ما در صفِ انتظار یک ایستگاه اتوبوس عمل می‌کنیم. این وضعیت در شکل ۹ـ ۱ (ب) به تصویر درآمده است : اولین نفرِ داخل صف اولین کسی است که سوار اتوبوس می‌شود. یک مثال دیگر از صف، مربوط به اتومبیل‌هایی است که در یک چهارراه در انتظار عبور به سر می‌برند، به‌محض سبزشدن چراغ راهنمایی، اولین اتومبیلِ داخل

صف اولین اتومبیلی است که از چهارراه عبور می‌کند.

(ج) گراف : گاهی اوقات داده‌ها یک رابطه بین جفت عناصر موجود در طبیعت را بیان می‌کنند که لزوماً به صورت سلسله مراتبی نیستند. برای مثال، خطوط هواپیمایی شکل ۹-۱ (ج) را درنظر بگیرید که ارتباط بین شهرها تنها با رسم خطوط نشان داده شده است. ساختمان داده‌ای که این نوع رابطه را بیان می‌کند یک **گراف** نام دارد. گرافها به صورت رسمی در فصل ۸ تعریف شده، مورد مطالعه و بررسی کامل قرار می‌گیرند.

توجه کنید : برای عناصر یک ساختمان داده از اسم‌های مختلف و گوناگونی استفاده می‌شود. برخی از اسم‌های متداول عبارتند از: عنصر داده‌ای، فیلد، مجموعه‌ای از فیلدها، رکورد، گره و شیئی داده‌ای. اسم خاصی که مورد استفاده قرار می‌گیرد، بستگی به نوع ساختمان داده‌ای دارد که در یک متن یا یک مبحث، از آن ساختمان داده استفاده می‌شود و برنامه‌نویسان از آن اسم استفاده می‌کنند. ما برای این منظور اصطلاح "عنصر داده‌ای" را ترجیح می‌دهیم اما هنگام بحث و بررسی فایلها از اصطلاح رکورد و هنگام مطالعه لیستهای پیوندی، و همچنین درختها و گرافها از اصطلاح گره استفاده می‌کنیم.

۴-۱ عملیات بر روی ساختمان داده‌ها

داده‌هایی که در ساختمان داده‌ها ظاهر می‌شوند به وسیلهٔ عملیات مشخصی پردازش می‌شوند. درواقع، ساختمان دادهٔ خاصی که برنامه‌نویس برای یک مسألهٔ معین انتخاب می‌کند بستگی زیاد به میزان عملیات خاصی دارد که در آن مسأله انجام می‌شود. در این بخش دانشجو با برخی از این عملیات که زیاد مورد استفاده قرار می‌گیرد، آشنا می‌شود.

در این کتاب چهار عمل زیر نقش اصلی را ایفا می‌کنند :

(۱) پیمایش : دسترسی به اطلاعات یک رکورد آن هم دقیقاً یکبار، **پیمایش** نامیده می‌شود، طوری که بتوان بعضی از فیلدهای رکورد را مورد پردازش قرار داد. گاهی اوقات به دسترسی و پردازش رکورد **ملاقات** نیز گفته می‌شود.

(۲) جستجو : به عمل یافتن مکان یک رکورد با یک مقدار کلیدی معین یا پیداکردن مکان تمام رکوردهایی که در یک یا چند شرط صدق می‌کنند، **جستجو** می‌گویند.

(۳) اضافه‌کردن : افزودن رکورد جدید به ساختمان داده است.

(۴) حذف‌کردن : حذف یک رکورد از ساختمان داده است.

گاهی اوقات ممکن است از دو یا چند عمل در یک کار مشخص استفاده شود. مثلاً چنانچه بخواهیم یک رکورد با کلید معلوم را حذف کنیم لازم است نخست مکان آن رکورد را جستجو (پیدا) کنیم.

دو عمل زیر نیز در وضعیت‌های خاص مورد استفاده قرار می‌گیرند که در زیر آنها را ازنظر می‌گذرانیم :

(۱) مرتب‌کردن : عبارت است از قراردادن رکوردها با یک نظم معین در کنار هم. مثلاً با یک نظم الفبایی براساس یک کلید نام **NAME** یا با یک نظم عددی برطبق یک کلید عددی **NUMBER** مانند شمارهٔ تأمین اجتماعی یا شمارهٔ حساب بانکی.

(۲) ادغام‌کردن : ترکیب رکوردهای دو فایل مرتب‌شدهٔ مختلف و قراردادن آنها در یک فایل مرتب‌شده **ادغام کردن** نام دارد.

عملیات دیگری مانند **کپی‌کردن** و **اتصال اطلاعات** دیرتر در این کتاب بررسی می‌شود.

مثال ۶ـ ۱

یک سازمان، یک فایل برای اعضایش دارد که در آن فایل، هر رکورد یک عضو معین، دارای فیلدهای زیر است :

Name, Address, Telephone Number, Age, Sex

(الف) فرض کنید این سازمان بخواهد یک گردهم‌آیی برای اعضایش ترتیب دهد. در آن‌صورت لازم است که عمل پیمایش در فایل انجام شود تا نام و آدرس هر عضو به‌دست آید.

(ب) فرض کنید این سازمان بخواهد نام تمام عضوهایی را که در یک منطقهٔ معین زندگی می‌کنند به دست آورد. برای این حالت نیز لازم است عمل پیمایش در فایل انجام شود تا اطلاعات موردنیاز به دست آید.

(ج) فرض کنید این سازمان بخواهد آدرس یک نام **Name** معین را به‌دست آورد. در آن صورت لازم است عمل جستجو در این فایل انجام شود تا رکورد مربوط به این نام **Name** به دست آید.

(د) فرض کنید شخص جدیدی به این سازمان پیوسته است. در آن صورت لازم است رکورد وی به فایل اضافه شود.

(ه) فرض کنید یک عضو این سازمان درگذشته است. در آن صورت لازم است رکورد وی از این فایل حذف شود.

(و) فرض کنید یک عضو محل زندگی‌اش را تغییر داده است و دارای آدرس و شمارهٔ تلفن جدید است. با معلوم‌بودن نام این عضو، نخست لازم است رکورد او را در فایل جستجو کنیم. آنگاه لازم است عمل "تازه شدن" یا **Update** انجام شود یعنی بعضی از فیلدهای رکورد او را با توجه به مشخصات تازهٔ او تغییر دهیم.

(ز) فرض کنید این سازمان بخواهد تعداد اعضای خود را که ۶۵ سال یا بیشتر سن دارند به دست آورد.

برای این حالت نیز لازم است عمل پیمایش در فایل انجام شود و تعداد این عضوها شمرده شوند.

۵ـ ۱ الگوریتم‌ها :پیچیدگی الگوریتم،توازن بین زمان اجرا و حافظه

یک الگوریتم، یک لیست خوش تعریف از مراحلی است که برای حل یک مسأله معین درنظر گرفته می‌شود. یکی از اهداف اصلی این کتاب ارائه الگوریتم‌های کارا و مؤثر برای پردازش داده‌ها است. از زمان و حافظه به عنوان دو معیار اصلی در کارآیی یک الگوریتم استفاده می‌شود. پیچیدگی یک الگوریتم، تابعی است که زمان اجرا و/ یا حافظه را برحسب تعداد داده‌های ورودی به دست می‌دهد. مفهوم پیچیدگی یک الگوریتم در فصل ۲ بررسی می‌شود.

هر یک از الگوریتم‌های این کتاب حاوی ساختمان دادهٔ خاصی است. براین اساس همیشه نمی‌توانیم از کاراترین الگوریتم استفاده کنیم چون انتخاب ساختمان داده به عوامل زیادی از قبیل، نوع داده‌ها و تعداد دفعاتی که با آن، عملیات داده‌ای متعدد انجام می‌شود بستگی دارد. گاهی اوقات در انتخاب ساختمان داده‌ها مسأله توازن (سبک و سنگینی) بین زمان اجرا و حافظه مورد توجه قرار می‌گیرد به این معنی که : جهت ذخیرهٔ داده‌ها، با افزایش مقدار حافظه، می‌توان زمان لازم برای پردازش داده‌ها راکاهش داد و برعکس. این ایده‌ها را با دو مثال روشن می‌کنیم.

الگوریتم‌های جستجو

فایل اعضای سازمان مثال ۶ـ ۱ را درنظر بگیرید که در آن هر رکورد به همراه چند فیلد دیگر، دارای دو فیلد نام **Name** و شماره تلفن **Telephone Number** است. فرض کنید که نامِ یک عضو داده شده است و بخواهیم شمارهٔ تلفن او را پیدا کنیم. یک راه برای این منظور جستجوی خطی در فایل است یعنی از الگوریتم زیر استفاده کنیم.

جستجوی خطی : در هر یک از رکوردهای فایل نامِ داده شده را جستجو کنید به‌محض پیدا شدن رکوردی با نام داده‌شده و درنتیجه با شماره تلفن متناظر با آن، به عمل جستجو خاتمه دهید.

قبل از هر کاری، واضح است که زمان موردنیاز اجرای این الگوریتم متناسب با تعداد مقایسه‌ها است. همچنین فرض کنید هر نام **Name** درون فایل، احتمال مساوی برای انتخاب‌شدن داشته باشند. به‌طور شهودی واضح است که میانگین تعداد مقایسه‌ها برای یک فایل با **n** رکورد برابر $n / 2$ است یعنی پیچیدگی الگوریتم جستجوی خطی به صورت $C(n) = n / 2$ است.

چنانچه بخواهیم عمل جستجو را در میان لیستی شامل چند هزار نام، مانند دفترچهٔ تلفن انجام دهیم استفاده از الگوریتم بالا عملاً غیرممکن است. با وجود این، اگر نامها به صورت الفبایی مرتب باشند

مانند دفترچه‌های تلفن، در آن صورت می‌توانیم از یک الگوریتم کارا به نام **جستجوی دودویی** استفاده کنیم. این الگوریتم به تفصیل در فصل ۴ شرح داده شده است ولی ایدهٔ کلی آن به اختصار در زیر شرح داده می‌شود.

جستجوی دودویی : نام داده شده را با نام وسط لیست مقایسه کنید. انجام این کار به ما می‌گوید که نام داده شده در کدام نیمه از لیست است. آنگاه نام داده شده را با نام وسط نیمهٔ راست مقایسه کنید تا معلوم شود این نام در کدام ربع از لیست وجود دارد. عمل جستجو را تا پیداشدن نام موردنظر در لیست ادامه دهید.

می‌توان نشان داد که پیچیدگی الگوریتم جستجوی دودویی از رابطهٔ

$$C(n) = \log_2 n$$

به دست می‌آید. برای مثال، برای پیداکردن یک نام **Name** داده شده در یک لیست با **25000** نام، به بیش از **15** مقایسه نیاز نداریم.

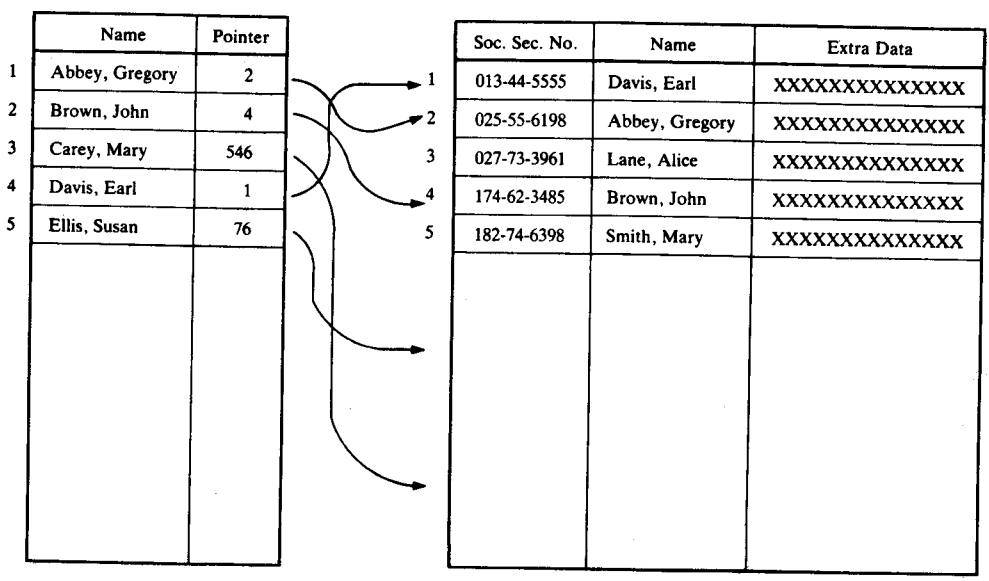
اگرچه الگوریتم جستجوی دودویی الگوریتم بسیار کارآیی است اما یک عیب اساسی دارد. به خصوص این که، این الگوریتم فرض می‌کند به نام وسط لیست یا لیستهای کوچک شدهٔ بعدی دسترسی مستقیم داریم به این معنی که لیست باید در نوعی از آرایه ذخیره شده باشد. متأسفانه، اضافه‌کردن یک عنصر در یک آرایه مستلزم آن است عناصر آرایه یک خانه به طرف پائین لیست انتقال داده شوند و حذف یک عنصر از یک آرایه نیز مستلزم انتقال عناصر آرایه یک خانه به طرف بالای لیست است.

شرکت تلفن، با چاپ یک دفترچهٔ تلفن جدید در هر سال مسألهٔ بالا را حل می‌کند و این درحالی است که شمارهٔ تلفن مشتریان جدید را به طور جداگانه در یک فایلِ موقت نگه می‌دارد یعنی شرکت تلفن هرساله فایل مشترکینش را به‌روز درمی‌آورد. ازطرف دیگر، یک بانک، می‌تواند مشخصات مشتری جدید را تقریباً به‌طور لحظه‌ای در فایل مربوطه اضافه کند. بنابراین، برای یک بانک یک لیست مرتب‌شدهٔ خطی بهترین ساختمان داده نیست.

یک مثال از توازن بین زمان اجرا و حافظهٔ مصرفی

فرض کنید یک فایل از رکوردهایی شامل فیلدهای نام، شمارهٔ تأمین اجتماعی و چند فیلد اضافی دیگر تشکیل شده است. مرتب‌کردن این فایل به صورت الفبایی با استفاده از یک جستجوی دودویی راه بسیار کارآیی در پیداکردن یک رکورد با نام معین است. از طرف دیگر فرض کنید تنها شمارهٔ تأمین اجتماعی یک شخص داده شده است. در آن صورت مجبوریم برای این رکورد یک جستجوی دودویی انجام دهیم که این روش، برای فایلی با تعداد رکوردهای بسیار زیاد زمان اجرا را به میزان قابل توجهی بالا

می‌برد. برای حل اینگونه مسایل چه روشی می‌توان درپیش گرفت؟ یک راه آن است که فایل دیگری داشته باشیم که آن فایل براساس شمارهٔ تأمین اجتماعی به صورت عددی مرتب شده باشد. باوجود این، این روش حافظهٔ موردنیاز برای مرتب‌کردن داده‌ها را به دو برابر افزایش می‌دهد. راه دیگر که در شکل ۱۰ـ ۱ نشان داده شده است، آن است که فایل اصلی را به صورت مرتب‌شدهٔ عددی براساس شمارهٔ تأمین اجتماعی و یک آرایهٔ کمکی تنها با دو ستون دراختیار داشته باشیم، که در این آرایه، ستون اول شامل نام‌ها با ترتیب الفبایی و ستون دوم شامل اشاره‌گرهایی باشد که این اشاره‌گرها به مکان رکوردهای متناظر آن در فایل اصلی اشاره کنند. این روش یکی از راه حلهای این مسأله است که اغلب مورد استفاده قرار می‌گیرد، زیرا حافظهٔ اضافی که تنها شامل دو ستون است برای مقدار اطلاعات اضافی تدارک دیده شده، حداقل حافظه می‌باشد.



	Name	Pointer
1	Abbey, Gregory	2
2	Brown, John	4
3	Carey, Mary	546
4	Davis, Earl	1
5	Ellis, Susan	76

	Soc. Sec. No.	Name	Extra Data
1	013-44-5555	Davis, Earl	XXXXXXXXXXXXXX
2	025-55-6198	Abbey, Gregory	XXXXXXXXXXXXXX
3	027-73-3961	Lane, Alice	XXXXXXXXXXXXXX
4	174-62-3485	Brown, John	XXXXXXXXXXXXXX
5	182-74-6398	Smith, Mary	XXXXXXXXXXXXXX

شکل ۱۰ـ ۱

آرایهٔ کمکی که به صورت الفبایی مرتب شده است.

فایل اصلی که براساس شمارهٔ تأمین اجتماعی Soc. Sec. No مرتب شده است.

توجه کنید: فرض کنید یک فایل به صورت الفبایی براساس شمارهٔ تأمین اجتماعی مرتب شده است.

به‌محض اضافه‌شدن یک رکورد جدید در این فایل، رکوردها باید پیوسته به مکان جدید خود انتقال پیدا کنند تا ترتیب مرتب‌شدهٔ رکوردها حفظ شود. یک راه ساده برای به حداقل‌رساندن مقدار جابجایی رکوردها آن است که از شمارهٔ تأمین اجتماعی به عنوان آدرس هر رکورد استفاده کنیم. با این کار به هنگام اضافه‌شدن یک رکورد، نه‌تنها هیچ‌گونه جابجایی صورت نمی‌گیرد بلکه در هر لحظه می‌توان به هر رکوردی دسترسی پیدا کرد. بنابراین برای تنها چندصد یا احتمالاً چندهزار رکورد، این روش مرتب‌کردن اطلاعات به یک میلیارد (10^9) خانهٔ حافظه احتیاج دارد واضح است که این مقدار حافظه در مقابل زمان اجرا مقرون به صرفه و اقتصادی! نیست. یک روش دیگر برای این منظور، تعریف یک تابع H از مجموعهٔ مقادیر کلید K یعنی، شمارهٔ تأمین اجتماعی به مجموعهٔ L از آدرس خانه‌های حافظه است. به یک چنین تابع H ای، **تابع درهم‌ساز Hashing** می‌گویند. تابعهای درهم‌ساز و خواص آن مطالب فصل ۹ این کتاب را تشکیل می‌دهد.

مسأله‌های حل‌شده

اصطلاحات اصلی

مسأله ۱ـ ۱ : یک استاد دانشگاه لیست دانشجویان یک کلاس را که شامل اطلاعات زیر است نگهداری می‌کند:

Name, Major, Student Number, Test Scores, Final Grade

(الف) موجودیتها، خصیصه‌ها و مجموعه موجودیت لیست را تعیین کنید.

(ب) مقادیر فیلدها، رکوردها و فایل‌ها را شرح دهید.

(ج) برای این لیست چه خصیصه‌هایی را می‌توان به عنوان کلیدهای اولیه مورد استفاده قرار داد؟

حل : (الف) هر دانشجو یک موجودیت است و مجموعه دانشجویان مجموعهٔ موجودیت است. ویژگی‌های دانشجویان ازقبیل نام دانشجو، رشتهٔ او و غیره خصیصهٔ آنها هستند.

(ب) مقادیر فیلد، مقادیری هستند که به خصیصه‌ها نسبت داده می‌شوند یعنی نامهای واقعی، نمرات امتحانات و غیره. مقادیر فیلد هر دانشجو یک رکورد را تشکیل می‌دهد و مجموعه‌ای از تمام رکوردهای دانشجویان فایل دانشجویی آنها است.

(ج) نام **Name** یا شمارهٔ دانشجویی **Student Number** را می‌توان به عنوان کلید اولیه مورد استفاده قرار داد، چون هر یک از این فیلدها رکورد دانشجویی را به‌طور منحصر بفرد مشخص می‌کنند. معمولاً استاد دانشگاه نام **Name** را به عنوان کلید اولیه به کار می‌برد. درحالی که ادارهٔ ثبت نامِ دانشگاه ممکن است از

شمارهٔ دانشجویی **Student Number** به عنوان کلید اولیه استفاده کند.

مسأله ۲ـ ۱ : یک بیمارستان مشخصات بیماران را در یک فایل نگهداری می‌کند که در آن هر رکورد شامل فیلدهای زیر است :

Admission Date, Social Security Number, Room, Bed Number, Doctor

(الف) چه فیلدی را می‌توان به عنوان کلید اولیه مورد استفاده قرار داد؟

(ب) کدام جفت فیلد را می‌توان به عنوان کلید اولیه مورد استفاده قرار داد؟

(ج) کدامیک از فیلدها چندقسمتی هستند؟

حل : **(الف)** نام **Name** و شمارهٔ تأمین‌اجتماعی **Social Security Number** را می‌توان به عنوان کلید اولیه مورد استفاده قرار داد. فرض کرده‌ایم هیچ دو بیماری همنام نیستند.

(ب) اتاق **Room** و شماره تخت **Bed Number** با هم به‌طور منحصر بفرد یک بیمار را مشخص می‌کنند.

(ج) نام **Name**، تاریخ پذیرش **Admission Date** و دکتر **Doctor** فیلدهای چندقسمتی هستند.

مسأله ۳ـ ۱ : کدامیک از فیلدهای داده‌ای زیر هنگامی که به عنوان فیلد در رکورد گنجانده شوند می‌توانند منتهی به رکوردهایی با طول متغیر شوند: **(الف)** سن **age** **(ب)** جنس **sex** **(ج)** نام همسر **(د)** نام بچه‌ها **(ه)** میزان تحصیلات **education** **(و)** کارمندان سابق.

حل : چون **(د)** و **(و)** می‌توانند فیلدهای چندقسمتی با تعداد کم یا تعداد زیادی زیرفیلد باشند از این‌رو می‌توانند منتهی به رکوردهایی با طول متغیر شوند. علاوه بر این **(ه)** می‌تواند چند فیلدی باشد مگر آنکه، فقط بالاترین سطح تحصیلات موردنظر باشد.

مسأله ۴ـ ۱ : سیستم‌های پایگاه اطلاعات تنها به اختصار در این کتاب مطرح شده است. چرا؟

حل : چون سیستم‌های پایگاه اطلاعات بخشی از علم کامپیوتر است که به داده‌های ذخیره شده در حافظهٔ ثانویهٔ کامپیوتر مربوط می‌شود. پیاده‌سازی و تجزیه و تحلیل ساختمان داده‌های حافظهٔ ثانویه بسیار متفاوت با ساختمان داده‌های حافظهٔ اصلی کامپیوتر هستند. این کتاب اساساً ساختمان داده‌های حافظهٔ اصلی را مورد بررسی قرار می‌دهد نه حافظهٔ فرعی یا ثانویه را.

ساختمان داده‌ها و عملیات بر روی آنها

مسأله ۵ـ ۱ : هر یک از اصطلاحات زیر را به اختصار شرح دهید. **(الف)** پیمایش **(ب)** مرتب کردن و **(ج)** جستجوکردن.

حل : **(الف)** دقیقاً به یکبار دسترسی و پردازش اطلاعات هر رکورد پیمایش می‌گویند.

(ب) قراردادن داده‌ها با یک نظم و ترتیب خاص در کنار هم مرتب کردن نام دارد.

(ج) یافتن مکان رکورد با کلید یا کلیدهای معین جستجو کردن نام دارد.

مسأله ۶ـ ۱ : هر یک از اصطلاحات زیر را به اختصار شرح دهید. (الف) اضافه‌کردن رکورد (ب) حذف رکورد.

حل : (الف) افزودن یک رکورد جدید به یک ساختمان داده، که در آن معمولاً نظم قبلی حفظ می‌شود.

(ب) حذف یک رکورد خاص از ساختمان داده، حذف رکورد نام دارد.

مسأله ۷ـ ۱ : آرایهٔ خطی NAME در شکل ۱۱ـ ۱ را درنظر بگیرید که به ترتیب الفبایی مرتب است.

	NAME
1	Adams
2	Clark
3	Evans
4	Gupta
5	Jones
6	Lane
7	Pace
8	Smith

شکل ۱۱ـ ۱

(الف) NAME[2] , NAME[4] , NAME[7] را پیدا کنید.

(ب) هرگاه بخواهیم Davis را به این آرایه اضافه کنیم، چه تعداد نام بایستی به مکانهای جدید انتقال یابد؟

(ج) هرگاه بخواهیم Gupta را از این آرایه حذف کنیم، چه تعداد نام بایستی به مکانهای جدید انتقال یابد؟

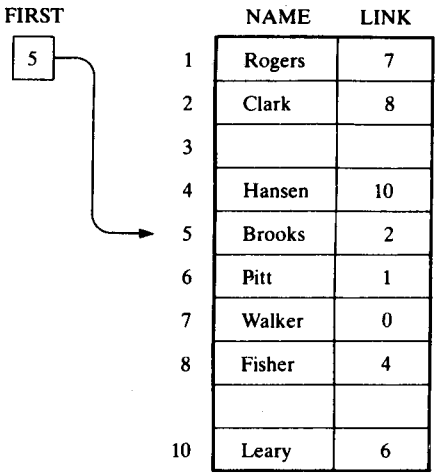
حل : (الف) در اینجا NAME[K] ، K امین نام در لیست است. از این‌رو

NAME[2] = Clark, NAME[4] = Gupta, NAME[7] = Pace

(ب) از آنجا که Davis در NAME[3] جایگزین خواهد شد، نامهای Evans تا Smith باید جابجا شوند. از این رو شش نام باید به مکان جدید انتقال یابد.

(ج) نامهای Jones تا Smith باید به بالای آرایه انتقال یابد. از این‌رو چهار نام باید جابجا شود.

مسأله ۸ـ ۱ : آرایه خطی NAME در شکل ۱۲ـ ۱ را درنظر بگیرید.



شکل ۱۲ـ ۱

مقادیر **FIRST** و **LINK[K]** در شکل، یک ترتیب خطی از نامها را به صورت زیر مشخص می‌کنند. **FIRST** به مکان اولین نام درون لیست اشاره می‌کند و **LINK[K]** به مکان نام بعدی **NAME[K]** اشاره می‌کند که در آن **0** علامت پایان لیست است. ترتیب خطی نامها را مشخص کنید.

حل : ترتیب موردنظر به صورت زیر به دست می‌آید :

FIRST = 5 ، از این رو اولین نام درون لیست **NAME[5]** یا **Brooks** است.

LINK[5] = 2 ، از این رو نام بعدی **NAME[2]** یا **Clark** است.

LINK[2] = 8 ، از این رو نام بعدی **NAME[8]** یا **Fisher** است.

LINK[8] = 4 ، از این رو نام بعدی **NAME[4]** یا **Hansen** است.

LINK[4] = 10 ، از این رو نام بعدی **NAME[10]** یا **Leary** است.

LINK[10] = 6 ، از این رو نام بعدی **NAME[6]** یا **Pitt** است.

LINK[6] = 1 ، از این رو نام بعدی **NAME[1]** یا **Rogers** است.

LINK[1] = 7 ، از این رو نام بعدی **NAME[7]** یا **Walker** است.

LINK[7] = 0 ، که نشان‌دهندهٔ پایان لیست است.

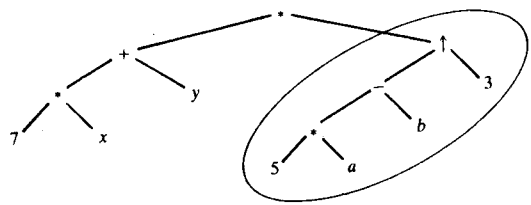
بنابراین، ترتیب خطی نامها عبارت است از:

Brooks , Clark , Fisher , Hansen , Leary , Pitt, Rogers , Walker

توجه دارید که این همان ترتیب الفبایی نامها است.

مسأله ۹ـ ۱ : عبارت جبری $(7x+y)(5a-b)^3$ را درنظر بگیرید. **(الف)** نمودار درختی متناظر با آن را مانند مثال ۵ رسم کنید. **(ب)** دامنهٔ تغییرات نماد توان‌رسانی را مشخص کنید. **(دامنهٔ تغییرات گرهٔ v در یک درخت، زیردرختی است که شامل v و گره‌های بعد از v است).**

حل : (الف) با استفاده از پیکان به طرف بالای ↑ برای توان و ستاره * به جای ضرب، درخت شکل ۱۳ـ ۱ به دست می‌آید.



شکل ۱۳ـ ۱

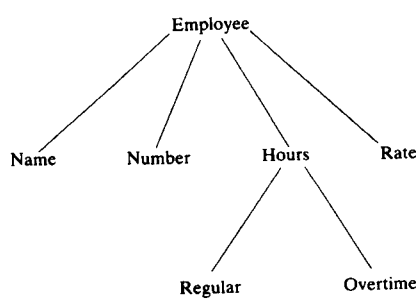
(ب) دامنهٔ تغییرات نماد توان ↑ در نمودار، زیردرختی است که با بیضی مشخص شده است و متناظر با عبارت $(5a-b)^3$ است.

مسألهٔ ۱۰ـ ۱ : اطلاعات زیر مربوط به یک ساختمان درختی است که به وسیلهٔ شماره‌های سطح آن، به صورتی که در مثال ۴ توصیف شده است مشخص شده است.

01 Employee 02 Name 02 Number 02 Hours 03 Regular 03 Overtime 02 Rate

نمودار درختی متناظر با آن را رسم کنید.

حل : نمودار درختی آن در شکل ۱۴ـ ۱ نشان داده شده است.



شکل ۱۴ـ ۱

در اینجا هر گرهٔ ۷، عنصر بعدی گره‌ای است که قبل از ۷ قرار دارد و شمارهٔ سطح آن پائین‌تر از ۷ است.

مسأله ۱۱ـ ۱ : مطلوب است بحث و بررسی این مطلب که از پشته یا صف، کدامیک ساختاری مناسب برای تعیین ترتیبی است که در آن، عناصر در هر یک از وضعیتهای زیر پردازش می‌شوند.

(الف) برنامه‌های کامپیوتری دسته‌ای **Batch** که به مرکز محاسبات ارائه می‌شود.

(ب) برنامهٔ **A** زیربرنامهٔ **B** را فرا می‌خواند که **B** به‌نوبهٔ خود، زیربرنامهٔ **C** را فرا می‌خواند و الی آخر.

(ج) کارمندان، قراردادی را به امضا می‌رسانند که در آن، روش خاص سیستم ارشدیت برای استخدام یا اخراج منظور می‌شود.

حل : (الف) صف، صرفنظر از حالتهای اولویت داده شده، برنامه‌هایی که اول از همه می‌آیند در اولین مرحله سرویس می‌گیرند.

(ب) پشته. آخرین زیربرنامه، اول اجرا می‌شود و نتیجه آن به برنامه‌ای داده می‌شود که قبل از برنامهٔ آخر است که بعد از آن اجرا می‌شود و الی آخر، تا این که اولین برنامهٔ فراخواننده اجرا شود.

(ج) پشته. در یک سیستم ارشدیت، آخرین عضو استخدامی اولین فرد قابل اخراج محسوب می‌شود.

مسألهٔ ۱۲ـ ۱ : پروازهای روزانهٔ یک شرکت هواپیمایی در شکل ۱۵ـ ۱ نشان داده شده است. **CITY** لیست شهرها و **ORIG[K]** و **DEST[K]** به ترتیب شهرهای مبدأ و مقصد پروازِ شمارهٔ **NUMBER[K]** را نمایش می‌دهد. گراف جهت‌دار متناظر با این اطلاعات را رسم کنید. (این گراف **جهت‌دار** است چون شماره‌های پرواز، پرواز از یک شهر به شهر دیگر را بدون برگشت نشان می‌دهد.)

	NUMBER	ORIG	DEST
1	701	2	3
2	702	3	2
3	705	5	3
4	708	3	4
5	711	2	5
6	712	5	2
7	713	5	1
8.	715	1	4
9	717	5	4
10	718	4	5

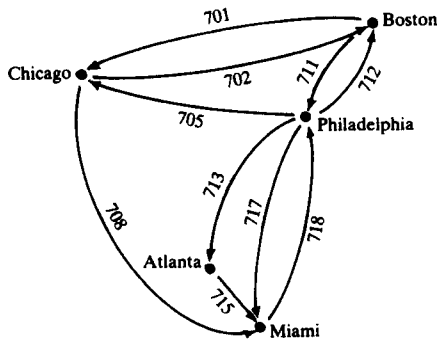
(ب)

	CITY
1	Atlanta
2	Boston
3	Chicago
4	Miami
5	Philadelphia

(الف)

شکل ۱۵ـ ۱

حل : گره‌های این گراف از پنج شهر تشکیل شده است. اگر پروازی از شهر **A** به **B** وجود داشته باشد یک پیکان از شهر **A** به شهر **B** رسم کنید و پیکان را با شمارهٔ پرواز، شماره‌گذاری کنید. گراف جهت‌دار این مسأله در شکل ۱۶ـ ۱ رسم شده است.



شکل ۱۶ـ ۱

پیچیدگی و توازن بین زمان اجرا و حافظه

مسألهٔ ۱۳ـ ۱ : هر یک از مفاهیم زیر را به اختصار شرح دهید.

(الف) پیچیدگی یک الگوریتم **(ب)** توازن بین زمان اجرا و حافظهٔ الگوریتم.

حل : (الف) پیچیدگی یک الگوریتم تابعی به صورت $f(n)$ است که مدت زمان اجرا و/ یا مقدار حافظهٔ استفاده شده توسط یک الگوریتم را برحسب تعداد داده‌های ورودی **n** اندازه می‌گیرد.

(ب) توازن بین زمان اجرا و حافظه به انتخابی در میان جوابهای الگوریتمی یک مسألهٔ پردازش داده‌ها مربوط می‌شود که اجازه می‌دهد زمان اجرای یک جواب الگوریتمی را با افزایشِ خانه‌های حافظه که داده‌ها را ذخیره می‌کند کاهش دهیم و برعکس.

مسألهٔ ۱۴ـ ۱ : فرض کنید **S** یک مجموعهٔ **n** عنصری از داده‌ها باشد.

(الف) T_1 زمان اجرای الگوریتم جستجوی خطی را با T_2 زمان اجرای الگوریتم جستجوی دودویی مقایسه کنید هرگاه **(i)** $n = 1000$ و **(ii)** $n = 10000$.

(ب) فرض کنید **S** به صورت یک لیست پیوندی ذخیره شده است. جستجوی یک عنصر مشخص در **S** را مورد بررسی قرار دهید.

حل: از بخش ۵ـ ۱ یادآور می‌شویم که زمان اجرای انتظاری الگوریتم جستجوی خطی برابر $f(n) = n / 2$

و از آن الگوریتم جستجوی دودویی برابر $f(n) = \log_2^n$ است. درنتیجه (i) به ازای $n = 1000$، $T_1 = 500$ ولی $T_2 = \log_2 1000 \approx 10$ و (ii) به‌ازای $n = 10000$، $T_1 = 5000$ ولی $T_2 = \log_2 10000 \approx 14$ است.

(ب) الگوریتم جستجوی دودویی فرض می‌کند مستقیماً می‌توان به عنصر وسط مجموعهٔ **S** دسترسی پیدا کرد اما در یک لیست پیوندی نمی‌توان مستقیماً به عنصر وسط دسترسی پیدا کرد. از این‌رو وقتی **S** به صورت یک لیست پیوندی ذخیره می‌شود بهتر است از الگوریتم جستجوی **خطی** استفاده شود.

مسألهٔ ۱۵ـ ۱ : داده‌های شکل ۱۵ـ ۱ را درنظر بگیرید که اطلاعاتی راجع به پروازهای مختلف یک خط هواپیمایی به دست می‌دهد. راههای مختلف ذخیرهٔ داده‌ها که زمان اجرا را در حالتهای زیر پائین می‌آورد مورد بررسی قرار دهید :

(الف) با معلوم بودن شماره پرواز، مبدأ و مقصد را پیدا کند.

(ب) با معلوم‌بودن دو شهر **A** و **B**، تعیین کنید که آیا پرواز از **A** به **B** وجود دارد یا خیر و در صورت مثبت‌بودن جواب، شمارهٔ پرواز آن را پیدا کنید.

حل : (الف) داده‌های شکل ۱۵ـ ۱ (ب) را در آرایه‌های **ORIG** و **DEST** ذخیره کنید که در آن همانند شکل ۱۷ـ ۱**(الف)**اندیس آرایه، شماره پرواز است.

(ب) داده‌های شکل ۱۵ـ ۱ (ب) را در آرایهٔ دوبعدی **FLIGHT** ذخیره کنید که در آن **FLIGHT[J,K]** شامل شمارهٔ پرواز از شهر **CITY[J]** به شهر **CITY[K]** است یا هرگاه هیچ پروازی صورت نگیرد شامل **0** است، که این وضعیت در شکل ۱۷ـ ۱ (ب) نشان داده شده است.

FLIGHT	1	2	3	4	5
1	0	0	0	715	0
2	0	0	701	0	711
3	0	702	0	708	0
4	0	0	0	0	718
5	713	712	705	717	0

(ب)

	ORIG	DEST
701	2	3
702	3	2
703	0	0
704	0	0
705	5	3
706	0	0
⋮	⋮	⋮
715	1	4
716	0	0
717	5	4
718	4	5

(الف)

شکل ۱۷ـ ۱

مسألهٔ ۱۶ـ ۱ : فرض کنید یک شرکت هواپیمایی به **n** شهر **s** پرواز عرضه می‌کند. معایب نمایش داده‌ها را به صورتی که در شکل ۱۷ـ ۱ (الف) و شکل ۱۷ـ ۱ (ب) آمده است شرح دهید.

حل : (الف) فرض کنید شماره‌های پرواز با فاصله خیلی زیاد از آدرس صفر، در حافظه ذخیره شده است یعنی فرض کنید نسبت شماره‌های پرواز **s** به شمارهٔ خانه‌های حافظه عدد بسیار کوچکی مثلاً تقریباً **0.05** باشد. در آن صورت است که گرفتن فضای حافظهٔ اضافی ممکن است مقرون به‌صرفه نباشد.

(ب) فرض کنید نسبت شماره پروازهای **s** به شماره خانه‌های حافظه **n** در آرایهٔ **FLIGHT** عدد بسیار کوچکی است یعنی آرایهٔ **FLIGHT** به‌گونه‌ای است که شامل تعداد بسیار زیادی صفر است. (به این‌گونه آرایه‌ها **ماتریس خلوت** یا **تُنُک** می‌گویند). در آن صورت است که گرفتن فضای حافظهٔ اضافی ممکن است مقرون به‌صرفه نباشد.

فصل ۲

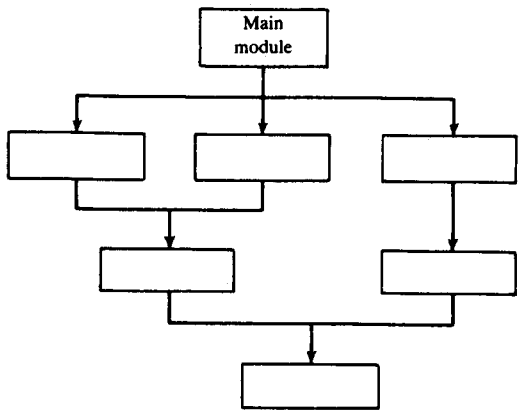
مفاهیم مقدماتی

۱ـ ۲ مقدمه

به‌دست آوردن الگوریتم‌هایی برای ایجاد و پردازش ساختمان داده‌ها، ویژگی اصلی این کتاب را تشکیل می‌دهد. در این فصل با مثالهای ساده، فُرمتهایی را توضیح می‌دهیم که به کمک آنها به ارائه الگوریتم‌ها می‌پردازیم. فُرمَتی که ما برای ارائهٔ الگوریتم‌ها انتخاب کرده‌ایم مشابهٔ فرمتی است که **دانلد کناث** **D.Knuth** از آنها در کتاب معروفش، **الگوریتم‌های بنیادی** استفاده کرده است. اگرچه فرمت مورد استفادهٔ ما مستقل از زبان برنامه‌نویسی است اما به‌اندازهٔ کافی ساختیافته بوده، جزئیات آن بیان شده است و به‌سادگی می‌توان آنها را به یکی از زبان‌های برنامه‌نویسی نظیر **PASCAL** ، **C** ، **FORTRAN** یا **BASIC** ترجمه کرد. شایان ذکر است که برخی از الگوریتم‌های کتاب، در بخش مسایل به این زبانها نوشته شده است.

الگوریتم‌ها تا حدودی پیچیده هستند. درک پیاده‌سازی الگوریتم‌های پیچیده به صورت برنامه‌های کامپیوتری، زمانی با سادگی زیاد امکان‌پذیر است که بتوان این برنامه‌ها را به چند قطعه برنامه آن هم به صورت سلسله مراتبی، مشابهٔ آنچه که در شکل ۱ـ ۲ نشان داده شده است سازماندهی کرد.

در یک چنین سازماندهی‌ای، در اول هر برنامه قطعه برنامهٔ اصلی قرار دارد که الگوریتم را به صورت کلی شرح می‌دهد. قطعه برنامهٔ اصلی چند زیربرنامه را احضار می‌کند که در آنها جزئیات بیشتری راجع



شکل ۱ـ ۲ سلسله مراتب قطعه برنامه‌ها

به کار برنامهٔ اصلی گنجانده شده است. هر زیربرنامه ممکن است به زیربرنامه‌های دیگری که دارای جزئیات بیشتری هستند رجوع کنند و الی آخر. سازماندهی یک برنامه به صورت چنین سلسله مراتبی از قطعه برنامه‌ها، معمولاً مستلزم استفاده از فلوچارتهای مختلف و دستورهای منطقی است که عموماً در هر زبان برنامه‌نویسیِ ساختیافته وجود دارد. در این فصل چند فلوچارت و دستورهای منطقی از نظر گذرانده می‌شود.

فصل حاضر با شرح مختصری از چند تابع ریاضی شروع می‌شود که در مطالعه الگوریتم‌ها و به‌طور کلی در علوم کامپیوتر با آن‌ها مواجه می‌شوید و با بحث روی انواع مختلف متغیرهایی که می‌توانند در الگوریتم‌ها و برنامه‌ها ظاهر شوند به پایان می‌رسد.

مفهوم پیچیدگی یک الگوریتم نیز در این فصل گنجانده شده است. این معیار بسیار مهم در الگوریتم‌ها، ابزاری را دراختیار دانشجو قرار می‌دهد تا بتواند جوابهای مختلف الگوریتمی مسأله خاص نظیر جستجو یا مرتب‌کردن اطلاعات را با هم مقایسه کند. مفهوم یک الگوریتم و پیچیدگی آن نه‌تنها در درس ساختمان داده‌ها، بلکه تقریباً در تمام شاخه‌های علوم کامپیوتر از اهمیت اساسی و بنیادی برخوردار است.

۲ـ ۲ نمادگذاری ریاضی و تابعهای کامپیوتری

در این بخش چند تابع ریاضی معرفی می‌شود که به همراه نمادشان بکرات در تجزیه و تحلیل

الگوریتم‌ها و به‌طور کلی در علم کامپیوتر ظاهر می‌شوند.

تابعهای کف و سقف

فرض کنید x یک عدد اعشاری باشد. در این صورت x بین دو عدد صحیح قرار دارد که **کف x** و **سقف x** نامیده می‌شوند. به‌خصوص این که :

$\lfloor x \rfloor$ که **کف x** نامیده می‌شود، بزرگترین عدد صحیحی را نشان می‌دهد که بزرگتر از x نباشد.

$\lceil x \rceil$ که **سقف x** نامیده می‌شود کوچکترین عدد صحیحی را نشان می‌دهد که کوچکتر از x نباشد.

اگر x یک عدد صحیح باشد، آنگاه $\lceil x \rceil = \lfloor x \rfloor$ در غیر اینصورت $\lceil x \rceil = \lfloor x \rfloor + 1$.

مثال ۱ـ ۲

$$\lfloor 3.14 \rfloor = 3, \quad \lfloor \sqrt{5} \rfloor = 2, \quad \lfloor -8.5 \rfloor = -9, \quad \lfloor 7 \rfloor = 7$$
$$\lceil 3.14 \rceil = 4, \quad \lceil \sqrt{5} \rceil = 3, \quad \lceil -8.5 \rceil = -8, \quad \lceil 7 \rceil = 7$$

تابع باقیمانده ، حساب باقیمانده

فرض کنید K یک عدد صحیح دلخواه و M یک عدد صحیح مثبت باشد. آنگاه

$$k \pmod M$$

(بخوانید **باقیمانده k بر M**) هنگام تقسیم k بر M برای نمایش باقیماندهٔ صحیح به‌کار می‌رود. به بیان دقیق‌تر، $K \pmod M$ برابر عدد صحیح منحصر بفرد r است طوری که

$$K = M q + r \qquad \text{که در آن} \qquad 0 \le r < M$$

هرگاه K مثبت باشد تنها با تقسیم K بر M باقیماندهٔ r به دست می‌آید. بنابراین :

$$25 \pmod 7 = 4, \quad 25 \pmod 5 = 0, \quad 35 \pmod{11} = 2, \quad 3 \pmod 8 = 3$$

در حالتی که k منفی باشد مسأله ۲ (ب) روشی را برای تعیین $K \pmod M$ نشان می‌دهد. از اصطلاح **mod** در رابطهٔ ریاضی همنهشتی نیز استفاده می‌شود که به صورت زیر نمایش داده‌شده، تعریف می‌شود:

$a \equiv b \pmod M$ اگر و فقط اگر M یک عامل $b - a$ باشد.

در اینجا M **سنج** یا **هنگ** نامیده می‌شود و $a \equiv b \pmod M$ به صورت " a همنهشت است با b به سنج M" خوانده‌می‌شود. ویژگی‌های زیر از رابطهٔ همنهشتی اغلب مفید هستند :

$$a \pm M \equiv a \pmod M \quad \text{و} \quad 0 \equiv M \pmod M$$

منظور از **حساب با باقیماندهٔ M** عملیات حسابی جمع، ضرب و تفریق است که در آن مقدار حسابی، جایگزین مقدار معادل آن در مجموعهٔ

$$\{0 , 1 , 2 , \dots , M - 1\}$$