فصل ۲

الگوریتم و فلوچارت

۱.۲ الگوریتم

۱.۱.۲ الگوریتم چیست؟

شاید برایتان پیش آمده باشد که دستگاه جدیدی خریداری کرده باشید که با کارکرد آن و روش استفاده از آن آشنایی ندارید. در این خصوص، اولین کاری که انجام می‌دهید آن است که راهنمای کاربری آن را بخوانید و بر مبنای اطلاعات موجود در آن چگونگی استفاده از دستگاه را یاد بگیرید. این راهنمای کاربری، توسط فردی نوشته شده است که به کارکرد دستگاه تسلط دارد و علاوه بر این به گونه‌ای نوشته شده است که هر کاربر دستگاه، با سواد و اطلاعات فنی محدود خود می‌تواند با استفاده از آن راهنما، از دستگاه استفاده کند.

یک روش برای بیان شیوه انجام یک کار، ارائه دستورالعمل‌های ساده و مرحله‌ای است که شخص کاربر می‌تواند آن دستورات را به‌ترتیب انجام دهد تا به نتیجهٔ مورد نظر برسد. این دستورالعمل اولاً باید شامل مراحلی باشد که هرکدام توسط کاربر معمولی قابل فهم باشد و امکان اجرایی آن را داشته باشد. ثانیاً ترتیب انجام دستورالعمل‌ها در آن مشخص باشد و در نهایت با انجام دستورالعمل‌ها، کار موردنظر به درستی انجام شود. ثالثاً باید تمام

حالاتی که ممکن است در اجرای دستورات رخ دهد درنظر گرفته شود و چگونگی ادامه انجام دستورالعمل‌ها در آن حالت بیان شود به عبارت دیگر، دستورالعمل باید جامع و کامل باشد و کاربر درحین اجرای آن به حالتی برخورد نکند که در دستورالعمل مشخص نشده باشد در آن حالت در مرحلهٔ بعد چه کاری باید انجام شود. شاید برخورد کرده باشید که در حین دنبال کردن دستورالعمل‌های یک دستگاه با پیغام خطایی مواجه می‌شوید که در دستورالعمل دستگاه اشاره‌ای به آن نشده است. دراین صورت دستورالعمل جامع و کامل نیست و درمواردی به نتیجهٔ مورد نظر نمی‌رسد.

این شکل بیان راه حل توسط خوارزمی، ریاضی‌دان، ستاره‌شناس و جغرافی‌دان شهیر ایرانی، در کتاب ''الجبر و المقابله'' جهت بیان راه حل برخی مسائل استفاده شد و به همین دلیل به نام وی''الخوارزمی'' یا ''الگوریتم'' نامیده می‌شود. پس منظور از الگوریتم بیان راه‌حل (برای انجام یک کار یا یک مسئله یا...) به صورت دنباله‌ای از دستورالعمل‌های ساده است که به انجام آن کار یا حل آن مسئله منجر شود.

تا همین‌جا، مشکلات بیان الگوریتمی برای یک راه‌حل مشخص می‌شود. از بین آن‌ها می‌توان به لزوم تشکیل آن از دستورات ساده و قابل فهم، در نظر گرفتن تمام حالاتی که ممکن است اتفاق بیفتد و همچنین درستی آن اشاره کرد. به‌همین دلیل کار طراحی یک الگوریتم برای یک مسئله (مثلاً نوشتن راهنمای کاربردی برای یک دستگاه) کار پیچیده‌ای است که جوانب مختلفی دارد.

با این مقدمه، تعریف زیر را برای الگوریتم می‌آوریم.

تعریف ۱.۱.۲ . *(الگوریتم) یک الگوریتم، دنباله‌ای از دستورالعمل‌ها با ویژگی‌های زیر است منجر به حل یک مسئله می‌شود:*

۱) ترتیب انجام دستورات کاملاً مشخص است.

۲) دستورات برای مجری الگوریتم کاملاً قابل فهم و اجراست.

۳) پایان‌پذیر است یعنی بعد از تعداد متناهی گام به نتیجه می‌رسد.

با بررسی این تعریف مشخص می‌شود که برای بیان الگوریتم حل یک مسئله باید موارد زیر رعایت شود. اولاً باید مجری الگوریتم کاملاً شناخته شده باشد و بدانیم امکان فهم و

انجام چه کارهایی را دارد. ثانیاً باید یک راه‌حل گام به گام برای حل مسئله مورد نظر با استفاده از دستوراتی که مجری الگوریتم امکان اجرای آن را داشته باشد پیدا کنیم. به عنوان مثال، اگر الگوریتم برای حل یک مسئله توسط یک کودک دبستانی نوشته شود، نمی‌توان از دستوری استفاده کرد که انتگرال یک تابع را محاسبه کند، زیرا توسط مجری الگوریتم قابل فهم و اجرا نیست ولی در بیان الگوریتمی که مجری آن یک دانشجوی ریاضی است می‌توان از چنین دستوری استفاده کرد. دقت کنید که یک وجه از قابل فهم بودن دستورات، این است که دستورات به زبان قابل فهم برای مجری بیان شوند. ثالثاً روش ارائه شده باید اتمام‌پذیر باشد و در تعداد گام‌های متناهی به جواب مسئله برسد و البته جواب درستی برای مسئله پیدا و ارائه کند.

۲.۱.۲ الگوریتم و کامپیوتر

الگوریتم‌ها صدها سال قبل از اختراع کامپیوتر معرفی و استفاده شده‌اند درحالی‌که امروزه تقریباً نام الگوریتم با کامپیوتر گره خورده است و اولین چیزی که با شنیدن نام الگوریتم به ذهن می‌آید کامپیوتر است. باتوجه به مطالب بخش قبل، عملاً الگوریتم وابستگی به کامپیوتر ندارد بلکه از الگوریتم می‌توانیم برای بیان راه‌حل یک مسئله برای کامپیوتر استفاده کنیم.

برای حل یک مسئله توسط کامپیوتر، باید روشی برای حل آن مسئله با توجه به قابلیت‌های کامپیوتر بیان شود. به‌عبارت دیگر، برای حل یک مسئله یا انجام یک کار توسط کامپیوتر، باید الگوریتمی برای آن ارائه شود که شامل دستورالعمل‌هایی باشد که توسط کامپیوتر قابل اجرا باشد. لذا برای حل یک مسئله توسط کامپیوتر، ابتدا باید قابلیت‌ها و محدودیت‌های کامپیوترهایی که در اختیار داریم را بشناسیم تا بتوانیم الگوریتم حل مسئله را با دستورات مناسب بیان کنیم.

به طور کلی می‌توان گفت که هر کامپیوتر امروزی توانایی انجام چهارعمل اصلی (جمع، تفریق، ضرب، تقسیم) مقایسهٔ دو عدد، قراردادن مقداری درحافظه (انتساب)، گرفتن مقداری از کاربر و اعلام مقداری به کاربر را دارد و همچنین می‌تواند مقادیری را درحافظه نگهداری کند تا در محاسبات آتی از آنها استفاده کند. اما کامپیوترهای فعلی قابلیت استدلال و

استنتاج را ندارند، یعنی نمی‌توانند با استدلال و استنتاج از یک سری نتایج، نتیجهٔ جدیدی را به‌دست آورند. این فرآیندی است که به ''هوش'' مربوط بوده و در حال حاضر، مختص انسان است.

بنابراین، اگر مجری یک الگوریتم، کامپیوتر باشد باید از دستورالعمل‌هایی استفاده کنیم که توسط کامپیوتر قابل اجرا باشد. به‌عبارت دقیق‌تر، برای حل هر مسئله، هرچند پیچیده، باید روشی ارائه کرد که با انجام چهارعمل اصلی روی اعداد، مقایسه و سایر اعمال ساده دیگر قابل انجام توسط کامپیوتر، به حل مسئله منجر شود. درگام اول، الگوریتمی با استفاده از دستورات با خواص ذکر شده اما به زبان خودمان به‌دست می‌آوریم. درگام بعدی و برای اجرای الگوریتم توسط کامپیوتر، باید الگوریتم به زبان قابل فهم توسط کامپیوتر، که ما آن را زبان برنامه‌نویسی می‌نامیم، بیان کنیم تا توسط کامپیوتر قابل فهم و اجرا باشد. در علوم کامپیوتر، گام اول را طراحی الگوریتم و گام دوم را برنامه‌نویسی گویند.

با این ترتیب، بدیهی است برای حل یک مسئله با استفاده از کامپیوتر، ابتدا باید الگوریتمی برای حل آن مسئله پیدا کنیم که برمبنای دستورات ساده، توسط کامپیوتر قابل اجرا باشد. شاید بتوان گفت این مشکل‌ترین گام است. پس از پیداکردن روش، این روش به یک زبان برنامه‌نویسی بیان می‌شود تا توسط کامپیوتر قابل اجرا باشد. این گام چیزی مشابه ترجمه متن از یک زبان به زبان دیگر است.

در این کتاب، به ترتیب به این گام‌ها می‌پردازیم. یعنی ابتدا برای به دست آوردن الگوریتم برای حل مسائل تمرکز می‌کنیم و سپس به بحث تبدیل الگوریتم به برنامه یا برنامه‌نویسی می‌پردازیم. سعی می‌کنیم مباحث را بر مبنای مثال‌هایی برای خوانندگان بیان کنیم اما در خصوص الگوریتم، عملاً فرآیند طراحی یک الگوریتم برای یک مسئله کاملاً بر مبنای هوش افراد است و چنین نیست که با دیدن تعدادی الگوریتم برای حل مسائل مختلف، بتوان طراحی الگوریتم را فراگرفت بلکه لازم است که فرد روی مسائل فکر کند و برای آن راه‌حل به دست آورد. این فرآیند مشابه حل یک انتگرال است. درست است که دیدن روش حل انتگرال‌های مختلف در یادگیری روش انتگرال‌گیری مناسب است اما با خواندن حل چند انتگرال، نمی‌توان انتظار داشت که بتوان هرانتگرالی را حل کرد. در فرآیند حل انتگرال لازم

است فرد راه‌حلی برای آن پیدا کند که کاملاً بر مبنای تفکر فرد است.

از این دید، قرارگرفتن شاخهٔ طراحی الگوریتم به‌عنوان زیرشاخه‌ای از ریاضیات کاملاً قابل توجیه است. خواننده محترم باید کاملاً آشنا به این موضوع باشد تا روند آموزش را برمبنای تسلط بر تکنیک‌ها و توسعهٔ قابلیت تفکر و نوآوری دنبال کند. هرنوع دیگر آموزش از قبیل خواندن و به خاطر سپردن الگوریتم‌های حل مسائل مختلف، کمکی در ارائه الگوریتم برای مسئله جدید نمی‌کند، بلکه برای هرمسئله باید روشی مناسب با آن مسئله ارائه کرد که فرآیندی کاملاً تفکری است. در ادامه سعی شده است از طریق مثال‌ها و تمرینات مختلف، به خواننده در دستیابی به این مهارت کمک شود.

مثال ۱.۱.۲. *الگوریتمی ارائه کنید که محیط یک مستطیل به طول ۴ و عرض ۳ را محاسبه و چاپ کند.*

حل: با توجه به این‌که طبق قوانین هندسه، محیط یک مستطیل، دو برابر مجموع طول و عرض آن است، الگوریتم سادهٔ زیر این کار را انجام می‌دهد.

الگوریتم ۱.۲: الگوریتم محاسبهٔ محیط یک مستطیل ۴×۳

۱ **شروع**
۲ $(۴+۳) \times ۲$ **را چاپ کن**
۳ **پایان**

□

الگوریتم ساده فوق، محیط یک مستطیل ۴×۳ را برای ما محاسبه و چاپ می‌کند. البته با توجه به این که کامپیوتر قابلیت گرفتن مقدار از کاربر را دارد (ورودی) می‌توان مسئله را به این صورت تعمیم داد که طول و عرض مستطیل را از کاربر دریافت کرده و سپس محیط مستطیل را محاسبه و چاپ کند.

همانطور که قبلاً بیان شد، یکی از قابلیت‌های کامپیوتر، امکان ذخیره‌سازی داده‌ها در حافظه‌اش است به‌گونه‌ای که برای محاسبات قابل استفاده باشد. در اغلب الگوریتم‌ها، ما

نیاز داریم داده‌هایی که از ورودی دریافت می‌شوند و یا داده‌هایی که حاصل از محاسبات میانی هستند را در محلی از حافظه ذخیره کنیم تا برای محاسبات بعدی استفاده شود.

برای ایجاد امکان بازیابی داده‌ها، داده‌ها را با یک نام ذخیره می‌کنیم و برای ارجاع به آن داده از نام آن استفاده می‌کنیم. با توجه به این که در یک محل حافظه می‌توان داده‌های مختلف را نگهداری کرد، البته در زمان‌های مختلف و نه همزمان، ما از نام "متغیر" برای محلی که امکان ذخیره داده را دارد استفاده می‌کنیم. لذا یک متغیر، محلی است که نامی برای آن در نظر گرفته شده است و با استفاده از نام آن می‌توان اطلاعاتی در آن ذخیره کرد یا اطلاعات ذخیره‌شدهٔ در آن را بازیابی و استفاده نمود.

درمثال زیر، مسئله محاسبهٔ محیط یک مستطیل کمی توسیع داده شده است به صورتی که طول و عرض مستطیل از کاربر دریافت شده و درمحلی از حافظه ذخیره می‌شود و سپس محیط مستطیل با استفاده از طول و عرض وارد شده و ذخیره شده در محل‌های حافظه محاسبه شده و به خروجی منتقل می‌شود. برای این منظور از متغیری به نام A برای نگهداری طول و متغیری به نام B برای نگهداری عرض مستطیل استفاده شده است.

مثال ۲.۱.۲. *الگوریتمی ارائه کنید که طول و عرض یک مستطیل را از ورودی دریافت کند و محیط مستطیل را محاسبه و چاپ کند.*

حل:

الگوریتم ۲.۲: الگوریتم دریافت طول و عرض یک مستطیل و محاسبهٔ محیط آن

۱ **شروع**
۲ طول و عرض مستطیل **را بگیر** و به ترتیب آنها را در متغیرهای A و B قرار بده.
۳ $۲ \times (A + B)$ **را چاپ کن**
۴ **پایان**

□

در خصوص متغیرها می‌توانیم اعمال زیر را روی آنها انجام دهیم:

۱. ذخیره یک مقدار در یک متغیر. این مقدار ممکن است مستقیماً از ورودی دریافت

شده باشد یا از طریق انجام محاسبات روی داده‌های ورودی به‌دست آمده باشد. درمثال فوق می‌توانستیم خط ۳ را با دو خط زیر جایگزین کنیم بدین صورت که ابتدا محیط را محاسبه و در متغیری به نام C ذخیره کند و سپس مقدار متغیر را درخروجی چاپ کند.

۳- قرار بده $C \leftarrow ۲ \times (A + B)$

۴- C را چاپ کن.

دقت کنید که در مرحلهٔ ۲، از نماد $\leftarrow$ برای قراردادن مقدار حاصل از یک یا چند محاسبه در یک متغیر استفاده کردیم. این عمل را «انتساب» می‌نامند. لازم به ذکر است که سمت چپ علامت انتساب، نام یک متغیر آورده می‌شود ولی سمت راست آن می‌تواند یک مقدار ثابت، یک متغیر یا یک عبارت محاسباتی باشد.

۲. خواندن مقدار ذخیره شده در یک متغیر. این مقدار ممکن است جهت محاسبات استفاده شود یا مستقیماً به خروجی منتقل شود.

۳. مقایسهٔ محتوای متغیرها گاهی لازم است بررسی کنیم که دربین دو مقدار کدام بزرگ‌تر است. امکان انجام این کار توسط کامپیوتر وجود دارد.

مثال ۳.۱.۲. *الگوریتمی ارائه کنید که سه عدد را از ورودی بگیرد و مجموع و میانگین آنها را محاسبه کرده و در خروجی چاپ کند.*

حل: الگوریتم حل این مسئله در الگوریتم ۳.۲ آمده است. □

نکته‌ای که در به کاربردن متغیرها باید در نظر قرار گیرد این است که یک متغیر امکان نگهداری تنها یک مقدار را دارد. لذا اگر در متغیری مقداری مانند ۱ ذخیره شده باشد و در حین اجرای الگوریتم، مقدار دیگری مانند ۲ در همان متغیر ذخیره شود، مقدار قبلی متغیر حذف شده و مقدار جدید جایگزین می‌شود. در صورت جایگزینی مقدار یک متغیر، مقدار قبلی به هیچ عنوان قابل برگشت یا محاسبه نیست. بنابراین اگر داده‌ای در متغیری ذخیره شده است که آن مقدار مبنای انجام محاسبات است، باید در نگهداری آن دقت شود.

الگوریتم ۳.۲: الگوریتم محاسبهٔ مجموع و میانگین سه عدد

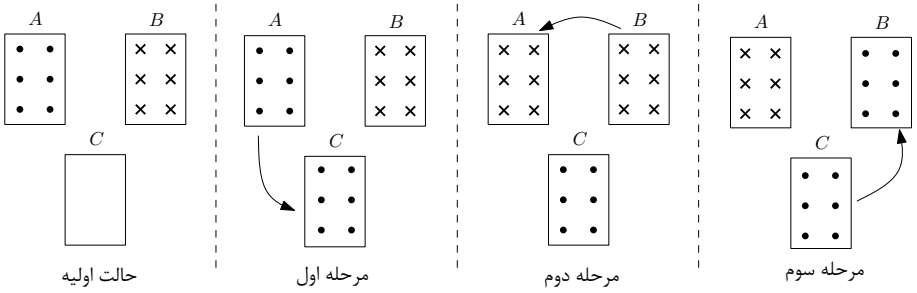
۱ **شروع**
۲ سه عدد را از ورودی **را بگیر** و آن را در متغیرهای A و B و C قرار بده.
۳ قرار بده $S \leftarrow A + B + C$ (جمع اعداد توسط کامپیوتر قابل انجام است و حاصل در S ذخیره شده است)
۴ قرار بده $AVE \leftarrow S/۳$ (تقسیم دو عدد توسط کامپیوتر قابل انجام است و حاصل در AVE ذخیره شده است)
۵ S و AVE **را چاپ کن**
۶ **پایان**

این نکته از این دید حائز اهمیت است که این فرآیند با فرآیند ذهنی ما درخصوص محل نگهداری اشیاء متفاوت است. به‌عنوان مثال اگر لیوان پر از مایعی داشته باشیم و مقدار دیگری مایع به آن اضافه کنیم، لیوان سرریز می‌شود. درخصوص متغیرها، این اتفاق نمی‌افتد، بلکه محتوای قبلی لیوان حذف و مایع جدید جایگزین قبلی خواهد شد. به عبارت دقیق‌تر، اگر لیوانی پر از مایع قرمز داشته باشیم و به آن لیوانی پر از مایع آبی اضافه کنیم، در واقعیت، لیوان سرزیر می‌کند و رنگ محتوا نیز از ترکیب دو رنگ آبی و قرمز تشکیل خواهد شد. درحالی که اگر لیوان مایع قرمز را به منزلهٔ یک متغیر در کامپیوتر درنظر بگیریم، با اضافه‌شدن لیوان مایع آبی به آن، محتوای قبلی کاملاً پاک شده و لیوان حاوی مایع آبی خواهد بود و محتوای قبلی لیوان تأثیری روی محتوای جدید نخواهد داشت. این دقیقاً اتفاقی است که در قراردادن مقادیر در متغیرهای کامپیوتر اتفاق می‌افتد و لازم است طراح الگوریتم در استفاده از متغیرها به آن توجه داشته باشد تا باعث اشتباه نشود. دقت کنید که شرط اول استفاده صحیح از یک دستگاه، دانستن شیوه کارکرد آن است و لذا طراح الگوریتمی که قرار است مجری آن کامپیوتر باشد باید با جزئیات عملکرد یک کامپیوتر در موارد مختلف آشنا باشد.

مثال زیر شاید به فهم این مطلب کمک کند.

مثال ۴.۱.۲. *الگوریتمی ارائه کنید که دو مقدار را گرفته و در دو متغیر ذخیره کند و سپس محتوای دو متغیر را جابجا کند.*

حل: برای درک بهتر مسئله فرض کنید دو لیوان یکی، به نام A، پر از مایع قرمز و دیگری، به نام B، پر از مایع آبی داریم. اگر بخواهیم محتوای این دو لیوان را جابجا کنیم، ساده‌ترین راه، استفاده از یک لیوان سوم خالی، به نام C، به عنوان ظرف کمکی است. محتوای لیوان A را در لیوان C میریزیم، سپس محتوای لیوان B را در لیوان A می‌ریزیم و در نهایت محتوای لیوان C را در لیوان B می‌ریزیم (شکل ۱.۲ را ببینید.).



شکل ۱.۲: مراحل جابجایی محتوای دو متغیر با استفاده از متغیر کمکی.

همین روش را برای جابجا کردن محتوای دو متغیر به کار می‌بریم (الگوریتم ۴.۲ را ببینید).

دقت کنید در مثال جابجایی محتوای دو لیوان، چاره‌ای جز استفاده از یک لیوان کمکی نیست ولی در خصوص متغیرها می‌توان حتی بدون متغیر کمکی هم این کار را انجام داد و علت این امر آن است که در انتقال محتوای لیوان‌ها، اگر به لیوان پر، چیز جدیدی اضافه شود، سرزیر می‌کند و ماده از دست می‌رود ولی درخصوص متغیرها همچنان که قبلاً بیان شد، مقدار قبلی از بین می‌رود و مقدار جدید جایگزین می‌شود. الگوریتم زیر، محتوای دومتغیر را بدون استفاده از متغیر کمکی جابجا می‌کند.

□

الگوریتم ۴.۲: الگوریتم جابجایی محتوای دو متغیر

۱ **شروع**
۲ دو عدد را از ورودی **را بگیر** و آن را در متغیرهای A و B قرار بده.
۳ $C \leftarrow A$
۴ $A \leftarrow B$
۵ $B \leftarrow C$
۶ **پایان**

الگوریتم ۵.۲: الگوریتم جابجایی محتوای دو متغیر بدون استفاده از متغیر کمکی

۱ **شروع**
۲ دو عدد را از ورودی **را بگیر** و آن را در متغیرهای A و B قرار بده.
۳ $A \leftarrow A + B$
۴ $B \leftarrow A - B$
۵ $A \leftarrow A - B$
۶ **پایان**

برای دیدن درستی کارکرد الگوریتم می‌توانید آن را با دو مقدار فرضی امتحان کنید. مثلاً فرض کنید کاربر اعداد ۲ و ۵ را وارد کرده است و ۲ در متغیر A و ۵ در متغیر B قرارگرفته است. درخط ۳ الگوریتم، مجموع $A + B$ درمتغیر A قرارمی‌گیرد و لذا مقدار ۷ درمتغیر A قرار می‌گیرد. دقت کنید که A از این لحظه مساوی ۷ است و نه ۲. سپس درخط ۳ الگوریتم مقدار $A - B$ در B قرار می‌گیرد، یعنی مقدار ۲ در B قرار می‌گیرد. درنهایت مقدار $A - B$ از A قرار می‌گیرد. در لحظهٔ اجرای این دستور A حاوی ۷ و B حاوی ۲ است و لذا مقدار ۵ در A قرار می‌گیرد. با این ترتیب A حاوی ۵ و B حاوی ۲ است که دقیقاً جابجایی محتوای A و B را نشان می‌دهد.

ما معمولاً توضیحات یک اجرای آزمایشی الگوریتم را درقالب جدولی به‌صورت زیر می‌آوریم. این جدول دارای ستون‌های مختلف است که شامل شماره خط الگوریتم و مقدار متغیرهای الگوریتم در لحظهٔ اجرای شماره خط است. توضیحات قبلی به‌صورت جدول زیر قابل نمایش است.

شماره خط الگوریتم	A	B
۱		
۲	۲	۵
۳	۷	
۴		۲
۵	۵	

در خطوطی که مقدار مشخص نشده است به معنی آن است که مقدار قبلی متغیر تغییری نکرده است.

از روش فوق معمولاً برای آزمایش درستی الگوریتم‌ها استفاده می‌کنیم. درصورتی که کارکرد الگوریتم با یک یا چند ورودی، درست باشد، تقریباً از درستی آن اطمینان حاصل می‌کنیم و درصورت اشتباه بودن جواب الگوریتم، نسبت به تصحیح آن و بررسی مجدد درستی آن اقدام می‌کنیم. تأکید می‌شود که این بررسی فقط تا حدی درستی الگوریتم را نشان می‌دهد و اثبات درستی الگوریتم نیست. اثبات درستی الگوریتم معمولاً کار مشکلی است و به استفاده از استدلال‌های ریاضی نیاز دارد.

نکتهٔ دیگر در این خصوص این است که برای حل یک مسئله، یعنی جابجایی محتوای دومتغیر، دو روش کاملاً متفاوت ارائه شد که هر دو درست است. لذا مشابه حل مسائل ریاضی که ممکن است افراد مختلف، آن‌ها را به روش‌های مختلف حل کنند، برای حل مسائل نیز ممکن است الگوریتم‌های مختلفی وجود داشته باشد که همگی آن مسئله را به درستی حل کنند.

در ادامه، گاهی جملات را در الگوریتم‌ها خلاصه می‌کنیم. مثلاً به‌جای ”یک عدد را از ورودی بگیر و درمتغیر A قرار بده“ می‌نویسیم ”A را بخوان“ و یا به جای ”مقدار متغیر B

را در خروجی چاپ کن" از "B را چاپ کن" استفاده می‌کنیم. دقت کنید که درست است که درمثال‌های قبلی، هرالگوریتم با **شروع** آغاز و با **پایان** به اتمام می‌رسد ولی در آینده مواردی خواهیم داشت که روند متفاوتی دارد. لذا هرچند شاید به‌ظاهر زاید به‌نظر آید ولی حتماً محل شروع و پایان الگوریتم را با این دستورات مشخص می‌کنیم.

تمرین

۱. الگوریتمی ارائه کنید که شعاع یک دایره را از ورودی بگیرد و محیط و مساحت آن را محاسبه کرده و درخروجی چاپ کند.
۲. الگوریتمی بنویسید که دو عدد را از ورودی دریافت کرده و حاصل چهار عمل اصلی روی اعداد وارد شده را محاسبه کرده و در خروجی چاپ کند.
۳. الگوریتمی بنویسید که پنج عدد را از ورودی دریافت کرده و مجموع و میانگین آنها را محاسبه کرده و در خروجی چاپ کند.

۲.۲ کامپیوتر: قابلیت‌ها و محدودیت‌ها

همانطور که قبلاً نیز اشاره شد، لازمهٔ نوشتن الگوریتم، دانستن قابلیت‌ها و محدودیت‌های مجری الگوریتم است. در اینجا مجری الگوریتم یک کامپیوتر است و لذا در این بخش به‌طور دقیق‌تر به بیان قابلیت‌ها و محدودیت‌هایی که یک کامپیوتر دارد می‌پردازیم.

قابلیت‌ها:

۱. انجام عملیات محاسباتی در زمان کوتاه. در کل یک کامپیوتر می‌تواند چهارعمل اصلی را بسیار سریع انجام دهد. البته گاهی در نوشتن الگوریتم‌ها پا را از این فراتر گذاشته و فرض می‌کنیم که کامپیوتر می‌تواند عملیاتی نظیر قدرمطلق، جذر و سایر عملیات پایه‌ای دیگر را نیز انجام دهد. هرچند این عملیات مستقیماً توسط پردازندهٔ کامپیوتر قابل انجام نیست ولی معمولاً زبان‌های برنامه‌نویسی که ما بعداً برای اجرای

الگوریتم‌های خود روی کامپیوتر از آن بهره خواهیم گرفت این عملیات اضافی را نیز پشتیبانی می‌کنند. منظور ما از پشتیبانی کردن این است که روش‌های از پیش تعریف شده در این زبان‌ها وجود دارد که این عملیات را با استفاده از چهارعمل اصلی برای ما انجام می‌دهد و نیازی نیست ما خودمان مثلاً روش محاسبهٔ جذر یک عدد را به‌صورت الگوریتم بنویسیم. البته تلاش ما این خواهد بود که حتی‌الامکان از دستوراتی استفاده کنیم که توسط کامپیوتر مستقیماً قابل اجرا باشد یا به طریقی اطمینان حاصل کنیم اجرای آن امکان‌پذیر است و جزئیات آن را حذف کنیم.

۲. امکان نگهداری داده‌ها (حافظه). شاید یک خاصیت متمایزکننده کامپیوترها از ماشین‌حساب‌های قدیمی، امکان ذخیره اطلاعات و اعداد درحافظه برای استفاده در آینده است. در عمل نیز از این خاصیت کامپیوترها بسیار استفاده می‌شود. به‌صورت روزافزون بانک‌های اطلاعاتی ادارات به‌صورت الکترونیک درآمده و در کامپیوترها ذخیره می‌شوند زیرا هم فضای به‌مراتب کمتری اشغال می‌کنند و هم امکان جستجو و دسترسی به آن‌ها و انتقال آن‌ها بسیار بسیار ساده‌تر و سریع‌تر است. ما نیز از این قابلیت در ارائهٔ الگوریتم‌ها استفاده خواهیم کرد.

محدودیت‌ها:

۱. محدودیت سرعت: درست است که کامپیوترها امکان آن را دارند که عملیات را بسیار سریع انجام دهند و این امر به سرعت درحال افزایش است ولی در هر حال، اگر زمان محدودی را درنظر بگیریم، تعداد عملیاتی که درآن زمان قابل انجام است متناهی است. مثلاً اگر کامپیوتری قادر به انجام یک میلیون عملیات اساسی دریک ثانیه باشد، مطمئناً برای اجرای الگوریتمی که لازمهٔ آن انجام یک میلیارد محاسبه است به زمان ۱۰۰۰ ثانیه یا حدوداً ۱۷ دقیقه زمان نیاز دارد. حال اگر مقدار این عملیات به ۱۰۰ میلیارد محاسبه و یا یک تریلیون (۱۰۰۰ میلیارد) محاسبه برسد به‌ترتیب زمان ۱۷۰۰ دقیقه یا ۲۸ ساعت و ۱۷۰۰۰ دقیقه یا ۱۲ روز وقت نیاز دارد. به‌راحتی می‌توان دید که با افزایش مقدار عملیات، حتی ممکن است اجرای یک

الگوریتم توسط کامپیوتر به سال‌ها وقت نیاز داشته باشد.

این امر البته وجود خارجی نیز دارد. بسیاری از مسائل وجود دارند که برای آن‌ها الگوریتم ارائه شده است اما با یک حساب سرانگشتی روی تعداد محاسباتی که انجام می‌دهند و درنظرگرفتن سریع‌ترین کامپیوترهای امروزی، گاهی به صدها سال زمان نیاز دارند تا مسئله را حل کنند. درست است که راه‌حل وجود دارد، ولی انجام آن دراین مدت زمان عملی نیست. لذا عملاً این مسائل حتی با کامپیوتر قابل حل نیستند. از این دید، همواره مطلوب است که برای مسائل، الگوریتمی ارائه شود که تعداد عملیات مورد نیاز آن به صورتی باشد که امکان اجرای آن در زمان معقولی وجود داشته باشد. این امر در بسیاری از کاربردها خود را نشان می‌دهد. مثلاً فرض کنید فردی یک الگوریتم برای پیش‌بینی وضعیت هوای ۲۴ ساعت آینده دارد که دقیقاً درست پیش‌بینی می‌کند. تنها مشکل آن این است که اجرای این الگوریتم روی یک کامپیوتر مثلاً ۲ روز جهت رسیدن به جواب طول می‌کشد. مشخص است که این الگوریتم برای استفاده کارآیی ندارد زیرا نتایج پیش‌بینی آن ۲۴ ساعت بعد از زمان وقوع پدیده‌ها مشخص می‌شود که عملاً کاربردی ندارد.

۲. محدودیت حافظه: حافظه کامپیوتر فضای زیادی جهت نگهداری اطلاعات دراختیار قرار می‌دهد ولی علیرغم افزایش روزافزون این فضا، درنهایت فضای در اختیار محدود است. مشابه بحثی که در خصوص محدودیت سرعت داشتیم، عیناً می‌توان برای محدودیت حافظه نیز تکرار کرد و البته بسیاری از مسائل وجود دارند که جهت حل آن‌ها، به‌فضای حافظه‌ای به‌مراتب بیشتراز فضای حافظهٔ پیشرفته‌ترین کامپیوترهای امروزی نیاز دارد و به‌همین دلیل قابل حل توسط کامپیوتر نیستند.

۳. عدم توانایی استدلال و استنتاج: قابلیت استدلال و استنتاج یکی از مشخصه‌های بشر است که مختص خود اوست و از آن به ”هوش“ یاد می‌شود. کامپیوتر این قابلیت را ندارد به این معنی که امکان قرار دادن نتایج مختلف کنار هم و رسیدن به نتیجهٔ دیگری راکه ما آن‌را ثمرهٔ ”تفکر“ می‌نامیم ندارد. این محدودیت عملاً باعث می‌شود که ما فقط بتوانیم بر قدرت محاسباتی کامپیوتر برای اجرای الگوریتم‌های

خود تکیه کنیم.

لازم به ذکر است که اضافه‌کردن هوش به کامپیوتر که به ''هوش مصنوعی'' معروف است، سالیان سال جزء شاخه‌های کاری دانشمندان بوده است و همچنان نیز می‌باشد. عملاً عملکرد این شاخه به این صورت است که فرآیند ''هوش'' را به نوعی تبدیل به کاری محاسباتی کنند که کامپیوتر بتواند با آن محاسبات به یک نتیجه برسد. حتی ساده‌ترین کار در این زمینه نیز بسیار پیچیده است. به‌عنوان مثال فقط اضافه کردن قابلیت راه‌پیمایی به یک ربات فرآیند پیچیده‌ای را می‌طلبد که سال‌ها برای رسیدن به آن تلاش شده است.

۳.۲ ساختارهای شرطی

همانطور که قبلاً اشاره شد، یک کامپیوتر، علاوه بر انجام عملیات محاسباتی، امکان مقایسه داده‌ها و اعداد را نیز دارد. با این معنی که کاربر می‌تواند با استفاده از عملیات مقایسه، داده‌های عددی مختلف، اعم از ثابت‌های عددی یا مقادیر موجود در متغیرها را با یکدیگر مقایسه کند. این امر به حل بسیاری از مسائل پیچیده‌تر کمک می‌کند. در این بخش به استفاده از این قابلیت برای حل برخی مسائل می‌پردازیم.

مثال ۱.۳.۲. *الگوریتمی ارائه کنید که یک عدد صحیح را از ورودی خوانده و زوج یا فرد بودن عدد را در خروجی چاپ کند.*

حل: برای کامپیوتر، حتی تشخیص زوج یا فرد بودن عدد ممکن نیست و لذا شما نمی‌توانید از عبارتی مانند ''آیا عدد n زوج است؟'' در الگوریتم خود استفاده کنید. لذا برای تشخیص زوج و فرد بودن عدد، باقیمانده تقسیم عدد بر۲ را محاسبه کرده و اگر این باقیمانده برابربا صفر بود آنگاه می‌توان نتیجه گرفت عدد مزبور زوج است. با این توضیح، روش حل مسئله در الگوریتم ۶.۲ آمده است.

در این الگوریتم دو نکته وجود دارد. اولاً این که آیا کامپیوتر می‌تواند جزء صحیح یک عدد را محاسبه کند (دستور خط ۲ را ببینید). همچنان که خواهیم دید مشکلی در این خصوص ندارد و بعداً روشی برای محاسبهٔ جزء صحیح یک عدد ارائه خواهیم کرد. نکتهٔ دوم، استفاده از شرط برای تصمیم‌گیری روند اجرای الگوریتم است. این تصمیم‌گیری بر مبنای محتوای یک متغیر صورت می‌گیرد. در آینده به کرات از چنین تصمیم‌گیری‌هایی در الگوریتم‌ها استفاده خواهیم کرد. □

الگوریتم ۶.۲: الگوریتم تشخیص زوج یا فرد بودن عدد

```
۱ شروع
۲ n را بگیر
۳ قرار بده r ← n − [n/۲] * ۲
۴ اگر r = ۰ آنگاه
۵ | "Even" را چاپ کن
۶ درغیراینصورت
۷ | "Odd" را چاپ کن
۸ پایان اگر
۹ پایان
```

مثال ۲.۳.۲. *الگوریتمی ارائه کنید که یک عدد را از ورودی خوانده و قدرمطلق آن را در خروجی چاپ کند.*

حل: مشابه قبل، محاسبهٔ قدرمطلق یک عدد جزء عملیات اساسی قابل انجام توسط کامپیوتر نیست ولی با ترفندی ساده می‌توان آن را حل کرد. الگوریتم ۷.۲ این کار را با استفاده از یک شرط انجام می‌دهد.

بررسی درستی الگوریتم فوق خیلی مشکل نیست. اگر عدد وارد شده یک عدد مثبت باشد، آنگاه شرط $x < ۰$ درست نیست و لذا قسمت در غیر اینصورت اجرا شده و $y = x$

قرار می‌گیرد. درصورت برقرار بودن شرط $x < ۰$ آنگاه مقدار $-۱ \times x$ در y قرار می‌گیرد که در هرحال مقدار چاپ شده یعنی y همان قدرمطلق عدد x است. آیا الگوریتم برای ورودی ۰ درست کار می‌کند؟ □

الگوریتم ۷.۲: الگوریتم محاسبهٔ قدرمطلق یک عدد

۱ **شروع**
۲ x **را بگیر**
۳ **اگر** $x < ۰$ **آنگاه**
۴ | قرار بده $y \leftarrow -۱ \times x$
۵ **درغیراینصورت**
۶ | قرار بده $y \leftarrow x$
۷ **پایان اگر**
۸ y **را چاپ کن**
۹ **پایان**

مثال ۳.۳.۲. *الگوریتمی ارائه کنید که سه عدد را از ورودی خوانده و بزرگترین عدد از بین سه عدد وارد شده را محاسبه و در خروجی چاپ کند.*

حل: باتوجه به امکان مقایسهٔ اعداد، الگوریتم ۸.۲ این کار را انجام خواهد داد. □

اگر بخواهیم مسئله فوق را به محاسبهٔ ماکزیمم از بین مقدار بیشتری عدد، تعمیم دهیم، مشخص است که تعداد دستورات الگوریتم به نسبت تعداد اعداد افزایش پیدا می‌کند. در آینده تکنیکی خواهیم دید که با الگوریتمی مشخص می‌توان ماکزیمم هر تعداد عدد را محاسبه کرد.

مثال ۴.۳.۲. *الگوریتمی ارائه کنید که سه عدد را از ورودی خوانده و مشخص کند آیا مثلث قائم‌الزاویه‌ای با طول اضلاع وارد شده وجود دارد یاخیر. اگر چنین مثلثی وجود دارد در خروجی عبارت YES و در غیر این صورت عبارت NO را چاپ کند.*

الگوریتم ۸.۲: الگوریتم محاسبهٔ ماکزیمم سه عدد

۱ **شروع**
۲ A و B و C **را بگیر**
۳ $MAX \leftarrow A$
۴ **اگر** $B > MAX$ **آنگاه**
۵ | قرار بده $MAX \leftarrow B$
۶ **پایان اگر**
۷ **اگر** $C > MAX$ **آنگاه**
۸ | قرار بده $MAX \leftarrow C$
۹ **پایان اگر**
۱۰ MAX **را چاپ کن**
۱۱ **پایان**

حل: برای حل این مسئله از قضیهٔ فیثاغورث استفاده می‌کنیم. درصورتی که سه عدد در شرط فیثاغورث صدق کنند آنگاه مثلث قائم‌الزاویه‌ای وجود دارد که طول اضلاع آن برابر با مقادیر داده شده باشد، تنها مشکل این است که کدام عدد را طول وتر درنظر بگیریم. برای سادگی ما سه حالت درنظرمی‌گیریم که هرکدام با درنظر گرفتن یکی از طول‌ها به‌عنوان طول وتر است. با این توصیف، الگوریتم حل مسئله در الگوریتم ۹.۲ آمده است.

دقت کنید که در الگوریتم ۹.۲ از متغیر کمکی $flag$ استفاده کردیم که وضعیت سه شرط را با استفاده از آن تعیین کنیم. هرکدام از این شرط‌ها برقرار باشد آنگاه به $flag$ مقدار ۱ داده می‌شود درحالی که مقدار اولیه اختصاص داده شده به آن برابر صفر است. با این ترتیب با بررسی مقدار $flag$ در پایان، می‌توانیم تشخیص دهیم که آیا یکی از شروط برقرار بوده است یا خیر.

البته درخصوص این الگوریتم می‌توانستیم از خاصیت دیگر مثلث قائم‌الزاویه نیز استفاده کنیم که بیان می‌کند که طول وتر مثلث قائم‌الزاویه، از طول دوضلع دیگر بزرگ‌تر است. با

این ترتیب کافی بود ماکزیمم از بین سه عدد پیدا شود و آن دقیقاً طول وتر است. سپس با استفاده از قضیه فیثاغورث و مشخص بودن وتر، فقط یک شرط بررسی شود. نوشتن این الگوریتم به خواننده واگذار می‌شود. □

الگوریتم ۹.۲: الگوریتم تشخیص وجود مثلث قائم‌الزاویه با طول ضلع‌های داده شده

۱ **شروع**
۲ A و B و C **را بگیر**
۳ قرار بده $flag \leftarrow ۰$
۴ **اگر** $A^۲ = B^۲ + C^۲$ **آنگاه**
۵ | قرار بده $flag \leftarrow ۱$
۶ **پایان اگر**
۷ **اگر** $B^۲ = A^۲ + C^۲$ **آنگاه**
۸ | قرار بده $flag \leftarrow ۱$
۹ **پایان اگر**
۱۰ **اگر** $C^۲ = A^۲ + C^۲$ **آنگاه**
۱۱ | قرار بده $flag \leftarrow ۱$
۱۲ **پایان اگر**
۱۳ **اگر** $flag = ۱$ **آنگاه**
۱۴ | YES **را چاپ کن**
۱۵ **درغیراینصورت**
۱۶ | NO **را چاپ کن**
۱۷ **پایان اگر**
۱۸ **پایان**

مرتب‌سازی داده‌ها یکی از مباحث اصلی در بحث الگوریتم‌ها است. در مثال زیر به مثال ساده‌ای از این مبحث می‌پردازیم.

مثال ۵.۳.۲. *الگوریتمی ارائه کنید که سه عدد را از ورودی خوانده و آنها را به ترتیب صعودی در خروجی چاپ کند.*

حل: جهت مرتب کردن سه عدد می‌توانیم به این‌صورت عمل کنیم. سه متغیر x و y و z درنظر می‌گیریم. از بین اعداد وارد شده کوچک‌ترین عدد را در x و دومین کوچک‌ترین عدد را در y و در نهایت بزرگ‌ترین عدد را در z قرار می‌دهیم. با چاپ x و y و z به همین ترتیب، اعداد به‌صورت صعودی مرتب می‌شوند. الگوریتم این مسئله در الگوریتم ۱۰.۲ آمده است. □

الگوریتم ۱۰.۲: الگوریتم مرتب‌سازی سه عدد ورودی

```
۱  شروع
۲  A و B و C را بگیر
۳  اگر A ≥ B آنگاه
۴  |  قرار بده x ← B و y ← A
۵  درغیراینصورت
۶  |  قرار بده x ← A و y ← B
۷  پایان اگر
۸  اگر x ≥ C آنگاه
۹  |  قرار بده z ← y و y ← x و x ← C
۱۰ درغیراینصورت
۱۱ |  اگر y ≥ C آنگاه
۱۲ |  |  قرار بده z ← y و y ← C
۱۳ |  درغیراینصورت
۱۴ |  |  قرار بده z ← C
۱۵ |  پایان اگر
۱۶ پایان اگر
۱۷ اول x و سپس y و سپس z را چاپ کن
۱۸ پایان
```

۴.۲ حلقه‌های تکرار

دربرخی مسائل لازم است تا برای انجام یک کار، چندین دستورالعمل به‌صورت تکراری انجام شود. مثلاً فرض کنید بخواهیم ۱۰۰ عدد را از کاربر دریافت کرده و مجموع آنها را محاسبه و چاپ کنیم. برای این منظور اگر بخواهیم به‌صورت معمول برای هر ورودی و جمع‌کردن یک دستور بنویسیم به الگوریتمی طولانی منجر خواهد شد که اکثر دستورات آن مشابه است. منظور از حلقه‌ی تکرار آن است که دستورات مشترک را مشخص کنیم و سپس با استفاده از دستورات شرطی و متغیرها، آن دستورات مشترک را به‌تعداد مورد نیاز تکرار کنیم.

برای این منظور، یک شرط مناسب در نظر گرفته می‌شود که درصورتی که آن شرط درست باشد دستورات مورد نظر انجام می‌شود و در صورت برقرار نبودن از حلقه خارج شده و سایر دستورات الگوریتم اجرا می‌شود.

برای مثال ذکرشده، یعنی محاسبه‌ی مجموع ۱۰۰ عدد وارد شده توسط کاربر، الگوریتم زیر رادرنظر بگیرید. در این الگوریتم باید به کرات عددی از ورودی گرفته شود و به مجموع اعداد قبلی اضافه شود. برای شمردن تعداد دفعات تکرار از یک متغیر استفاده می‌کنیم که مقدار اولیهٔ آن یک است و با هر بار تکرار دستورات، یکی به آن اضافه می‌شود. ما چنین متغیری را اصطلاحاً ''شمارنده'' می‌نامیم. از یک متغیر نیز که با مقدار صفر مقداردهی اولیه شده است برای اضافه کردن هرعدد وارد شده استفاده می‌کنیم. درنهایت این متغیر، حاوی مجموع اعداد وارد شده خواهد بود.

مثال ۱.۴.۲. *الگوریتمی ارائه کنید که ۱۰۰ عدد را بگیرد و مجموع آنها را محاسبه و چاپ کند.*

حل: با توجه به توضیحات بیان شده در بالا، الگوریتم برای حل مسئله در الگوریتم ۱۱.۲ آمده است. □

الگوریتم ۱۱.۲: الگوریتم دریافت ۱۰۰ عدد و محاسبهٔ مجموع آنها

۱ **شروع**
۲ $s \leftarrow ۰$ و $i \leftarrow ۱$
۳ **اگر** $i \leq ۱۰۰$ **آنگاه**
۴ | A **را بگیر**
۵ | $s \leftarrow s + A$
۶ | $i \leftarrow i + ۱$
۷ | به خط ۳ برو
۸ **پایان اگر**
۹ s **را چاپ کن**
۱۰ **پایان**

مثال ۲.۴.۲. *الگوریتمی ارائه کنید که یک عدد طبیعی n را بگیرد و مشخص کند آیا اول است یا خیر.*

حل: برای تشخیص اول بودن یک عدد مثل n کافی است بخشپذیری آن بر هر عدد صحیح i بین ۲ تا $n/۲$ بررسی شود. برای این بررسی از یک حلقهٔ تکرار استفاده می‌کنیم. الگوریتم ۱۲.۲ این روند را نشان می‌دهد.

الگوریتم ۱۳.۲ همین کار را با اندکی تغییر انجام می‌دهد. در اینجا تاکید می‌شود که مشابه مسائل ریاضی که راه حل منحصر به فرد ندارند و ممکن است برای یک مسئله راه حل‌های متفاوت وجود داشته باشد، برای حل یک مسئله توسط کامپیوتر نیز ممکن است الگوریتم‌های متعددی وجود داشته باشد. البته در مباحث پیشرفته‌تر، معیارهایی برای مقایسه الگوریتم‌ها از نظر کارآیی آورده شده است که در اینجا به آن نمی‌پردازیم. □

دقت کنید که در الگوریتم ۱۳.۲، شرط خط ۴ از ترکیب عطفی دو شرط استفاده شده است. درصورتی که هر کدام از این دو شرط غلط باشد، ترکیب عطفی آن‌ها نیز غلط است

الگوریتم ۱۲.۲: الگوریتم تشخیص اول بودن یک عدد

```
۱ شروع
۲ n را بگیر
۳ i ← ۲
۴ اگر i ≤ n/۲ آنگاه
۵     اگر باقیمانده تقسیم n بر i برابر صفر است آنگاه
۶         "Not Prime" را چاپ کن
۷         پایان
۸     درغیراینصورت
۹         i ← i + ۱
۱۰        به مرحلهٔ ۴ برو
۱۱    پایان اگر
۱۲ پایان اگر
۱۳ "Prime" را چاپ کن
۱۴ پایان
```

و حلقه پایان می‌یابد. درحقیقت خط ۸ بررسی می‌کند که کدام شرط دلیل غلط بودن ترکیب عطفی است. اگر $i \geq n/۲$ آنگاه شرط اول، دلیل خارج شدن از حلقه است که به این معنی است که در تمام اعداد بررسی شده، n بر آن اعداد بخشپذیر نبوده است و این به‌معنی اول بودن آن است. در غیر این صورت شرط بخشپذیری برقرار بوده است که به معنی اول نبودن عدد n است.

نکتهٔ دیگر استفاده کردن از جمله‌ی "n بر i بخشپذیر است" در خط ۴ الگوریتم است. این دستور جزء عملیات اساسی قابل انجام توسط کامپیوتر نیست. اما به سادگی می‌توانیم آن را با محاسبهٔ باقیماندهٔ تقسیم n بر i محاسبه کنیم. قبلاً باقیمانده تقسیم n بر ۲ را دیده‌ایم و به‌شکل مشابه $r = n - [n/i] \times i$ باقیماندهٔ تقسیم n بر i را به r می‌دهد. با بررسی

الگوریتم ۱۳.۲: الگوریتم دوم تشخیص اول بودن یک عدد

۱ **شروع**
۲ n **را بگیر**
۳ $i \leftarrow ۲$
۴ **اگر** $i < \frac{n}{۲}$ *و* n *بر* i *بخشپذیر نیست* **آنگاه**
۵ $\quad i \leftarrow i + ۱$
۶ $\quad$ به مرحلهٔ ۴ برو
۷ **پایان اگر**
۸ **اگر** $i \geq \frac{n}{۲}$ **آنگاه**
۹ $\quad$ "Prime" **را چاپ کن**
۱۰ **درغیراینصورت**
۱۱ $\quad$ "Not Prime" **را چاپ کن**
۱۲ **پایان اگر**
۱۳ **پایان**

صفر بودن این مقدار به‌راحتی می‌توانیم بخشپذیری n بر i را بررسی کنیم. با توجه به این موضوع، ما معمولاً از نماد ٪ برای تعیین باقیمانده تقسیم یک عدد صحیح بر عدد صحیح دیگر استفاده می‌کنیم. به عنوان مثال در این الگوریتم می‌توانیم به جای شرط "باقیمانده تقسیم n بر i برابر صفر است" از "$n \,٪\, i = ۰$" استفاده کنیم.

مثال ۳.۴.۲. *الگوریتمی ارائه کنید که یک عدد طبیعی n را بگیرد و $n!$ را محاسبه و چاپ کند.*

حل: برای حل این مسئله باید حاصل‌ضرب اعداد یک تا n را محاسبه کنیم. پس از روشی مشابه مثال قبل استفاده می‌کنیم. برای این منظور از یک شمارنده‌ی i استفاده می‌کنیم که مقدار اولیه‌ی آن ۱ است و دریک حلقه‌ی تکرار، هربار یکی به آن اضافه می‌شود و هربار

عدد i دریک متغیر که برای نگهداری $n!$ درنظر گرفته شده است و مقدار اولیه‌ی آن ۱ است ضرب می‌کنیم. این روند تا وقتی‌که i کمتر یا مساوی n است تکرار می‌شود و به محض این که i بزرگ‌تر از n شود تکرار، قطع می‌شود. الگوریتم ۱۴.۲ را ببینید. □

الگوریتم ۱۴.۲: الگوریتم محاسبهٔ فاکتوریل یک عدد

۱ **شروع**
۲ n **را بگیر**
۳ $i \leftarrow ۱$ و $FACT \leftarrow ۱$
۴ **اگر** $i \leq n$ **آنگاه**
۵ | $FACT \leftarrow FACT \times i$
۶ | $i \leftarrow i + ۱$
۷ | به خط ۴ برو
۸ **پایان اگر**
۹ $FACT$ **را چاپ کن**
۱۰ **پایان**

۵.۲ فلوچارت (نمودار گردشی)

یک فلوچارت (نمودار گردشی) نوعی نمودار است که روند یک الگوریتم، جریان یک کار یا فرآیند را نمایش می‌دهد. این نمایش نموداری، راه‌حل یک مسئله را با استفاده از اشکال هندسی نمایش می‌دهد به صورتی که دنبال‌کردن و بررسی و فهم آن ساده‌تر است. استفاده از فلوچارت به دهه‌ی ۱۹۲۰ برمی‌گردد که برای کارهای مختلف درمهندسی از آن استفاده شده است. امروزه فلوچارت‌ها نه‌تنها در بیان نموداری الگوریتم‌ها، بلکه در بسیاری از شاخه‌های دیگر استفاده می‌شوند و روز به روز به دستورالعمل‌های فلوچارتی برای انجام کارهای اداری، استفاده از یک دستگاه جدید و مانند آن اضافه می‌شود.

فلوچارت‌ها یک روش استاندارد و همگانی برای توصیف الگوریتم‌های کامپیوتری هستند

و هنوز نیز برای این منظور استفاده می‌شوند. روش‌های مدرن نظیر نمودار فعالیت UML یا نمودار $Drakon$ به نوعی توسیع فلوچارت‌ها هستند.

در این بخش به منظور نمایش نموداری الگوریتم‌ها، به مبحث فلوچارت می‌پردازیم. تأکید می‌شود که فلوچارت صرفاً نمایش نموداری الگوریتم است که فهم روند اجرای الگوریتم و نمایش آن را ساده‌تر می‌کند و صرفاً بیان روش حل مسئله، به شکلی دیگر است و لذا در به‌دست آوردن روش حل مسئله عملاً نقشی ندارد. در آینده می‌توانید برای بیان روش حل یک مسئله، هم از روش الگوریتم و هم از فلوچارت استفاده کنید اما همچنان که خواهیم دید، استفاده از فلوچارت به شفافیت بیان روش حل و فهم بهتر آن کمک می‌کند و همچنین روند تبدیل روش به برنامه کامپیوتری را که از مباحث بعدی این کتاب است، تسهیل می‌کند.

با توجه به این که در الگوریتم‌ها، ما از انواع محدودی از دستورات استفاده می‌کنیم، درفلوچارت برای هرکدام از این نوع دستورات یک شکل هندسی درنظر می‌گیریم. با این ترتیب، دستورات به جای نوشته شدن، در یک شکل هندسی قرار می‌گیرند. برای رعایت ترتیب اجرای دستورات نیز به‌جای استفاده از شماره‌ها، از پیکان‌هایی استفاده می‌کنیم که ترتیب اجرای دستورات را نشان می‌دهد. به‌عنوان مثال اگر در الگوریتمی، دستور B باید بلافاصله بعد از دستور A اجرا شود، از شکل هندسی مربوط به دستور A یک پیکان به شکل هندسی مربوط به دستور B رسم می‌شود.

بلوک‌های ساختمانی فلوچارت

همانطور که قبلاً بیان شد، هر نوع از دستورات در الگوریتم‌ها، با یک شکل هندسی در فلوچارت بیان می‌شود که از آنها به عنوان بلوک‌های ساختمانی فلوچارت یاد می‌کنیم. دراین قسمت، این بلوک‌های ساختمانی معرفی می‌شوند و سپس روند تبدیل الگوریتم به فلوچارت با استفاده از چند مثال تشریح می‌شود. دوباره تأکید می‌کنیم که فلوچارت صرفاً یک الگوریتم را با یک نمودار نشان می‌دهد و عملاً تغییر شکل بیان یک راه‌حل است.

شکل‌های هندسی مربوط به دستورات مختلف به‌صورت زیر است:

- شروع و پایان: برای شروع و پایان از بیضی استفاده می‌شود که در داخل آن کلمه‌ی ''شروع'' یا ''پایان'' نوشته شده است.

- عبارت محاسباتی یا انتساب: برای عبارت‌های محاسباتی یا انتساب (یعنی قرار دادن یک مقدار داخل یک متغیر)، از مستطیل استفاده می‌شود. عبارت محاسباتی مورد نظر در داخل مستطیل قرار می‌گیرد. به‌عنوان مثال دستور $i \leftarrow i+1$ در داخل یک مستطیل قرار داده می‌شود.

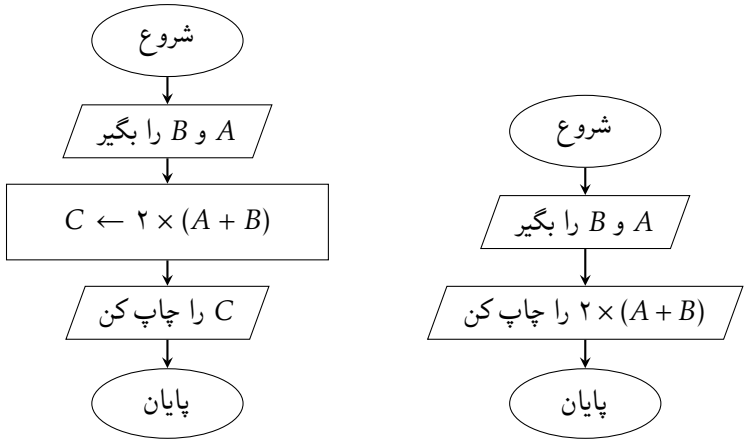
- ورودی-خروجی: برای ورودی و خروجی از متوازی‌الاضلاع استفاده می‌شود. مثلاً برای دستور ''مقدار n را بگیر''، همین عبارت داخل یک متوازی‌الاضلاع قرار می‌گیرد. به‌طور مشابه برای ''x را چاپ کن''.

- شرط یا تصمیم‌گیری: در دستورات شرطی، از یک لوزی استفاده می‌کنیم. معمولاً شرط مورد بررسی به نتیجه‌ی بله/خیر منجر می‌شود و برمبنای آن دو خروجی برای لوزی درنظر گرفته می‌شود. به‌طورمشابه ممکن است شرط مورد بررسی برای درست یا نادرست باشد که به شکل مشابه استفاده می‌شود. درمواردی که شرط مورداستفاده پیچیده‌تر است توصیه می‌شود ابتدا به صورت ترکیب چند شرط ساده‌تر نوشته شود و سپس به نمودار تبدیل شود.

- ترتیب انجام دستورات: همانطور که قبلاً نیز بیان شد، از پیکان برای مشخص‌کردن ترتیب اجرای دستورات استفاده می‌کنیم.

- درصورتی‌که به‌دلیل محدودیت فضا، امکان رسم قسمتی از فلوچارت وجود نداشته باشد، با استفاده از یک دایره که داخل آن عبارت یا کاراکتری قرار گرفته است و اسم همان شکل درمحلی دیگر، می‌توانیم ارائه فلوچارت را درمحلی دیگر بیاوریم.

نمادهای دیگری نیز برای فلوچارت وجود دارد که در صورت نیاز، به معرف آنها پرداخته خواهد شد. با استفاده از نمادهای معرفی شده در بالا می‌توانیم الگوریتم‌های خود را به راحتی به فلوچارت تبدیل کنیم.



شکل ۲.۲: فلوچارت محاسبهٔ محیط مستطیل (آ) به طور مستقیم (ب) با استفاده از متغیر کمکی.

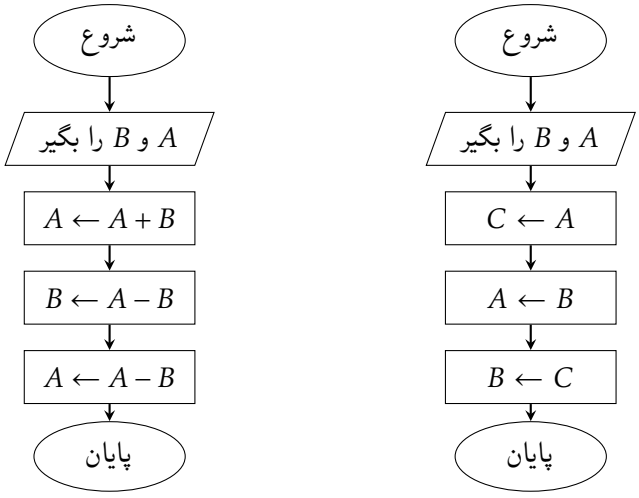
مثال ۱.۵.۲. *فلوچارتی رسم کنید که طول و عرض یک مستطیل را از ورودی دریافت کند و محیط مستطیل را محاسبه و چاپ کند.*

حل: با توجه به الگوریتم ۲.۲، فلوچارت در شکل ۲.۲ (آ) آمده است.

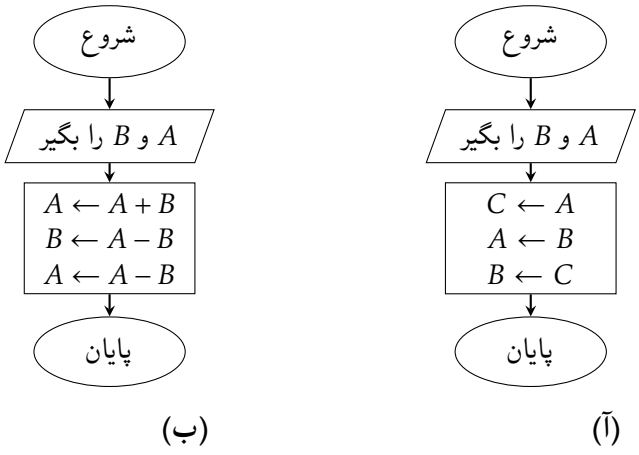
البته با توجه به این که در متوازی‌الاضلاع صرفاً باید عمل ورودی یا خروجی ذکر شود، بهتراست خط ۳ الگوریتم ۲.۲ با دستورات ذکر شده در توضیحات بعد از مثال مربوطه جایگزین شده و سپس برای الگوریتم حاصل، فلوچارت رسم شود. فلوچارت در این حالت در شکل ۲.۲ آمده است. □

مثال ۲.۵.۲. *فلوچارتی رسم کنید که دو مقدار را گرفته و در دو متغیر ذخیره کند و سپس محتوای دو متغیر را (با استفاده و بدون استفاده از متغیر کمکی) جابجا کند.*

حل: با توجه به الگوریتم‌های ۴.۲ و ۵.۲، فلوچارت متناظر با این الگوریتم‌ها در شکل ۳.۲ آمده است.



شکل ۳.۲: فلوچارت جابجایی محتوای دو متغیر (آ) با استفاده از متغیر کمکی (ب) بدون استفاده از متغیر کمکی.



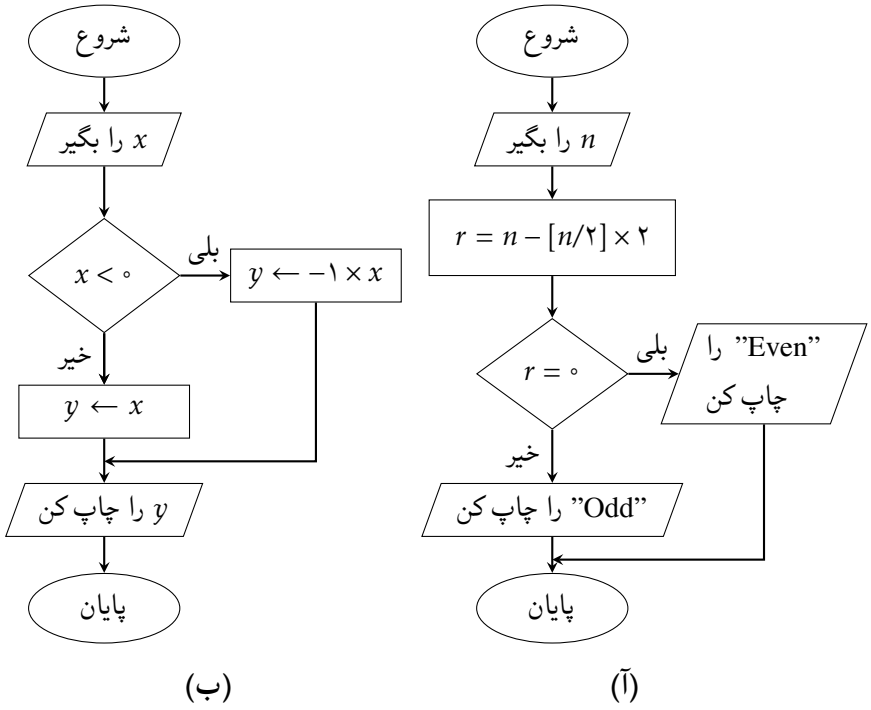
شکل ۴.۲: فلوچارت جابجایی محتوای دو متغیر (آ) با استفاده از متغیر کمکی (ب) بدون استفاده از متغیر کمکی.

در موارد مشابه فوق که چند دستور محاسباتی پشت سرهم قرار می‌گیرند، می‌توانیم جهت صرفه‌جویی در فضا، همهٔ آن‌ها را در یک مستطیل قرار دهیم. البته دقت شود که

در این صورت ترتیب اجرای دستورات باید از بالا به پایین باشد زیرا تغییر ترتیب اجرای دستورات ممکن است به نتایج نادرستی منجر شود. فلوچارت‌های فوق را با این توضیحات در شکل ۴.۲ بازنویسی کرده‌ایم. □

مثال ۳.۵.۲. *فلوچارتی رسم کنید که یک عدد صحیح را از ورودی خوانده و زوج یا فرد بودن عدد را در خروجی چاپ کند.*

حل: با توجه به الگوریتم ۶.۲، فلوچارت متناظر با این الگوریتم در شکل ۵.۲ (آ) آمده است. □



شکل ۵.۲: فلوچارت (آ) تشخیص زوج یا فرد بودن عدد. (ب) چاپ قدرمطلق عدد.

مثال ۴.۵.۲. *فلوچارتی رسم کنید که یک عدد را از ورودی خوانده و قدرمطلق آن را در خروجی چاپ کند.*

حل: با توجه به الگوریتم ۷.۲، فلوچارت متناظر با این الگوریتم در شکل ۵.۲ (ب) آمده است. □

مثال ۵.۵.۲. *فلوچارتی رسم کنید که سه عدد را از ورودی خوانده و ماکزیمم عدد از بین سه عدد وارد شده را محاسبه و در خروجی چاپ کند.*

حل: با توجه به الگوریتم ۸.۲، فلوچارت متناظر با این الگوریتم در شکل ۶.۲ (آ) آمده است. □

مثال ۶.۵.۲. *فلوچارتی رسم کنید که سه عدد را از ورودی خوانده و مشخص کند آیا مثلث قائم‌الزاویه‌ای با طول اضلاع وارد شده وجود دارد یاخیر. اگر چنین مثلثی وجود دارد در خروجی عبارت YES و در غیر این صورت عبارت NO را چاپ کند.*

حل: با توجه به الگوریتم ۹.۲، فلوچارت متناظر با این الگوریتم در شکل ۶.۲ (ب) آمده است. □

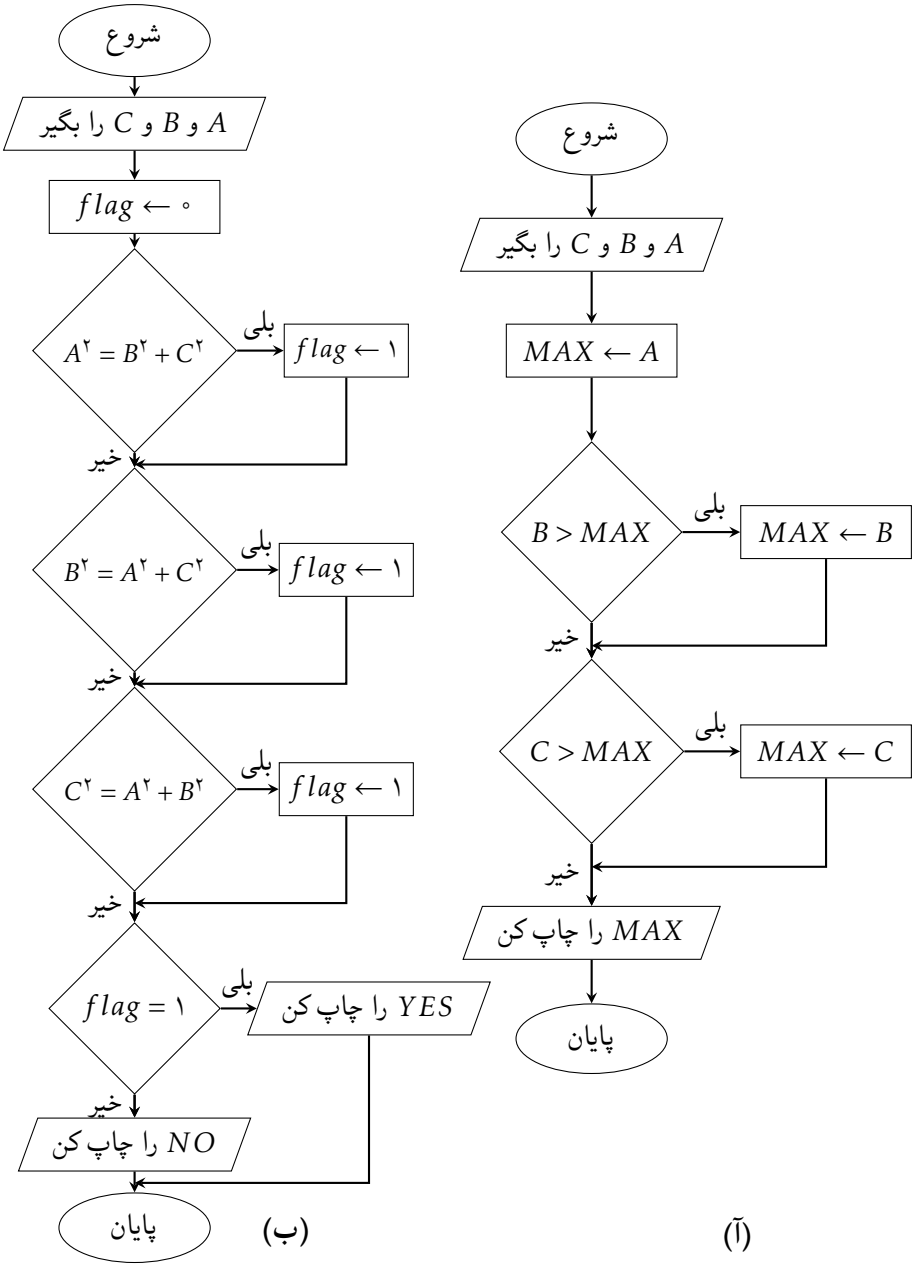
مثال ۷.۵.۲. *فلوچارتی ارائه کنید که سه عدد را از ورودی خوانده و آنها را به ترتیب صعودی در خروجی چاپ کند.*

حل: با توجه به الگوریتم ۱۰.۲، فلوچارت متناظر با این الگوریتم در شکل ۷.۲ (آ) آمده است. □

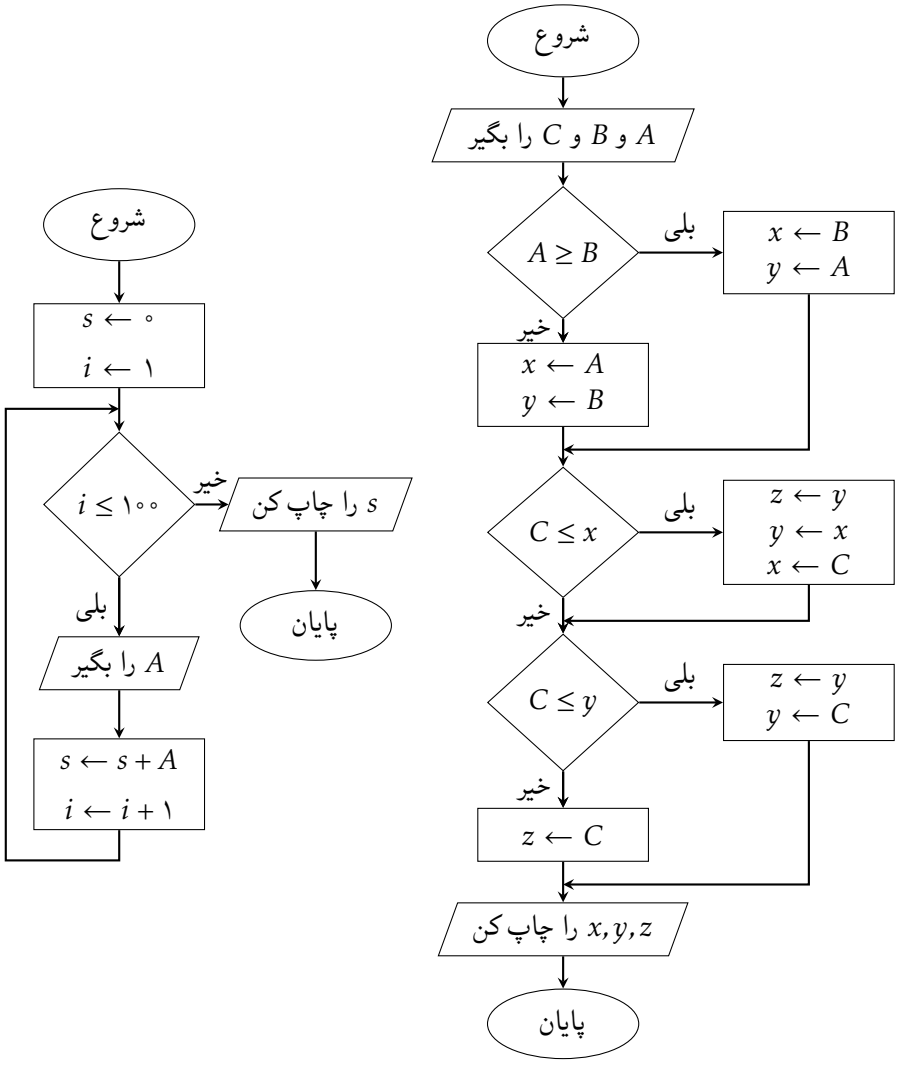
مثال ۸.۵.۲. *فلوچارتی رسم کنید که ۱۰۰ عدد را بگیرد و مجموع آنها را محاسبه و چاپ کند.*

حل: با توجه به الگوریتم ۱۱.۲، فلوچارت متناظر با این الگوریتم در شکل ۷.۲ (ب) آمده است. □

در مثال فوق، کاملاً حلقهٔ تکرار بودن قسمتی از الگوریتم مشخص است. در فلوچارت‌ها، حلقه‌های تکرار معمولاً با یک دستور شرطی شروع می‌شوند و پس از آن تعدادی دستور وجود دارد با پیکانی به قبل از دستور شرطی اولیه برمی‌گردد. تشخیص حلقه‌های تکرار در الگوریتم و فلوچارت‌ها مهم است زیرا در آینده برای تبدیل آنها به برنامه، از دستورات مشخصی برای حلقه‌ها استفاده خواهیم کرد. هرچند، بدون دانستن حلقه نیز، تبدیل آنها به برنامه بدون استفاده از دستورات در نظر گرفته شده برای حلقه‌ها در زبان برنامه‌نویسی ممکن است اما استفاده از دستورات حلقه، کار برنامه‌نویسی و فهم برنامه را ساده‌تر می‌کند.



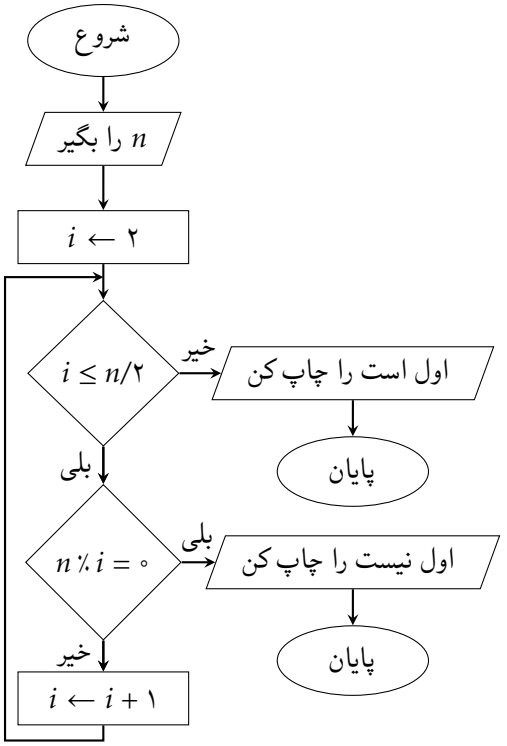
شکل ۶.۲: فلوچارت (آ) محاسبهٔ ماکزیمم سه عدد. (ب) تشخیص وجود مثلث قائم‌الزاویه با طول اضلاع داده شده.



شکل ۷.۲: فلوچارت (آ) مرتب کردن سه عدد. (ب) محاسبهٔ مجموع ۱۰۰ عدد.

مثال ۹.۵.۲. *فلوچارتی رسم کنید که یک عدد طبیعی n را بگیرد و مشخص کند آیا اول است یا خیر.*

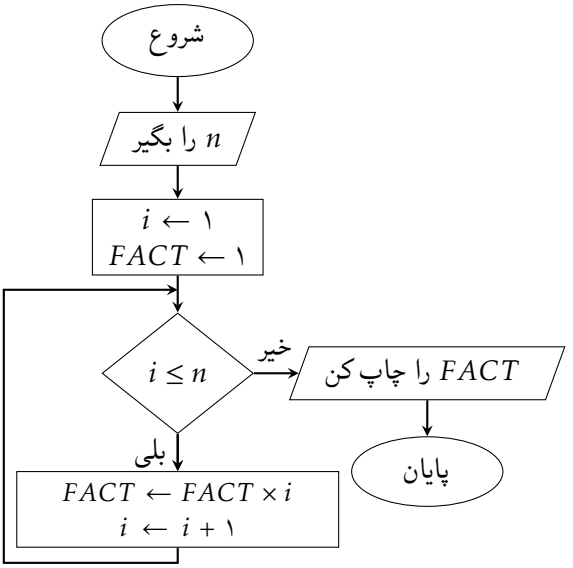
حل: با توجه به الگوریتم ۱۲.۲، فلوچارت متناظر با این الگوریتم‌ها، در شکل ۸.۲ آمده است. رسم فلوچارت برحسب الگوریتم ۱۳.۲ به عنوان تمرین به خواننده واگذار می‌شود. □



شکل ۸.۲: فلوچارت بررسی اول بودن عدد وارد شده.

مثال ۱۰.۵.۲. *فلوچارتی رسم کنید که یک عدد طبیعی n را بگیرد و $!n$ را محاسبه و چاپ کند.*

حل: با توجه به الگوریتم ۱۴.۲، فلوچارت متناظر با این الگوریتم، در شکل ۹.۲ آمده است. □



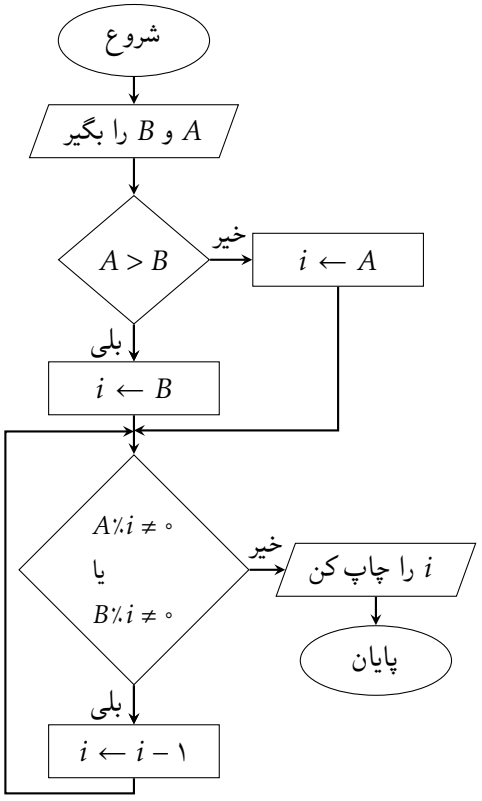
شکل ۹.۲: فلوچارت محاسبهٔ فاکتوریل عدد وارد شده.

مثال ۱۱.۵.۲. *فلوچارتی رسم کنید که دو عدد طبیعی را به عنوان ورودی بگیرد و بزرگ‌ترین مقسوم علیه (ب.م.م.) آنها را محاسبه و چاپ کند.*

حل: می‌دانیم ب.م.م. دو عدد A و B بزرگ‌ترین عدد صحیحی است که A و B هر دو بر آن بخشپذیر باشند. بدیهی است که ب.م.م دو عدد از هر دو عدد کوچکتر یا مساوی است. لذا برای حل مسئله، از کوچکترین عدد از بین A و B شروع کرده و بررسی می‌کنیم آیا هر دو عدد بر آن بخشپذیر است یا خیر. اگر بخشپذیر باشد، این عدد همان ب.م.م. است. در غیر این صورت از این عدد یکی کم کرده و دوباره بخشپذیری را بررسی می‌کنیم. به اولین عددی که هر دوی A و B بر آن بخشپذیر باشد، همان ب.م.م. است. بر این مبنا، الگوریتم و فلوچارت حل مسئله در الگوریتم ۱۵.۲ و شکل ۱۰.۲ آمده است. توجه کنید که در هر بار تکرار، یکی از عددی که برای ب.م.م. بودن بررسی می‌کنیم کم می‌شود. این روند حتما پایان پذیر است زیرا در نهایت در جایی عدد مورد بررسی به یک می‌رسد که مطمئناً هر دو عدد بر آن بخشپذیر است. این روند وقتی اتفاق می‌افتد که دو عدد نسبت به هم اول باشند.

الگوریتم ۱۵.۲: الگوریتم محاسبهٔ ب.م.م. دو عدد صحیح

۱ **شروع**
۲ A و B **را بگیر**
۳ **اگر** $A > B$ **آنگاه**
۴ | $i \leftarrow B$
۵ **درغیراینصورت**
۶ | $i \leftarrow A$
۷ **پایان اگر**
۸ **اگر** $A \,\%\, i \neq ۰$ یا $B \,\%\, i \neq ۰$ **آنگاه**
۹ | $i \leftarrow i - ۱$
۱۰ | برو به خط ۸
۱۱ **پایان اگر**
۱۲ i **را چاپ کن**
۱۳ **پایان**



شکل ۱۰.۲: فلوچارت محاسبهٔ ب.م.م. دو عدد صحیح وارد شده.

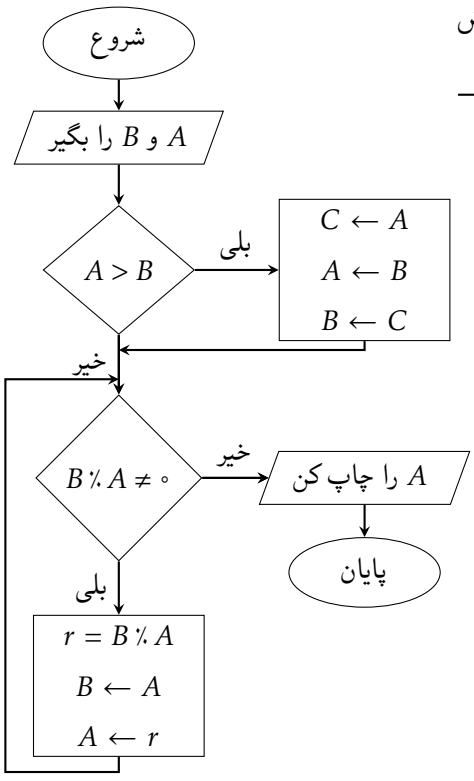
آیا الگوریتم ۱۵.۲ اگر اعداد وارد شده منفی باشد درست جواب می‌دهد؟

روش دیگری برای محاسبهٔ ب.م.م. وجود دارد که به روش نردبانی معروف است. در این روش برای محاسبهٔ ب.م.م. دو عدد A و B (با فرض این که $A < B$)، ابتدا بخشپذیری B بر A بررسی می‌شود. اگر B بر A بخشپذیر باشد، آنگاه A همان ب.م.م. دو عدد است. در غیر این صورت اعداد q و r ($۰ < r < A$) وجود دارند به طوری که $B = qA + r$. این رابطه نشان می‌دهد که r نیز باید بر ب.م.م. A و B بخشپذیر باشد. از طرفی B نیز باید بر ب.م.م. r و A بخشپذیر باشد. پس ب.م.م. r و A، همان ب.م.م. A و B است. با

استفاده از این روش، برای محاسبهٔ ب.م.م. دو عدد، ابتدا عدد بزرگ‌تر را بر عدد کوچک‌تر تقسیم می‌کنیم. اگر باقیماندهٔ تقسیم برابر با صفر باشد، مقسوم علیه برابر با ب.م.م. دو عدد است. در غیر این صورت مقسوم علیه را به عنوان عدد اول جدید و باقیمانده را به عنوان عدد دوم جدید در نظر گرفته و دوباره بخشپذیری عدد اول بر عدد دوم بررسی می‌شود. این روند تا زمانی تکرار می‌شود که باقیماندهٔ تقسیم برابر با صفر شود که در این حالت، مقسوم علیه آخرین تقسیم برابر با ب.م.م. است. الگوریتم ۱۶.۲ و فلوچارت شکل ۱۱.۲ بر این مبنا، ب.م.م. دو عدد صحیح را محاسبه می‌کند. □

الگوریتم ۱۶.۲: الگوریتم محاسبهٔ ب.م.م. دو عدد صحیح به روش نردبانی

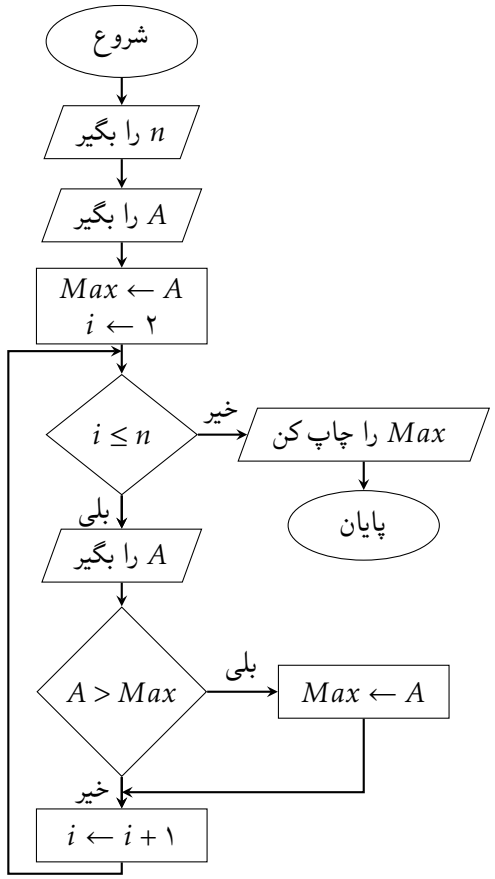
```
۱  شروع
۲  A و B را بگیر
۳  اگر A > B آنگاه
۴      C ← A
۵      A ← B
۶      B ← C
۷  پایان اگر
۸  اگر B ٪ A ≠ ۰ آنگاه
۹      r = B ٪ A
۱۰     B ← A
۱۱     A ← r
۱۲     برو به خط ۸
۱۳ پایان اگر
۱۴ A را چاپ کن
۱۵ پایان
```



شکل ۱۱.۲: فلوچارت محاسبهٔ ب.م.م. دو عدد صحیح وارد شده به روش نردبانی.

مثال ۱۲.۵.۲. *الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را به عنوان ورودی بگیرد و ماکزیمم آنها را محاسبه و چاپ کند.*

حل: برای یافتن ماکزیمم n عدد کافی است، اعداد را یکی یکی بخوانیم. اولین عدد را به عنوان ماکزیمم در نظر می‌گیریم و اعداد بعدی را با این ماکزیمم مقایسه می‌کنیم و در صورتی که این عدد بزرگ‌تر از ماکزیمم باشد، آن عدد را به عنوان ماکزیمم در نظر می‌گیریم. در الگوریتم از متغیر Max برای نگهداری ماکزیمم و از متغیر i برای شمارندهٔ حلقه استفاده می‌کنیم. □



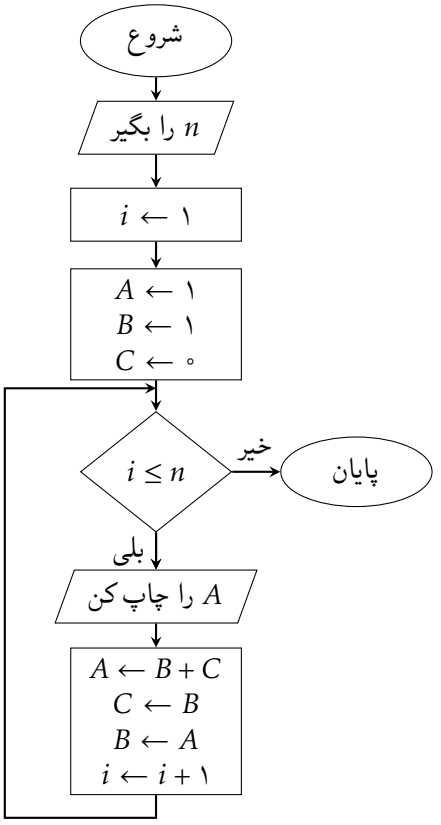
الگوریتم ۱۷.۲: الگوریتم محاسبهٔ ماکزیمم n عدد

```
۱  شروع
۲  n را بگیر
۳  A را بگیر
۴  Max ← A
۵  i ← ۲
۶  اگر i ≤ n آنگاه
۷  |  A را بگیر
۸  |  اگر A ≥ Max آنگاه
۹  |  |  Max ← A
۱۰ |  پایان اگر
۱۱ |  i ← i + ۱
۱۲ |  برو به خط ۶
۱۳ پایان اگر
۱۴ Max را چاپ کن
۱۵ پایان
```

شکل ۱۲.۲: فلوچارت محاسبهٔ ماکزیمم اعداد وارد شده.

الگوریتم ۱۸.۲: الگوریتم محاسبه و چاپ n جملهٔ اول دنبالهٔ فیبوناچی

۱ **شروع**
۲ n **را بگیر**
۳ $i \leftarrow ۱$
۴ $A \leftarrow ۱, B \leftarrow ۱, C \leftarrow ۰$
۵ **اگر** $i \leq n$ **آنگاه**
۶ | A **را چاپ کن**
۷ | $A \leftarrow B + C$
۸ | $C \leftarrow B$
۹ | $B \leftarrow A$
۱۰ | $i \leftarrow i + ۱$
۱۱ | برو به خط ۵
۱۲ **پایان اگر**
۱۳ **پایان**



شکل ۱۳.۲: فلوچارت محاسبهٔ n جملهٔ اول دنبالهٔ فیبوناچی.

مثال ۱۳.۵.۲. *الگوریتم و فلوچارتی ارائه کنید که عدد n را به عنوان ورودی بگیرد و n جملهٔ اول دنبالهٔ فیبوناچی را محاسبه و چاپ کند.*

حل: می‌دانیم که دو جملهٔ اول دنبالهٔ فیبوناچی برابر با ۱ و هر جملهٔ بعدی برابر با مجموع دو جملهٔ قبل از آن است. برای محاسبهٔ جملات دنبالهٔ فیبوناچی، از یک روند تکراری استفاده می‌کنیم. از متغیر i برای نگهداری تعداد جملات تولید شده از دنباله استفاده می‌کنیم و از متغیرهای A، B و C به ترتیب برای نگهداری آخرین جملهٔ دنبالهٔ فیبوناچی که تاکنون

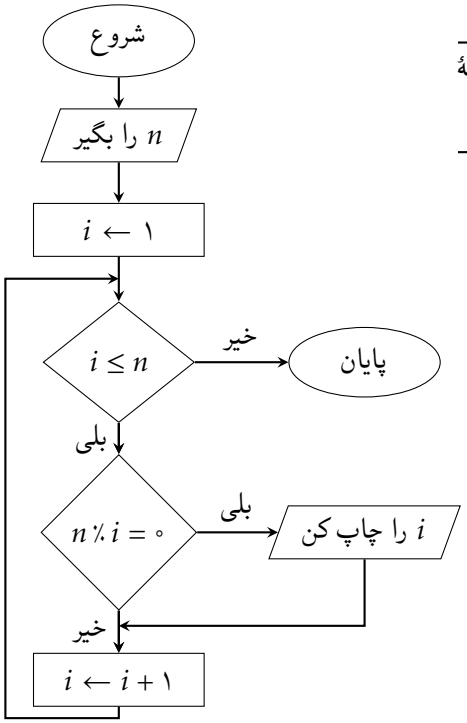
محاسبه شده و دو جملهٔ قبلی استفاده شده است. دقت کنید که بعد از تولید جملهٔ جدید، باید متغیرهای B و C به صورت مناسب تغییر کند تا دو جملهٔ قبلی حالت فعلی را نگهداری کند. □

مثال ۱۴.۵.۲. *الگوریتم و فلوچارتی ارائه کنید که عدد صحیح n را به عنوان ورودی بگیرد و تمام مقسوم علیه‌های آن را محاسبه و چاپ کند.*

حل: برای این منظور مشابه قبل، تمام مقادیری که ممکن است مقسوم علیه عدد وارد شده باشند را بررسی و به شرط داشتن خاصیت، آنها را چاپ می‌کنیم. بدیهی است مقسوم علیه‌های عدد n، بین ۱ تا n است. لذا الگوریتم و فلوچارت مربوطه به صورت زیر است.

الگوریتم ۱۹.۲: الگوریتم محاسبهٔ مقسوم علیه‌های عدد n.

```
۱  شروع
۲  n را بگیر
۳  i ← ۱
۴  اگر i ≤ n آنگاه
۵  |  اگر n ٪ i = ۰ آنگاه
۶  |  |  i را چاپ کن
۷  |  پایان اگر
۸  |  i ← i + ۱
۹  |  برو به خط ۴
۱۰ پایان اگر
۱۱ پایان
```



شکل ۱۴.۲: فلوچارت محاسبهٔ مقسوم علیه‌های عدد n.

با اندکی تغییر در این الگوریتم، می‌توان الگوریتمی یافت که یک عدد را دریافت کرده و تشخیص دهد کامل است یا خیر. منظور از یک عدد کامل؛ عددی است که با مجموع مقسوم علیه‌های کوچک‌تر از خودش برابر باشد. (به عنوان مثال ۶ یک عدد کامل است زیرا مقسوم علیه‌های کوچک‌تر از خودش اعداد ۱ و ۲ و ۳ هستند). برای این منظور کافی است در الگوریتم فوق، به جای چاپ مقسوم علیه‌های عدد، مجموع آنها را محاسبه کرد و در نهایت مقدار آن را با خود عدد مقایسه کرد. جزئیات این الگوریتم به عنوان تمرین به خواننده واگذار می‌شود. □

مثال ۱۵.۵.۲. *الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را به عنوان ورودی بگیرد و تعداد ارقام آن را محاسبه و چاپ کند.*

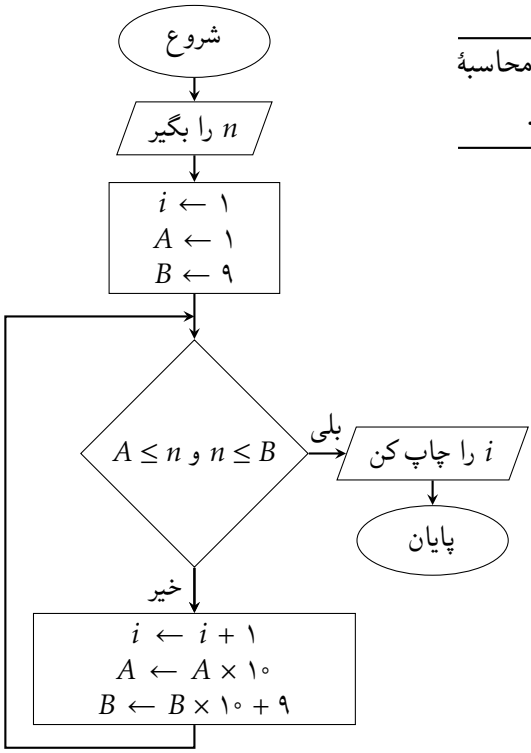
حل: شاید این اولین مثالی باشد که راه حل مسئله خیلی واضح نیست و لازم است برای آن روشی ارائه کرد. اولین روشی که شاید به ذهن برسد، بررسی قرار گرفتن عدد وارد شده در بازه اعداد تک رقمی؛ یعنی بین ۱ و ۹، اعداد دو رقمی؛ یعنی بین ۱۰ و ۹۹، و الی آخر است. با توجه به تعداد زیاد این بازه‌ها، بررسی تک تک آنها منطقی نیست. البته می‌توان آن را با استفاده از تکرار یک سری دستورات انجام داد. برای این منظور بازهٔ $[A, B]$ را در نظر می‌گیریم که مقدار A و B با ترتیب ۱ و ۹ است. از متغیر i نیز برای نگهداری تعداد ارقام اعداد در این بازه استفاده می‌کنیم که به وضوع در ابتدا باید با ۱ مقداردهی شود. اگر عدد n در بازهٔ $[A, B]$ باشد، آنگاه i چاپ شده و الگوریتم پایان می‌یابد. در غیر این صورت بازهٔ $[۱۰ \times A, ۱۰ \times B + ۹]$ را در نظر گرفته و به i یکی اضافه می‌کنیم. با تکرار این روند، در نهایت به بازه‌ای خواهیم رسید که n در بازه بررسی شده است و i در آن حالت جواب مسئله است. الگوریتم ۲۰.۲ و فلوچارت ۱۵.۲ این روند را نشان می‌دهد.

روش دیگری که این مسئله را می‌توان حل کرد به صورت زیر است. باقیمانده تقسیم هر عدد طبیعی بر ۱۰ برابر با رقم یکان آن عدد است. از طرفی دیگر خارج قسمت تقسیم یک عدد طبیعی بر عدد ۱۰، عددی است که رقم یکان عدد قبلی را ندارد. حال اگر ما طی یک روندی، عدد n را بر ۱۰ تقسیم کنیم و خارج قسمت تقسیم را دوباره در n نگهداری کنیم، با هر بار تکرار یکی از ارقام n کم می‌شود. اگر این روند را تا صفر شدن n ادامه دهیم

و با شمارنده‌ای تعداد مراحل را بشماریم، بوضوح، تعداد ارقام عدد n مشخص می‌شود. الگوریتم ۲۱.۲ و فلوچارت ۱۶.۲ این روش را با جزئیات کافی بیان می‌کنند. لازم به ذکر است که برای محاسبهٔ خارج قسمت تقسیم عدد صحیح A بر عدد صحیح B، کافی است جزء صحیح تقسیم A بر B را محاسبه کنیم.

الگوریتم ۲۰.۲: الگوریتم محاسبهٔ تعداد ارقام عدد طبیعی n.

۱ **شروع**
۲ n **را بگیر**
۳ $i \leftarrow ۱$
۴ $A \leftarrow ۱, B \leftarrow ۹$
۵ **اگر** $A \leq n$ و $n \leq B$ **آنگاه**
۶ | i **را چاپ کن**
۷ | **پایان**
۸ **پایان اگر**
۹ $i \leftarrow i + ۱$
۱۰ $A \leftarrow A \times ۱۰$
۱۱ $B \leftarrow B \times ۱۰ + ۹$
۱۲ برو به خط ۵



شکل ۱۵.۲: فلوچارت محاسبهٔ تعداد ارقام عدد طبیعی n.

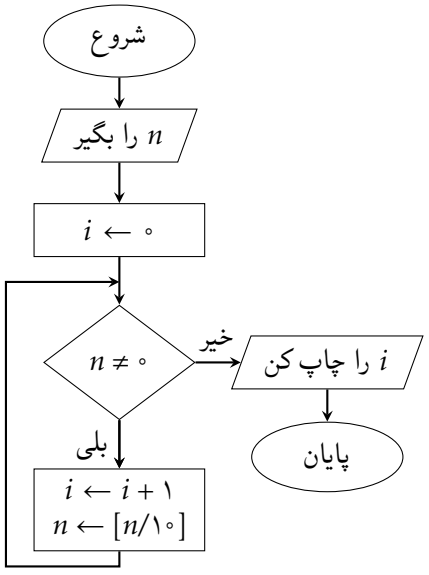
با توجه به تردیدی که ممکن است نسبت به درستی عملکرد این الگوریتم وجود داشته باشد، اجازه دهید این الگوریتم را برای یک عدد آزمایش کنیم. فرض کنید ورودی الگوریتم عدد ۱۰۳۷۰ باشد. جدول زیر، تغییرات انجام شده روی متغیرها را در روند اجرای الگوریتم نشان می‌دهد. جهت صرفه‌جویی در فضا، جدول به صورت افقی آورده شده است.

n	۱۰۳۷۰	۱۰۳۷	۱۰۳	۱۰	۱	۰
i	۰	۱	۲	۳	۴	۵

□

الگوریتم ۲۱.۲: الگوریتم روش دوم محاسبهٔ تعداد ارقام عدد طبیعی n.

۱ **شروع**
۲ n **را بگیر**
۳ $i \leftarrow ۰$
۴ **اگر** $n \neq ۰$ **آنگاه**
۵ $\quad i \leftarrow i + ۱$
۶ $\quad n \leftarrow [n/۱۰]$
۷ $\quad$ برو به خط ۴
۸ **پایان اگر**
۹ i **را چاپ کن**
۱۰ **پایان**



شکل ۱۶.۲: فلوچارت روش دوم محاسبهٔ تعداد ارقام عدد طبیعی n.

تمرین

۱. الگوریتمی و فلوچارتی ارائه کنید که n عدد را دریافت کند و ماکزیمم و می‌نیمم اعداد وارد شده را محاسبه و چاپ کند.

۲. الگوریتمی و فلوچارتی ارائه کنید که ضرایب یک چندجمله‌ای از درجه ۲ برحسب x و مقدار x را بخواند و مقدار چندجمله‌ای را به‌ازای مقدار داده شدهٔ x محاسبه و چاپ کند.

۳. الگوریتم و فلوچارتی ارائه کنید که n را دریافت کند و n جملهٔ اول دنبالهٔ زیر را

محاسبه و چاپ کند.

$$۱, ۲, ۴, ۸, ۱۶, ۳۲, \ldots, ۲^n.$$

۴. الگوریتم و فلوچارتی ارائه کنید که n را دریافت کرده و $\frac{۱}{n!}$ را محاسبه و چاپ کند.

۵. الگوریتم و فلوچارتی ارائه کنید دو عدد صحیح (مثبت یا منفی) را دریافت کرده و حاصلضرب آنها را بدون استفاده از عمل ضرب محاسبه و چاپ کند.

۶. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و مشخص کند آیا فاکتوریل عددی هست یا خیر.

۷. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و حاصل عبارت زیر را محاسبه و چاپ کند.

$$۰! + ۱! + ۲! + \cdots + n!.$$

۸. الگوریتم و فلوچارتی ارائه کنید که دو عدد طبیعی m و n را دریافت کرده و مشخص کند آیا دو عضو متوالی دنبالهٔ فیبوناچی هست یا خیر؟

۹. فرض کنید $a_۰ = ۰$، $a_۱ = a_۲ = ۱$ و جملات بعدی دنباله مجموع سه جمله قبلی آنهاست. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و nامین جمله دنباله را محاسبه و چاپ کند.

۱۰. الگوریتم و فلوچارتی ارائه کنید که یک عدد دودیی (باینری) را دریافت کرده و آن را به مبنای ۸، ۱۰ و ۱۶ تبدیل کند.

۱۱. الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد طبیعی n و سپس n عدد اعشاری را دریافت کرده و ماکزیمم، مینیمم و میانگین اعداد وارد شده را محاسبه و چاپ کند.

۱۲. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و سپس تمام اعداد صحیح کمتر از ۱۰۰۰۰ که دقیقا n مقسوم علیه دارند را محاسبه و چاپ کند.

۱۳. الگوریتم و فلوچارتی ارائه کنید که یک عدد صحیح را دریافت کرده و مجموع تمام مقسوم علیه‌های متمایز آن را محاسبه و چاپ کند.

۱۴. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و مشخص کند n کامل است یا خیر. عددی کامل است که مجموع تمام مقسوم علیه‌های آن، به غیر از خودش، با خودش برابر باشد. عدد ۶ مثالی از عدد کامل است.

۱۵. الگوریتم و فلوچارتی ارائه کنید که دو عدد صحیح را دریافت کرده و تمام مقسوم علیه‌های مشترک اول آنها را محاسبه و چاپ کند.

۱۶. الگوریتم و فلوچارتی ارائه کنید که مجموع دو کسر $\frac{a}{b}$ و $\frac{c}{d}$ را محاسبه و چاپ کند. کسر حاصل باید تا حد امکان ساده شده باشد.

۱۷. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n و سپس n عدد را دریافت کرده و سپس k را دریافت کرده و kامین کوچکترین عدد وارد شده را محاسبه و چاپ کند.

۱۸. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و حاصل عبارت زیر را محاسبه و چاپ کند.

$$\frac{۱}{۰!} + \frac{۱}{۱!} + \frac{۱}{۲!} + \cdots + \frac{۱}{n!}.$$

تمرینات تکمیلی

۱. الگوریتم و فلوچارتی ارائه کنید که عدد n را بگیرد و مجموع زیر را محاسبه و چاپ کند.

$$S_n = \sum_{i=۱}^{n} \frac{i}{i+۱} = \frac{۱}{۲} + \frac{۲}{۳} + \cdots + \frac{n}{n+۱}.$$

۲. دنباله‌ای را در نظر بگیرید که جملهٔ اول آن ۱، جملهٔ دوم آن ۲ است و جملات بعدی هرکدام میانگین دو جملهٔ قبلی است. الگوریتم و فلوچارتی ارائه کنید که عدد n را بگیرد و جملهٔ nام دنبالهٔ فوق را محاسبه و چاپ کند.

۳. الگوریتم و فلوچارتی ارائه کنید که سه عدد متمایز را از ورودی بخواند و همهٔ جایگشت‌های ممکن آن اعداد را درخروجی چاپ کند.

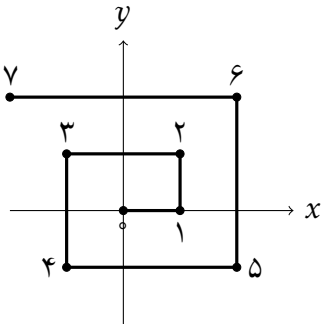
۴. اعداد صحیح m و n را متحابه گویند اگر مجموع تمام مقسوم علیه‌های m به جز خود m برابر با n و مجموع تمام مقسوم علیه‌های n به جز خود n برابر با m باشد. الگوریتم و فلوچارتی ارائه کنید که دو عدد را دریافت کرده و مشخص کند آیا متحابه هستند یا خیر.

۵. الگوریتم و فلوچارتی ارائه کنید که عدد طبیعی n را دریافت کرده و تمام اعداد طبیعی کوچک‌تر از ۱۰۰۰ را که با n متحابه هستند را محاسبه و چاپ کند.

۶. الگوریتمی بنویسید که ابتدا دو عدد A و B $(A < B)$ را دریافت کند و سپس با دریافت یک عدد n، بازهٔ $[A, B]$ را به n زیربازه با طول مساوی تقسیم و هر زیربازه را به صورت جداگانه چاپ کند. به عنوان مثال $n = ۴$ در بازه $[۰, ۱]$ خروجی زیر را چاپ کند.

$$[۰, ۰/۲۵], [۰/۲۵, ۰/۵], [۰/۵, ۰/۷۵], [۰/۷۵, ۱]$$

۷. الگوریتم و فلوچارتی ارائه کنید که عدد n را بگیرید و با توجه به شکل زیر، مختصات نقطه nام را به صورت دوتایی مرتب (x, y) محاسبه و چاپ کند.



۸. الگوریتم و فلوچارتی ارائه کنید که یک عدد اعشاری را از ورودی دریافت کرده و قسمت‌های صحیح واعشاری آن را به صورت دو عدد صحیح مجزا محاسبه کرده و چاپ کند.

۹. الگوریتم و فلوچارتی ارائه کنید که عدد n را از ورودی دریافت و تمام اعداد کوچک‌تر از n را که بر۳ یا ۵ بخشپذیر هستند را محاسبه و چاپ کند.

۱۰. الگوریتم و فلوچارتی ارائه کنید که دو عدد A و B را دریافت کرده و تمام اعداد دنبالهٔ فیبوناچی که بین A و B هستند را چاپ کند.

۱۱. الگوریتم و فلوچارتی ارائه کنید که ابتدا n و سپس n عدد را دریافت کرده مشخص کند چه تعداد از آن اعداد در دنبالهٔ فیبوناچی هستند.

۱۲. الگوریتم و فلوچارتی ارائه کنید که تاریخ تولد شخصی را (به صورت روز، ماه و سال) و تاریخ امروز را از ورودی دریافت کرده و مشخص کند شخص، چند روز عمر دارد.(همه ماه‌ها را ۳۰ روز فرض کنید.)

۱۳. برنامه‌ای بنویسید که یک عدد (صحیح یا اعشاری) را از ورودی دریافت کرده و مقلوب آن را محاسبه و چاپ کند.

۱۴. برنامه‌ای بنویسید که n را دریافت کند و مجموع زیر را محاسبه و چاپ کند؟

$$S = \frac{۱}{۲} - \frac{۲}{۳} + \frac{۳}{۴} - \cdots + \frac{n-۱}{n}.$$