

دانش اگر در ثریا هم باشد مردمانی از سرزمین پارس بر آن دست خواهند یافت.

پیامبر اعظم (ص)

شرح جامع

طراحی الگوریتمها

«مهندسی کامپیوتر – مهندسی فناوری اطلاعات IT– علوم کامپیوتر»

مولف: محمد کدخدا



پیشگفتار ناشر

خداوند بزرگ را شاکریم که فرصتی عنایت فرمود تا بتوانیم قدمی در جهت گسترش علم و فرهنگ کشور عزیزمان برداریم و مسیری هموار را در اختیار مخاطبان و داوطلبان علاقمند به ادامه تحصیل قرار دهیم. از آنجائی که درحوزه کتاب‌های درسی و کمک درسی کارشناسی ارشد کتاب‌های زیادی به رشته تحریر در آمده است با این حال وجود یک مجموعه جامع، دقیق، کامل و دارای رفرنس(منابع) مستند احساس می‌شود.

همیشه دغدغه داشتیم که ای کاش می‌شد تمامی منابع پراکنده و گاهاً دست نایافتنی کنکور در یک مجموعه به گونه‌ای جذاب و شیوا آماده می‌شد و همچنین سوالات دوره‌های قبل کنکور نیز به گونه‌ای تشریحی پاسخ داده می شدند، که خدشه‌ای بر آنها وارد نشود و خیال داوطلبان عزیز آنقدر آسوده باشد که اطلاعات مندرج در پاسخنامه‌ها را برای بهره‌گیری در آزمون خود بکار گیرند. این کار با یاری خداوند بزرگ و همکاری نخبه‌ترین دانشجویان و اساتید کشور در انتشارات ارشد هماهنگ گردید و در تمامی مقاطع تحصیلی از کاردانی به کارشناسی، کارشناسی ارشد و دکتری در حوزهٔ آزمون‌های سراسری – آزاد و پیام نور بسته‌های آموزشی شامل کتاب‌های بانک سوالات و کتاب‌های شرح جامع درس آماده گردید. در تهیه مجموعه‌های فوق‌الذکر از فارغ‌التحصیلان کارشناسی ارشد و دکتری که حائز رتبه‌های تک رقمی کنکور می‌باشند، استفاده کرده‌ایم، تا شما عزیزان بتوانید در راستای افزایش توان علمی خود در جهت نیل به مقصود خویش نایل آیید. بدون تردید می‌توان گفت که برای اولین بار در نظام آموزشی کشور مجموعه‌ای به این سبک عرضه می‌شود که در کیفیت و کمیت بی‌نظیر بوده و تمامی نیازهای دانشجویان و داوطلبان عزیز را پوشش می‌دهد. امید است که استفاده از این مجموعه عزیزان را در جهت موفقیت در آزمون یاری رسانده باشد. در پایان لازم به ذکر است که بی نقصی محض از آن خداوند متعال می‌باشد، بنا براین از تمامی خوانندگان و داوطلبان عزیز خواهشمندیم نظرات سازنده خود را در مورد این کتاب به سایت انتشارات ارشد اعلام نمایند تا ما را در جهت هر چه بهتر آماده کردن این محصولات یاری رسانند.

با تشکر

ناشـــــر

www.arshadbook. ir

تقدیم به:

مهربان مادرم

پیشگفتار مولف

مجموعه‌ی حاضر که به ارائه‌ی شرح جامعی از درس طراحی الگوریتم‌ها می‌پردازد حاصل تلاش مؤلف برای ایجاد یک متن علمی سودمند برای دانشجویان دوره‌های کارشناسی رشته‌های مرتبط با کامپیوتر بود. که نگارنده خود در ضمن پرداختن به الگوریتم‌ها و از ما بین منابع موجود فراهم ساخته است. برخی از جنبه‌های آن مانند بخش اول، برای بسیاری تازگی دارد، و اگر چه یادداشت‌های این بخش از عمق چندان بالایی برخودار نیست اما زمینه‌ی خوبی برای ورود به حل مسائل الگوریتم‌ها ایجاد می‌کند. محتوای دیگر بخش‌ها نیز غالباً در کتاب‌های طراحی الگوریتم به چشم می‌خورند. آنچه می‌تواند این اثر را سودمند سازد، نحوه‌ی پرداختن آن به جنبه‌های مختلف الگوریتم‌ها می‌باشد.

حل مثالهای متنوع و ارائه‌ی تمرین‌های خوب در پایان مباحث هر درس و گزیده‌ای از سوالات آزمون‌های ورودی دوره‌های تحصیلات تکمیلی در پایان هر بخش می‌تواند برای علاقه‌مندانش مفید باشد. این اثر مورد بازبینی قرار گرفته اما شاید همچنان اشتباهاتی در آن یافت شود که امید است خوانندگان محترم اشکالات را با مؤلف یا انتشارات ارشد در میان بگذارند تا در چاپ‌های بعدی مورد اصلاح قرار گیرد. در پایان از تمامی کسانی که در تهیه و چاپ این اثر نقش داشته‌اند کمال تشکر را دارم.

محمد کدخدا

فهرست مطالب

عنوان **صفحه**

پیشگفتار ناشر ۳
پیشگفتار مولف ۵

بخش اول: درآمدی بر ریاضیات

فصل اول: نظریه‌ی مجموعه‌ها ۱۵
فصل دوم: مبانی حساب ۲۱
فصل سوم: نظریه اعداد ۲۶
فصل چهارم: نظریه‌ی بازگشت ۳۰
فصل پنجم: نظریه‌ی گراف ۴۴
فصل ششم: نظریه‌ی احتمال ۴۹
تست‌های آخر بخش ۵۱
پاسخ تست‌های آخر بخش ۵۴

بخش دوم: ساختمان داده‌ها

فصل هفتم: آرایه‌ها ۶۱
فصل هشتم: پشته، صف ولیست پیوندی ۶۸
فصل نهم: انواع درخت‌ها ۷۸
فصل دهم: گراف‌ها ۸۶
تست‌های آخر بخش ۸۹
پاسخ تست‌های آخر بخش ۹۲

بخش سوم: الگوریتم‌های پایه

فصل یازدهم: جستجوی شاخص‌های آماری ۹۷
فصل دوازدهم: جستجوی دودویی ۱۰۳
فصل سیزدهم: جستجو درگراف ۱۰۸
فصل چهاردهم: جداول درهم سازی ۱۱۵
فصل پانزدهم: الگوریتم‌های مرتب سازی ۱۱۸
فصل شانزدهم: الگوریتم های مرتب سازی خاص ۱۲۸
فصل هفدهم: تحلیل الگوریتم ها ۱۳۲
تست‌های آخر بخش ۱۴۱
پاسخ تست‌های آخر بخش ۱۴۴

عنوان **صفحه**

بخش چهارم: روشهای طراحی الگوریتم

فصل هجدهم: روش تقسیم و غلبه ۱۵۱
فصل نوزدهم: برنامه‌سازی پویا ۱۵۹
فصل بیستم: روش حریصانه ۱۷۵
تست‌های آخر بخش ۱۸۷
پاسخ تست‌های آخر بخش ۱۹۱

منابع و مآخذ ۱۹۴

بخش اول

درآمدی بر ریاضیات

عناوین اصلی

- نظریه‌ی مجموعه‌ها
- مبانی حساب
- نظریه‌ی اعداد
- نظریه‌ی بازگشت
- نظریه‌ی گراف
- نظریه‌ی احتمال

مقدمه

استفاده از روش های الگوریتمیک در حل مسایل یکی از ارزشمندترین دستاوردهای علمی است. عنوان الگوریتم، تغییر شکل یافته‌ی نام الخوارزمی، یکی از دانشمندان ایرانی سده های دوم و سوم هجری قمری می باشد. محمدبن موسی، معروف به الخوارزمی در اثر ریاضی خود به نام المختصر فی الحساب الحیر و المقابله به ابداع این روش پرداخته و برای اولین بار تفکر الگوریتمیک را در حل مسایل بکار برده است، امروزه پس از قرنها این علم به طور چشمگیری توسعه یافته و در زمینه های مختلفی جای باز کرده است.

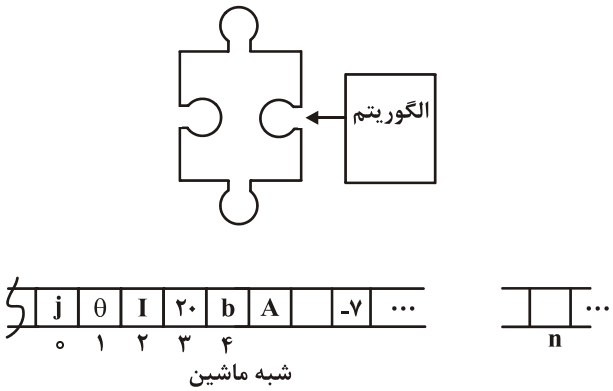
مطالعه ی الگوریتم ها در علوم کامپیوتر ساخت‌یافته و روش‌مند است، دارای اصطلاحات و تعاریف دقیقی است و از چارچوب های خاصی نیز تبعیت می کند. به همین خاطر می توان گفت کاربرد و یادگیری الگوریتم ها خود یک الگوریتم است بطوریکه در مواجهه با حل هر مسأله ای کارهای خاصی را دیکته می‌کند. اگر چه خلاقیت هم در این میان تأثیرگذار است.

یک الگوریتم از جنبه ی چیستی اش، **"را ه حلی کلی است. برای یک موضوع خاص که روی تمام نمونه های آن نتیجه مورد انتظار را بدست می دهد.** "و از نظر چگونگی" **چینش دستوراتی روشن، شدنی و پایان پذیر برای تولید یک یا چند نتیجه"** می باشد. روشن بودن مربوط به قطعیت آن می شود تا امکان دو برداشت (هر چند شدنی) از آن وجود نداشته باشد شرط شدنی بودن دستورات به آن معناست که امکان انجامش برای آنچه که قرار است آن را اجرا کند فراهم باشد. مثلاً اگر از شما بخواهند دو ملکول هیدروژن را از اکسیژن آب جدا کنید این امکان را ندارید اما شاید یک دستگاه تجزیه ی شیمیایی از این امکان بهره مند باشد. در نهایت باید کار اجرای دستورات به سرانجامی برسد و متوقف شود. نتیجه نیز مورد انتظار است و ما باید با انجام یک الگوریتم به چیزی رسیده باشیم و الا چرا آن را انجام می دهیم. یک الگوریتم باید بتواند تمام این شرطها را برآورده سازد.

بیان دستورات نیازمند یک زبان است، به نحوی که برای طراح آنقدر غنی باشد که هر چه را می خواهد با آن بیان کند و برای ماشینی که قرار است آن را اجرا کند کارا، تا از همه ی توانمندیهای آن بهره بگیرد. زبانی که ما در اینجا آن را به کار خواهیم برد شبه کدهایی است که هر چیزی را تا آنجا که منظور را برساند مختصر و ساده بیان می نماید.

اجرای الگوریتم ها نیز بر عهده ی یک **شبه ماشین** است که به یک نوار حافظه دسترسی داشته که از یک طرف تا بی نهایت امتداد دارد و هرخانه آن با یک عدد طبیعی به عنوان ایندکس شناخته می شود در یک خانه از حافظه داده ای از هر نوع می تواند جا بگیرد. در حالت طبیعی **شبه ماشین** به هر خانه ی حافظه مستقیماً دسترسی دارد مگر اینکه دسترسی توسط ساختمان داده ی ارائه شده روی مجموعه ای از خانه های حافظه با محدودیت هایی مواجه باشد.

شبه ماشین قابلیت انجام تمام اعمال پایه ساختمان داده ها را دارد. و اجرای هر دستور پایه یک واحد از زمان آن را می گیرد. نحوه ی عملکرد شبه ماشین بر خلاف ماشین به رفتار یک فرد در انجام یک الگوریتم نزدیک است به همین دلیل ما اغلب به جای کلمه ی اجرا، از پیگیری دستورات استفاده می کنیم. تصور رفار شبه ماشین در اجرای یک الگوریتم به خواننده کمک می‌کند تا چگونگی انجام یک الگوریتم را بهتر دریابد.



شبه ماشین

الگوریتم ها می توانند ورودی نداشته باشند اما اگر به ورودی نیازمند باشند، هر ورودی یک الگوریتم را یک **نمونه** از آن الگوریتم می نامیم. برای اینکه نحوه ی کار الگوریتمی را روی یک نمونه اش ببینیم، با تقلید رفتار شبه ماشین، فرآیند اجرا را دنبال کرده و در هر مرحله وضعیت را ثبت می کنیم. برای سهولت تنها به ثبت تغییرات می پردازیم. در این میان می توانیم از جداول و اشکالی نیز برای بیان وضعیت ها استفاده نمائیم.

مجموعه حاضر از چهار بخش تشکیل شده است که در زیر به نقش هر یک از این بخش‌ها اشاره می کنیم.

۱- ریاضیات: ریاضیات پیش از آنکه مبنایی برای الگوریتم ها بحساب آید، زبانی برای بیان مسأله ها است و به این ترتیب با الگوریتمی که برای پاسخ به آن سؤال طراحی می شود گره می خورد. شروع کار با ریاضیات می تواند بسیار امیدوار کننده باشد تا نقشی که ریاضیات به طور مستقل در رابطه با الگوریتم ها ایفا می کند را بهتر دریابیم. هر چند این نقش جنبه های مختلفی دارد و از میان نظریه هایی که ارائه شده اند دسته ای تنها در حد یک قالب مطرح اند که به شکل دهی برخی انواع داده ی مجرد در ساختمان های داده کمک می‌کنند. برخی ایده هایی برای ساخت الگوریتم ها به ما می دهند و تعدادی نیز به عنوان ابزاری در تحلیل الگوریتم ها مورد استفاده هستند گردآوری همه‌ی آنها در یک بخش ما را در قسمت های دیگر جلو خواهد برد و به یکنواختی بحث ها نیز کمک خواهد کرد.

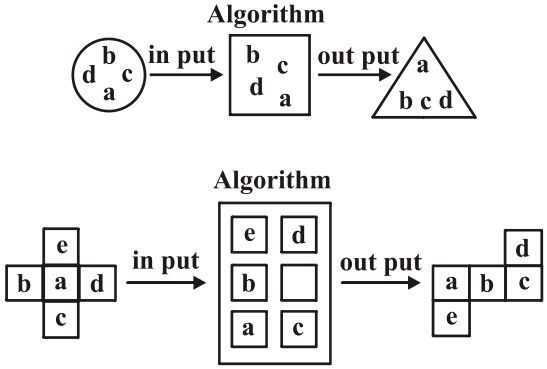
۲- ساختمان دادها: ساختمان داده ها در ارتباط با نمایش و دستکاری داده ها مطرح اند و کار هر الگوریتم در تعامل با ساختاری (یا ساختارهایی) که داده های اولیه را در خود جای داده تعریف می شود به بیانی دیگر یک الگوریتم چیزی جز دستکاری کننده داده‌ها از راه های موجود توسط ساختار ها نیست تا یا در آرایش آنها تغییری ایجاد کند و یا یک نتیجه ی جدیدی تولید نماید. که هر دو را نیز در آخر کار در یک ساختار به ما تحویل خواهد داد.

به این ترتیب قابلیت های یک الگوریتم به ساختار داده های آن محدود می شود هم از جنبه ی دسترسی و هم ارائه نتیجه و تفسیر آن.

مثلاً اگر قرار باشد یک الگوریتم داده هایی با محدودیت سمت چپ را به نحوی تغییر دهد که در آن حروف صدا دار نسبت به هم a,b,c نسبت به هم و c,d با نسبت معکوس هم به طور قاموسی مرتب باشند. باید همانند شکل عمل شود بخشی از الگوریتم ها کاری به جز این ندارند. به همین دلیل بحث و بررسی این ساختارها اهمیت بسیاری دارد.

ساختمان های داده بر مفهوم نوع داده ی انتزاعی (ADT) بنا می شوند یک نوع داده انتزاعی عبارت است از " داده هایی با ارتباط های تعریف شده و مجموعه ای از اعمال روی این داده ها که با آن سازگاری دارد." به طور خلاصه دوتایی (A,F) را که A کلاسی از داده ها و F مجموعه ای عمل که روی عناصر A واقع می شود. نیاز ما به انواع مختلفی از داده ها سبب تعریف کلاس های متنوعی از A خواهد شد و عملهای F برای دستکاری های مورد نظر توسط یک الگوریتم به کار

گرفته می شوند بخشی از این عمل ها، پایه اند و برخی ترکیبی، عمل های پایه مستقیماً قابل انجام هستند اما برای عمل های ترکیبی باید از روالهایی که با استفاده از اعمال پایه طراحی می شوند. استفاده می‌شود.



۳- الگوریتم های پایه: این بخش یک شروع جدی از مباحث الگوریتم هاست، در ابتدا الگوریتم های جستجو طرح و مورد بررسی قرار می گیرند و بعد از آن الگوریتم های مرتب سازی، این ترتیب مفهوماً قابل توجیه است و جستجو ابتدایی تر است از مرتب سازی.

این بخش با بحث تحلیل الگوریتم ها به پایان می رسد و در آن با ارائه ابزارهایی به تحلیل کلیه ی الگوریتم های آن بخش و بخش‌های قبل از آن پرداخته ایم. این بخش با مسایل بسیار زیادی همراه است که برای یادگیری بسیار مفیدند. موضوعات این بخش به عنوان سنگ بنای الگوریتم ها بشمار می رود.

۴- روش های طراحی: نخستین چیزیکه برای طراحی یک الگوریتم مهم است ایده ای است که بر آن حاکم می سازیم امروزه این ایده ها ساخت یافته شده و هر یک به عنوان روشی در طراحی الگوریتم‌ها مطرح می‌باشند که برای اعمال آن ها در طراحی مکانیزم هایی وجود دارد. استفاده از هر نگرشی در طراحی نیازمند شناخت قابلیت ها و نیز محدودیت های آن است، ما در این بخش با پیاده نمودن انواع روش های طراحی در یک الگوریتم های مختلف به بیان این قابلیت ها و نیز محدودیت ها خواهیم پرداخت، اینکه برای هر مسأله ی دیگری کدام روش مناسب تر است نیازمند تحلیل و ارزیابی دقیق هر یک از آنهاست . به ازای هر مسأله ای که برای آن یک الگوریتم طراحی کرده ایم چند تمرین برای حل توسط خواننده قرار داده ایم تا میزان درکش از نحوه ی پیاده سازی و جنبه های مختلف آن را محک بزند زیاد نبودن تمرین ها به آن خاطر است که پیشرفت خواننده سریع باشد.

در هر حال موضوعاتی در باب الگوریتم ها وجود دارند که در جریان بحث ها نمی توان جایی برای آن باز گردد. مسائل جالبی که با نبوغ سروکار دارند، ایده های ترکیبی برای طراحی و برخی بحث های جالب پیشرفته که می‌توانند با رجوع به منابع پیشنهادی تأمین شوند. در تمام بخشها، دقت و سادگی مورد نظر بوده اند و هر آنچه از آن صحبت شده را تا جایی که به معنای آن لطمه ای وارد نشود در نهایت اختصار بیان داشته ایم. هرچند ممکن است در مواردی یادداشت‌ها آنقدر واضح و شفاف به نظر نیاید ولی خواننده با کمی تأمل خواهد توانست تمامی آنها را دریابد. درسنامه‌ای بر الگوریتم‌ها تنها فرصتی گذرا به مطالب وسیعی از الگوریتم‌ها و جنبه‌های مهم آن دارد. و از اشاره‌هایی مفید، نگاه‌هایی تازه والگوریتم‌هایی متفاوت برخوردار است.

فصل اول

نظریه‌ی مجموعه‌ها

۱. نظریه‌ی مجموعه‌ها

اگرچه برای مجموعه در ریاضیات تعریفی وجود ندارد اما بیان زیر در بسیاری موارد کافی است.

«دسته‌ای از اشیاء که دارای یک ویژگی خاص بوده یا منظور مشترکی را برآورده ساخته و یا دنبال می‌کنند.»

به‌طور مثال مجموعه‌های زیر که هر یک را با حرفی نام‌گذاری کرده‌ایم اینگونه‌اند.

$$A = \{\text{Tehran}, \text{Number}۱۰۵۲, \text{Enghlab street}\}$$

$$B = \{X, Y, Z\}$$

$$N = \{۱, ۲, ۳, ...\}$$

مجموعه‌ی A از عناصری که آدرسی را مشخص می‌کنند، ترکیب یافته اعضای B برای کار خاصی انتخاب شده‌اند و مجموعه‌ی N را تنها اعداد طبیعی تشکیل داده‌اند. هر زیرمجموعه از یک مجموعه را یک کلاس از آن می‌نامند. عمل‌های زیادی در ارتباط با مجموعه‌ها مطرح‌اند از جمله اجتماع، اشتراک، حاصل ضرب و . . . . مجموعه‌ها مفهوماً در طراحی و تحلیل الگوریتم‌ها نقش دارند و هر الگوریتم خود به عنوان یک مجموعه مطرح است که اعضای آن را دستوراتی تشکیل می‌دهند که از یک جنبه ترتیبی نسبت به هم برخوردارند. به‌طور مثال مجموعه‌ی دستورات زیر را در نظر بگیرید.

$$D = \{Z \leftarrow X, Y \leftarrow Z, X \leftarrow Y\}$$

از هر آرایش این دستورات می‌توان یک روال را بدست می‌آورد. که عمل خاصی را انجام می‌دهد

1	exchang (X,Y)	equal to (Y)	eliminate (Z)
2	$Z \leftarrow X$	$X \leftarrow Y$	$Z \leftarrow X$
3	$X \leftarrow Y$	$Z \leftarrow X$	
4	$Y \leftarrow Z$		
	۳	۲	۱

- هر زیرمجموعه از یک حاصلضرب دکارتی دو مجموعه را یک رابطه می‌نامیم لذا از دید ریاضی هر الگوریتم به عنوان یک رابطه مطرح است، برای روال ۲ رابطه زیر را داریم.

$$\{\ (1\ , \text{equalto}(y)), (\ 2\ , x \leftarrow y), (\ 3\ , z \leftarrow x)\}$$

و هر نمونه‌ای ورودی یک الگوریتم نوع خاصی از رابطه است که در آن دو عضو مختلف با مؤلفه اول یکسان وجود ندارد. به این نوع رابطه‌ها تابع گفته می‌شود.

روال ۱، محتوای Z را پاک می‌کند. هرچند این کار با ریختن X در آن صورت می‌گیرد. روال دو محتوای همه‌ی متغیرها را یکی می‌کند و روال آخر محتوای متغیرهای X و Y را جابه‌جا می‌کند. هر چند در این میان Z با X پاک می‌شود. روال ۱، زیرروالی از ۲ و ۳ به حساب می‌آید. یک نمونه از ورودی‌های روال‌های فوق با ضرب مجموعه‌ی B در مجموعه‌ای از اعداد بدست می‌آید. به طور مثال داریم:

$$\{X,Y,Z\}\times\{۱,۲,۳,...\}=\{(X,۱),(Y,۱),(Z,۱),.....\}$$

$$\{(X,۵),(Y,۲),(Z,۵)\}\subseteq BX\mathbb{N}$$

خروجی روال‌های سه گانه برای نمونه داده شده به این صورت بدست می‌آید.

$$۱,\{(X,۵),(Y,۲),(Z,۵)\}$$

$$۲,\{(X,۲),(Y,۲),(Z,۲)\}$$

$$۳,\{(X,۲),(Y,۵),(Z,۵)\}$$

با همین مثال ساده روشن می‌شود که طراحی الگوریتم‌ها در حقیقت مدیریت مجموعه‌ای این چنین از دستورات است. البته الگوریتم‌های پیچیده با مجموعه‌ی بزرگ‌تری از دستورات ارتباط پیدا می‌کنند. گاهی این مجموعه آنقدر بزرگ است که برای یک طراحی خوب سعی می‌شود آن‌را به زیرمجموعه هایی تجزیه و با استفاده از هر یک در طراحی بخشی از الگوریتم اصلی هدف طراحی دنبال شود. این کار به عنوان نگرش **پیمانه‌ای** در طراحی مطرح است. در زیر مثال ساده‌ای از این نوع طراحی را می‌آوریم.

مثال:

برای ضرب دو عدد X و Y روش‌های جالبی وجود دارد یکی از آن‌ها روش راسی است. در این روش نیازمند دانستن فرد بودن یک عدد X می‌باشیم لذا روال زیر را برای آن ارائه می‌کنیم.

```
۱)  odd(X, ans)
۲)      Z ← X
۳)      while(Z > ۱)
۴)          Z ← Z − ۲
۵)      ans ← Z
```

خاصیت راسی نیز که مورد استفاده است و می‌تواند به صورت روال زیر ساخته می‌شود.

```
۱)  russe(A[۱,...,n], B[۱,...,n])
۲)      A[۱] ← X
۳)      B[۱] ← Y
۴)      while A[i] > ۱do
۵)          A[i + ۱] ← A[i] / ۲
۶)          B[i + ۱] ← B[i] + [i]
۷)              i ← i + ۱
```

حال روش ضرب راسی را به کمک روال‌های فوق اینگونه می‌نویسیم

```
۱)   russe Multiply(X, Y, P)
۲)       i ← ۱
۳)       p ← ۰
۴)       russe(A[۱,..., n], B[۱,..., n])
۵)       while i > ۰
۶)           if odd (A[i] , ans) then
۷)               p ← p + B[i]
۸)           i ← i − ۱
```

در زیر پیگیری الگوریتم ضرب فوق را برای یک نمونه آورده‌ایم.

russe

step	۰	۱	۲	۳	۴
i	۱	۲	۳	۴	۵
A[i]	۱۹	۹	۴	۲	۱
B[i]	۷	۱۴	۲۸	۵۶	۱۱۲

$X = ۱۹ \quad Y = ۷$

russe Multiply

i	۵	۴	۳	۲	۱	
ans	۱	۰	۰	۱	۱	
p	۱۱۲	۱۱۲	۱۱۲	۱۲۶	۱۳۳	

نتیجه‌ی نهایی ضرب دو عدد داده شده در p قرار می‌گیرد.

نگرش پیمانه‌ای به درک بهتر الگوریتم‌ها و تحلیل درست‌تری از آن‌ها منجر می‌شود. و زمانی که الگوریتم‌ها پیچیدگی درهم تنیده‌ی خود را حفظ کنند، تحلیل آن‌ها نخواهد توانست به این دقت و درستی انجام شود و نتیجه‌ای جز تقریب و خطا نخواهد داشت این موضوع زمانی که الگوریتم‌ها دارای مجموعه‌ی دستورات زیاد و متنوعی هستند بیشتر بروز خواهد کرد.

اثبات درستی عملکرد یک الگوریتم بحث بسیار مهمی است از کجا بدانیم که الگوریتم طراحی شده به ازای هر نمونه‌اش خروجی مورد نظر را به ما می‌دهد. به‌طور مثال به ازای هر دو عدد x و y از کجا می‌توان مطمئن بود که الگوریتم ضرب راسی، نتیجه حاصلضرب این دو عدد را به ما می‌دهد. روشن است که ما امکان بررسی تمام نمونه‌ها را برای تایید این موضوع نداریم چراکه نمونه‌ها بسیار زیادند. اما چه باید کرد یک راه بررسی منطق الگوریتم‌ها است. در این روش درستی الگوریتم‌ها به عنوان یک قضیه ادعا شده و سعی می‌شود با یک فعالیت ریاضی این موضوع اثبات شود. اما راه دیگری نیز هست که با مجموعه‌ها ارتباط دارد. در این روش به جای بررسی تأیید روی تمام نمونه‌ها این موضوع برای نماینده‌ای از کلاس‌های هم‌نهشتی نمونه‌ها انجام می‌گیرد. به عبارتی مجموعه‌ی نمونه‌ها می‌تواند به زیر مجموعه‌هایی افراز شوند که رفتار الگوریتم‌ها برای اعضای هر زیر مجموعه کاملاً مشابه است پس در صورت درستی عملکرد یک الگوریتم روی نماینده‌ای از هر کلاس همنهشتی آن برای تمام افرازها و در نتیجه برای تمام نمونه‌ها بدست می‌آید.

منظور از افراز یک مجموعه مانند A عبارت است از کلاسی از زیرمجموعه‌های A مانند $\{A_۱, A_۲, ..., A_n\}$ با دو ویژگی:

۱- برای هر i و j که $A_i \cap A_j = \phi \;,\; i \neq j$

۲- $\bigcup_{۱}^{n} A_i = A$

یکی دیگر از مواردی که مطالعه افرازها در آن بسیار اهمیت دارد، افراز توابع پیچیدگی و کلاس‌های مربوط به آن است. چراکه پیچیدگی هر الگوریتمی در حالات مختلف می‌تواند با یک تابع طبیعی بیان شود. مجموعه‌ی تمام چنین تابع‌هایی را COM می‌نامیم.

$$COM = \left\{ f \mid \mathbf{f} : \mathbb{N} \to R^* \right\}$$

که $\mathbb{N}$ همان مجموعه‌ی اعداد طبیعی است و R^* مجموعه اعداد حقیقی نامنفی می‌باشد.

افرازهای مجموعه‌ی توابع پیچیدگی بر اساس مفهوم، هم‌ارزی مجانبی صورت می‌گیرد. گفته می‌شود $f \in \theta(g(n))$ است یا در کلاس هم‌ارزی $g(n)$ قرار می‌گیرد اگر و تنها اگر

$$\left(\exists c,d \in \mathbb{R}^+\right)\left(\exists n_\circ \in \mathbb{N}\right)\left(\forall n \ge n_\circ\right)[c\mathbf{g}(n) \le \mathbf{f}(n) \le d\mathbf{g}(n)]$$

در اینجا R^+ به اعداد حقیقی مثبت اشاره دارد.

در این صورت $\mathbf{f}(n)$ و $\mathbf{g}(n)$ هم‌ارز مجانبی یکدیگر خوانده می‌شوند. هم‌ارزی مجانبی ترکیبی از دو شرط کف و سقف مجانبی است که برای هر تابع $\mathbf{g}(n)$ مجموعه‌ی توابعی که این شرط‌ها را دارند به ترتیب $\Omega(\mathbf{g}(n))$ و $O(\mathbf{g}(n))$ می‌نامیم که تعریف دقیق آن‌ها به این صورت است.

$$\Omega(\mathbf{g}(n)) = \left\{ \mathbf{t} : \mathbb{N} \to \mathbb{R}^* \mid (\exists C \in \mathbb{R}^+)(\exists n_\circ \in \mathbb{N})(\forall n \ge n_\circ)\left[\mathbf{t}(n) \ge c\mathbf{g}(n)\right] \right\}$$

$$O(\mathbf{g}(n)) = \left\{ \mathbf{t} : \mathbb{N} \to \mathbb{R}^* \mid (\exists d \in \mathbb{R}^+)(\exists n_\circ \in \mathbb{N})(\forall n \ge n_\circ)\left[\mathbf{t}(n) \le d\mathbf{g}(n)\right] \right\}$$

و هم‌ارزی مجانبی اشتراک این دو می‌باشد لذا:

$$\theta(\mathbf{g}(n)) = O(\mathbf{g}(n)) \cap \Omega(\mathbf{g}(n))$$

به‌طور مثال می‌توان هر یک از موارد زیر را اثبات نمود.

۱- $n^3 + n \in \Omega(n^3 + 2n^2 + n)$

اگر $c = \frac{1}{3}$ و $n_\circ = 1$ داریم:

$$cg(n) = \frac{1}{3}(n^3 + 2n^2 + n) \le \frac{1}{3}(n^3 + 2n^3 + n) \le n^3 + n$$

در نتیجه حکم ثابت شد.

۲- $n^2 + n\mathbf{Lg}n \in O(n^{2/1})$ اگر $d = 2$ و $n_\circ = 1$ داریم:

$$n^2 + n\mathbf{Lg}n \le n^2 + n^2 \le 2n^{2/1} \qquad \forall n \in \mathbb{N}$$

۳- $\mathbf{f}(n) + \mathbf{g}(n) \in \Omega(\min(\mathbf{f},\mathbf{g}))$ اگر $c = n_\circ = 1$ لذا داریم:

$$\min\big(\mathbf{f}(n),\mathbf{g}(n)\big)\le \mathbf{f}(n)+\mathbf{g}(n) \qquad \forall n\in\mathbb{N}$$

۴- $f(n)+g(n)\in O\big(\max(f,g)\big)$ اگر $d=۲$ و $n_۰=۱$ لذا داریم:

$$\begin{matrix}\mathbf{f}(n)\le \max\big(\mathbf{f}(n),\mathbf{g}(n)\big)\\ \mathbf{g}(n)\le \max\big(\mathbf{f}(n),\mathbf{g}(n)\big)\end{matrix} \quad \mathbf{f}(n)+\mathbf{g}(n)\le ۲\max\big(\mathbf{f}(n),\mathbf{g}(n)\big) \quad \forall n\in\mathbb{N}$$

۵- $n^۳+n^۲Lgn\in\theta(n^۳)$ اثبات این موضوع نیز با گرفتن $c=۱$ و $d=۲$ و $n_۰=۱$ داریم:

$$n^۳\le n^۳+n^۲Lgn\le ۲n^۳$$

که برقرار می‌باشد.

نمادهای Ω,O و θ به نمادهای مجانبی معروف‌اند و مهم‌ترین ابزار تحلیل پیچیدگی الگوریتم‌ها به شمار می‌آیند. نحوه‌ی استفاده از آن‌ها و تحلیل الگوریتم‌ها به کمک هر یک در درس ۱۷ مورد بررسی قرار می‌گیرد.