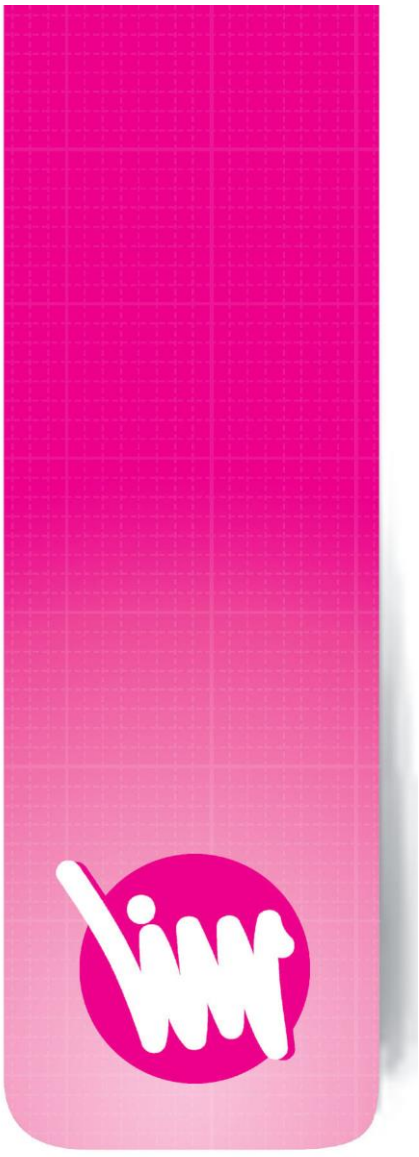

# درسنامه جامع ساختمان داده ها

نویسنده: مریم نجاتی گرائی

شابک : ۱۶۵۰۰۰۰ ریال۶-۶۳-۸۴۵۶-۶۰۰-۹۷۸:
شماره کتابشناسی ملی : ۴۵۶۰۴۸۰
عنوان و نام پدیدآور : درسنامه جامع ساختمان داده‌ها/ نویسنده مریم نجاتی‌گرائی.
مشخصات نشر : تهران: انتشارات علمی سنا، ۱۳۹۵.
مشخصات ظاهری : ۱۷۸ ص.: مصور، جدول، نمودار .
موضوع : ساختار داده‌ها -- راهنمای آموزشی (عالی)
موضوع : Data structures (Computer science) -- Study and teaching (Higher)
موضوع : ساختار داده‌ها -- مسائل، تمرین‌ها و غیره (عالی)
موضوع : Data structures (Computer science) -- Problems, exercises, etc. (Higher)
رده بندی دیویی : ۷۳۰۷/۰۰۵
رده بندی کنگره : QA۹/۷۶ ۱۳۹۵ ۳ن۲س/
سرشناسه : نجاتی گرائی، مریم، ۱۳۶۶ -
وضعیت فهرست نویسی : فیپا

**انتشارات علمی سنا (مرجع تخصصی علوم پزشکی )**

**نام کتاب: درسنامه جامع ساختمان داده‌ها**
**نویسنده: مریم نجاتی‌گرائی**
**ناشر: علمی سنا**
**شابک: ۶-۶۳-۸۴۵۶-۶۰۰-۹۷۸**
**نوبت چاپ: اول – ۱۳۹۵**
**صفحه آرایی: سحر زعفرانی عیلام**
**طراح جلد: هادی طغیانی**
**پست الکترونیک:** elmisana@gmail.com
**فروش اینترنتی (با تخفیف ) :** www. sanabook.com
**تیراژ: ۳۰۰**
**قیمت: ۳۵۰/۰۰۰ ریال**

فهرست:

# فصل اول:
# الگوریتم

**مقدمه:**

داده‌ها به صورت‌های مختلفی سازماندهی می‌شوند. مدل منطقی یا ریاضی خاصی را که به منظور سازماندهی داده‌ها به کار می‌رود، ساختمان داده‌ها نامیده می‌شود. ساختمان داده ها در واقع مبحثی است که به بررسی نحوه‌ی ذخیره سازی و آدرس‌دهی داده‌ها در حافظه و زمان مورد نیاز برای دسترسی به این داده‌ها می‌پردازد. با این توصیف، ساختمان داده‌ها مبحثی مستقل از نوع کامپیوتر و زبان‌های برنامه نویسی است. بر این اساس الگوریتم‌ها و برنامه‌های ارائه شده در این بخش هر چند که در قالب یک زبان برنامه‌نویسی خاص ارائه شده باشد،قابل تعمیم C مانند بوده و با سایر زبان‌های برنامه نویسی نیز می‌توان این الگوریتم‌ها و برنامه‌ها را پیاده سازی کرد. لذا در مبحث ساختمان داده‌ها نمادها و دستورات گرامری مورد استفاده در یک زبان خاص مانند C مورد توجه نمی باشند. برای فهم بهتر، می‌توان الگوریتم را شبیه به یک برنامه‌ی کامپیوتری نوشته شده دانست که این برنامه می‌تواند پایان پذیر نباشد و از زمان روشن شدن کامپیوتر تا زمان خاموش شدن آن در یک سیکل قرار گیرد و مدام تکرار شود اما یک الگوریتم حتماً پایان‌پذیر است.

داشتن ورودی: یک الگوریتم می‌تواند ورودی نداشته باشد یا چندین ورودی داشته باشد.

داشتن خروجی: یک الگوریتم باید حداقل یک کمیت به عنوان خروجی داشته باشد.

داشتن قطعیت: هر دستور باید کاملاً واضح باشد.

داشتن محدودیت: یعنی حتماً خاتمه پذیر باشد.

داشتن کارایی: باید بتوان کارایی و نحوه ی کارکرد الگوریتم را با استفاده از کاغذ و قلم امتحان کرد یعنی هر دستورالعمل انجام پذیر باشد.

در این بخش مشخصات الگوریتم‌ها را از دیدگاه زمانی مورد تجزیه و تحلیل قرار می‌دهیم. برای حل مسائل مختلف، الگوریتم‌هایی متفاوت وجود دارند. طراحی یک الگوریتم کارآمد در استفاده بهینه و توسعه سیستم‌های کامپیوتری، نقش مهمی ایفا می‌کند. الگوریتم، در واقع مجموعه ای از دستورالعمل‌هاست که اگر دنبال یا اجرا شود، هدف خاصی را ایفا می‌کند. در بررسی یک الگوریتم به لحاظ ساختمان داده‌ای به دو فاکتور مهم بسیار توجه می‌شود، یکی میزان حافظه ای که الگوریتم مورد نظر اشغال می‌کند و دیگری میزان زمان مصرفی. بدیهی است الگوریتمی بهتر است که فضا و زمان کمتری نیاز داشته باشد. در اینجا بیشتر بازدهی الگوریتم‌ها برحسب زمان مورد بررسی قرار می‌گیرد. در تحلیل میزان زمان مصرفی یک الگوریتم روش‌های مختلفی وجود دارد که در ذیل به دو نوع مهمترین آنها یعنی تعیین تعداد گام‌ها و پیچیدگی زمانی الگوریتم.

**مرتبه و پیچیدگی زمانی**

در میان الگوریتم‌ها، الگوریتم‌هایی از کارایی بهتری برخوردار هستند که از ۲ نظر کارا باشند:

الف) حافظه کمتری نیاز داشته باشند.

ب) زمان اجرای کمتری داشته باشند، یعنی در زمان کمتری نتیجه دلخواه را به ما ارائه کنند.

محاسبه‌ی تعداد گامها و تعداد مراحل یک الگوریتم مفهومی مشابه با پیچیدگی زمانی اما کمی متفاوت‌تر از آن است. در محاسبه‌ی تعداد گامهای یک الگوریتم به هر کدام از دستورات برنامه که توسط CPU مورد محاسبه قرار می‌گیرد، ۱ واحد زمانی اختصاص می‌یابد. هر کدام از این واحدهای زمانی با گام‌های اجرای تمام دستورات یک برنامه، تعداد گام‌های یک برنامه به نکات زیر باید توجه شود:

۱) عبارت توضیحی (comments) :

در هنگام برنامه نویسی یا نوشتن الگوریتم گاهی اوقات یک برنامه‌نویس جهت فهم بهتر یک خط برنامه، یک عبارت توضیحی را در مقابل خط مورد نظر می‌نویسد که به آن comment می‌گویند. Comment ها ارزش اجرایی ندارند و تعداد گام‌های آنها صفر است.

۲) عبارت تعیین نوع (Declarative statements) :

در برنامه نویسی تمامی متغیرهایی که در طول برنامه استفاده می‌شوند، باید در ابتدای برنامه تعیین شوند. برای مثال باید نوع متغیر نوشته شود که نشان می‌دهد این متغییر از نوع اعداد صحیح است یا کاراکتر و یا .... است. این عبارت نیز چون ارزش اجرایی ندارند تعداد گامهای آن صفر است.

۳) عبارت و دستورات انتساب ( Expressions & assignment statements):

منظور از این بخش هنگامی است که مقدار یک متغییر را تعیین می‌شود برای مثال $x = ۵$ . تعداد گام این دستور برابر با یک است.

۴) عبارت تکرار (Iteration statement) :

دستوراتی که باعث تکرار یک عمل می‌شوند مانند for و while که در درس برنامه نویسی با آن آشنا شدید در این قسمت قرار می‌گیرند. در این دستورات شمارش گام فقط برای قسمت کنترل در نظر گرفته می‌شود.

**انواع حلقه‌های تکرار:**

الف) حلقه‌هایی که ابتدا شرط بررسی می‌شود و سپس دستورات اجرا می‌شوند. تعداد تکرار قسمت کنترل $k+۱$، است و تعداد تکرار دستورات k بار است. (مانند حلقه ی For یا While)

ب) حلقه‌هایی که ابتدا دستورات اجرا می‌شوند و سپس شرط حلقه چک می‌شود. تعداد تکرار قسمت کنترل و دستورات هر دو k بار است. (مانند حلقه ی Do While)

۵) عبارت نام تابع: تعداد گام صفر است.
۶) عبارت { }: تعداد گام صفر است.
۷) عبارت return : تعداد گام یک است.

**مثال ۱:** تعداد گام‌های برنامه‌ی زیر را بنویسید.

پاسخ:

۱) $int\ test\ (int\, n)$
۲) {
۳) $int\quad i,j,a=۵,\ b=۱۰\ ,\ c=a+b;$
٤) $for\ (i=۱\ ;\ i\ <=n\ ;\ j++)$
۵) $for\ (\ j=۱\ ;\ j<=n\ ,j++)$
٦) {
۷) $a+=۱;$
۸) $b+=۱;$
۹) $c+=a+b;$
۱۰) }
۱۱) $return\ c;$
۱۲) }

خط شماره ۱: این خط تابع test را معرفی می‌کند و نوع و تعداد خروجی و ورودی تابع را نشان می‌دهد. تعداد گامهای خط معرفی کننده تابع که به آن decleration می گویند برابر با صفر است.

خط شماره ۲: این خط علامت آکولاد باز است. تعداد گام این خط برابر صفر می‌باشد.

خط شماره ۳: توجه شود که در این خط، ۲ کار انجام شده است. بعضی از متغیرها تنها نوع آنها تعریف شده است، مثل $i$ و $j$ بنابراین تعداد گامهای این بخش برابر صفر است چون فقط نوع متغیر مشخص شده است اما متغیر $a$ و $b$ و $c$ علاوه بر تعیین نوع مقدار دهی نیز شده اند. مقدار $a$ برابر با ۵ قرار داده شده است بنابراین گام این مقداردهی ۱ است. مقدار $b$ برابر ۱۰ قرار داده شده است پس گام این مقداردهی ۱ است و مقدار آن برابر با $a+b$ قرار داده شده است بنابراین گام این مقداردهی نیز ۱ است. در نتیجه تعداد کل گامهای این خط برابر ۳ می‌باشد.

خط شماره ۴: توجه شود که این خط یک حلقه ی تکرار وجود دارد که شرط آن در ابتدای حلقه چک می‌شود و شمارنده‌ی حلقه آن از شماره ۱ تا $n$ را می‌شمارد. طبق توضیحات گفته شده در مورد عبارات تکرار، تعداد گامهای این حلقه تکرار برابر $n+۱$ با می‌باشد.

خط شماره ۵: این خط نیز یک حلقه ی تکرار است که دقیقاً شبیه خط شماره ی ۴ می‌باشد بنابراین

$n+1$ بار شرط این حلقه بررسی می‌شود. اما، دقت فرمایید که این حلقه یک حلقه‌ی مستقل نیست چون درون یک حلقه‌ی بیرونی قرار گرفته است. قبلاً در مورد عبارات تکرار توضیح داده شد که اگر شرط حلقه ابتدای حلقه قرار گرفته باشد گام خود حلقه $n+1$ است اما گام دستورات داخلی آن $n$ بار است. در اینجا نیز گام خط شماره‌ی ۴ ، $n+1$ است و دستورات داخلی خط ۴ (یعنی خط ۵) $n$ بار به ازاء خط ۴ تکرار می‌شود.

طبق توضیحات گفته شده گام خود خط ۵ به تنهایی $n+1$ است. گام دستورات داخلی خط ۴، $n$ مرتبه است. بنابراین در کل تعداد گامهای خط ۵ $n(n+1)$ می‌باشد.

خط شماره ۶: قبلاً گفته شد که { } دارای گام صفر می‌باشد.

خط شماره ۷: این خط یک دستور مقدار دهی به متغیر $a$ است که به تنهایی دارای گام ۱ است. اما توجه شود که این دستور، یک دستور داخلی برای حلقه‌ی تکرار خط ۵ است بنابراین به از ۱ ، خط ۵، $n\times 1$ مرتبه تکرار می‌شود. اما فراموش نشود که خود حلقه ی تکرار خط ۵ نیز حلقه‌ی داخلی برای خط ۴ است و به ازاء خط ۴، n مرتبه تکرار می‌شود.

پس در کل خط شماره ۷ دارای تعداد گامهای $n\times n\times 1$ یعنی $n^2$ می‌باشد.

خط شماره ۸: توضیحات کاملاً شبیه به خط شماره ۷ می‌باشد و $n^2$، تعداد گامهای این خط است.

خط شماره ۹: توضیحات کاملاً شبیه به خط شماره ۷ می‌باشد و $n^2$ تعدادگامهای این خط است.

خط شماره ۱۰: یک آکولاد بسته است که تعداد گامهای آن برابر صفر می‌باشد. مانند خط ۲ و ۶ که آکولادهای باز هستند.

خط شماره ۱۱: یک دستور return است که کاملاً مستقل می‌باشد بنابراین تعداد گامهای آن برابر با ۱ است.

خط شماره ۱۲: توضیحات شبیه خط شماره ۱۰، می‌باشد و تعداد گام آن برابر صفر است.

حال تعداد گامهای هر خط را محاسبه شد. کافی است که تک تک آنها را با هم جمع شوند تا در نهایت تعداد گام‌های برنامه را بدست آید.

تعداد گامهای برنامه = $0+0+3+(n+1)+(n\times(n+1))+0+0+n^2+n^2+n^2+0+1+0$

تعداد گامهای برنامه = $3+n+1+n^2+n+n^2+n^2+n^2+1=4n^2+2n+5$

**تحلیل پیچیدگی زمانی**

پیچیدگی زمانی چیست؟ پیچیدگی زمانی به این معناست که چه زمانی طول می‌کشد که یک الگوریتم به ازای یک مقدار مشخص اجرا شود.

در محاسبه‌ی پیچیدگی زمانی یک الگوریتم، زمان کل الگوریتم مورد نظر، برحسب زمان کلی آن که اغلب به میزان ورودی‌ها بستگی دارد، محاسبه می‌شود و تک تک دستورات الگوریتم یا به عبارت دیگر تعداد گام‌های برنامه شمارش نمی‌شوند. چرا که تعداد دستورات اغلب به نوع زبان برنامه نویسی و نحوه نوشتن برنامه بستگی دارد. محاسبه‌ی پیچیدگی زمانی یک الگوریتم، بر اساس تعداد دفعات اجرای یک عمل اصلی که آن را عمل مبنایی نیز می‌نامند، انجام می‌شود. لذا به طور کلی زمان اجرای یک الگوریتم اغلب به میزا ورودی $(n)$ آن الگوریتم بستگی دارد و با افزایش تکرار عمل اصلی ، افزایش می‌یابد. عمل اصلی می‌تواند یک یا گروهی از دستورات باشد. توجه داشته باشید که در محاسبه‌ی پیچیدگی زمانی یک الگوریتم زمان اجرای تک تک دستورات با گام‌های برنامه مدنظر نیست، بلکه ملاک، محاسبه تعداد دفعاتی است که عبارت اصلی اجرا می‌شود. در حالیکه در محاسبه تعداد گامهای یک برنامه، اجرای هر کدام از دستورات، یک گام محسوب می‌شود و مجموع کل دفعاتی که تمام دستورات یک برنامه اجرای می‌شوند، تعداد گامها با تعداد کل مراحل آن برنامه را تشکیل می‌دهند.

**نکته۱:** در تحلیل پیچیدگی زمانی در واقع تجزیه و تحلیل کارایی الگوریتم به عنوان تابعی از اندازه ورودی، با تعیین تعداد دفعاتی است که عمل با عملیات اصلی صورت می‌گیرد. این روش یکی از رویکردهای استاندارد تحلیل سیستم است. در حالیکه بررسی گامهای یک برنامه، به طبیعت آن برنامه باز می‌گردد و در واقع زمان دقیق اجرای آن برنامه را تعیین می‌کند.

برای مثال اگر قرار باشد تنها ۱ مرتبه در الگوریتم ما مقدار $a+b$ شود پیچیدگی زمانی آن ۱ در نظر گرفته می‌شود اما اگر گفته شود $n$ مرتبه $a+b$ شود، حال پیچیدگی زمانی آن بستگی به $n$ دارد و برابر با $n$ می‌شود. بنابراین در نظر داشته باشید که:

در تحلیل پیچیدگی زمانی ۳ فاکتور بسیار مهم است.

الف) تعداد ورودی‌ها یا خروجی‌ها یا ترکیب آنها که نشان می‌دهد عمل مورد نظر چند مرتبه تکرار می‌شود.

ب) عملی که در الگوریتم مورد نظر بیشترین زمان را برای اجرا صرف خود می‌کند.

ج) رابطه ای که بین مورد الف و ب وجود دارد که آن را می‌توان به صورت یک تابع $T(n)$ نوشت.

(در مثال قبل که تعداد گامهای دقیق برنامه را بدست آورده شد می توان گفت که $T(n)$ برای آن برنامه برابر است با $T(n)=۴n^۲+۲n+۵$)

در تحلیل پیچیدگی زمانی ما با ۳ مفهوم سر و کار داریم که اگر آنها به دقت مطالعه شود تحلیل تمامی الگوریتم ها را می توان با هر کدام از این ۳ مفهوم بیان کرد.