عنوان مقاله: الگوریتم برنامه نویسی چیست؟

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: ۲ مهر ۱۴۰۲

منبع: https://nikamooz.com/what-is-a-programming-algorithm

الگوریتم برنامه نویسی و شناخت و درک آن در جهانی که برنامه‌نویسی به‌سرعت دستخوش تغییرات و بروزرسانی است، امری ضروری به‌حساب می‌آید. درواقع الگوریتم‌ها به‌عنوان ستون اصلی کدنویسی کارآمد است که به واسطه آن، نرم‌افزار بادقت و سرعت بهتری اجرا می‌شود. در این مقاله قصد داریم به چیستی الگوریتم در برنامه نویسی، مزایا و معایب آن و نحوه طبقه‌بندی آن‌ها بپردازیم.

الگوریتم برنامه نویسی

الگوریتم که از اصطلاحات برنامه نویسی به‌شمار می‌رود، به بیان ساده، مجموعه دستورالعمل‌های گام‌به‌گامی هستند که به‌صورت دقیق تعریف شده‌اند تا با کمک آن‌ها بتوانیم یک مسأله خاص را حل کنیم یا یک وظیفه (Task) مشخص را به مرحله اجرا بگذاریم. این الگوریتم‌ها به‌عنوان بلوک‌های ساخته‌شده برای توسعه نرم‌افزار (Software Development) به کار می‌روند و به برنامه‌نویس این امکان را می‌دهند که به‌صورت کارآمد، کدنویسی کنند و به‌دنبال آن، به خروجی مطلوب برسند. به بیان دیگر، الگوریتم برنامه نویسی یک مجموعه دستورالعمل برای کامپیوتر است که در نحوه پردازش داده‌ها و اجرای وظایف کمک‌کننده به‌حساب می‌آید.

ساختار الگوریتم برنامه نویسی

در ادامه، به اجزای اصلی در ساختار الگوریتم برنامه نویسی می‌پردازیم تا با جریان و منطق آن‌ها آشنا شوید.

- **مقداردهی اولیه (Initialization)**

معمولاً در ابتدای آغاز یک الگوریتم در برنامه نویسی، لازم است متغیرها و ساختار داده‌های موردنیاز «مقداردهی اولیه» می‌شوند. این مرحله به‌عنوان تنظیم کردن وضعیت (State) اولیه الگوریتم قلمداد می‌شود.

- **ورودی (Input)**

ممکن است الگوریتم‌ها برای آغاز کار، به داده‌های ورودی نیاز داشته باشد. این داده‌ها ازطریق کاربر، خواندن یک فایل، دریافت ازطریق سنسور و چنین راه‌هایی عملی می‌شوند. توجه کنید که داده‌های ورودی، یکی از ضروری‌ترین اجزای عملکرد الگوریتم محسوب می‌شوند.

- **پردازش**

بخش پردازش به‌عنوان قلب یک الگوریتم برنامه نویسی در نظر گرفته می‌شود و در آن، عملیات و محاسبات اصلی برای حل مسئله یا کامل کردن Task انجام می‌گیرد. معمولاً قسمت پردازش یک الگوریتم، موارد مختلفی مانند گام‌های پشت سر هم، عبارات شرطی (Conditional Statements)، لوپ ها (Loops) و فراخوانی تابع را دربرمی‌گیرد. گام‌های پردازش به‌طور خاص به مسئله‌ای بستگی دارد که قصد داریم آن را حل کنیم.

- **خروجی (Output)**

الگوریتم پس از فرآیند پردازش داده‌های ورودی، خروجی تولید می‌کند. این خروجی می‌تواند حاصل محاسبات، نمایش اطلاعات کاربر، نوشتن داده در یک فایل و هر عمل مرتبط دیگری باشد.

- **نهایی‌سازی یا پاکسازی**

در برخی موارد، لازم است فرآیند پاکسازی (Cleanup) یا نهایی‌سازی (Finalization) وضعیت در الگوریتم برنامه نویسی انجام بگیرد. این عمل می‌توانند مواردی مانند آزادسازی منابع (Resources)، بستن فایل‌ها و از تخصیص درآوردن (Deallocation) حافظه را شامل شود.

- **رسیدگی به خطا**

الگوریتم‌ها می‌توانند مکانیزم‌هایی را داشته باشند که به واسطه آن‌ها، رسیدگی به خطاهای مختلف و شرایط استثنایی انجام می‌شود. این موضوع، مواردی مانند بررسی خطاها در طول ورودی، پردازش و خروج و انجام عملی متناسب با آن را در برمی‌گیرد.

- **تست و اعتبارسنجی**

پیش از آن که الگوریتم برنامه نویسی را در محیط تولید محصول استفاده کنید، لازم است آن را تست (Test) و اعتبارسنجی (Validation) کرده تا از کارایی و صحت کارکرد آن اطمینان داشته باشید. برای این کار، الگوریتم خود را با ورودی‌های مختلف در سناریوهای متفاوت تست، یا همان آزمایش، می‌کنید تا مطمئن شوید الگوریتم، آن گونه که شما می‌خواهید رفتار می‌کند.

- **مستندات و نظرها**

شما باید در طول کدنویسی الگوریتم در برنامه نویسی، کامنت‌ها و مستندسازی (Documentation) خود را قرار دهید تا با کمک آن‌ها اهداف کدهای شما، نحوه کارکرد آن الگوریتم و سایر موارد مهم شرح داده شوند. با این کار، درک، فهمیدن و همچنین نگهداری (Maintenance) الگوریتم بهبود پیدا خواهد کرد.

- **تجزیه و تحلیل پیچیدگی الگوریتم**

زمانی که الگوریتم برنامه نویسی پیچیده می‌شود، تجزیه و تحلیل دقیق در پیچیدگی زمانی و فضایی (Time and Space Complexity) آن امری ضروری است و می‌تواند شما را در درک بهتر در مقیاس پذیری (Scalability) و ویژگی‌های اجرایی آن کمک کند.

- **بهینه‌سازی**

شما باید براساس نیازهای اجرایی و پیچیدگی الگوریتم، آن را بهینه‌سازی (Optimization) کنید. با کمک بهینه‌سازی، زمان اجرای الگوریتم یا مقدار حافظه مصرفی آن کاهش می‌یابد.

در ادامه بررسی الگوریتم برنامه نویسی، قصد داریم اصلی‌ترین مشخصه‌های یک الگوریتم را شرح دهیم تا بهتر با آن‌ها آشنا شوید.

مشخصه های الگوریتم در برنامه نویسی

با توجه به این که الگوریتم‌ها با هدف حل مسائل یا اجرای وظایف مشخصی ایجاد شده‌اند، بنابراین مشخصه‌هایی دارند که دانستن آن‌ها خالی از لطف نیست. در ادامه، مشخصه‌های الگوریتم برنامه نویسی را مورد بررسی قرار می‌دهیم.

۱. **شفافیت و عدم وجود ابهام**

صریح بودن و شفافیت دستورالعمل باعث می‌شود تا نیاز به تفسیر بیش از حد آن‌ها نباشد. درواقع، دستورالعمل الگوریتم‌ها باید به گونه‌ای باشد که در آن، مراحل لازم برای حل مسئله به‌صورت دقیق ذکر شده باشند.

۲. **متناهی یا کران‌دار**

لازم است الگوریتم برنامه نویسی پس از گام‌های محدود (Finite) پایان بیابد. درواقع، ما به الگوریتمی نیاز داریم که پاسخ مورد نیازمان را در یک بازه زمانی عقلانی به ما ارائه دهد و تا بی‌نهایت ادامه‌دار نباشد.

۳. **کارایی (Effectiveness)**

الگوریتم‌ها باید به‌صورت کارآمد عمل کنند. این یعنی هر گام آن قابل اجرا باشد و قابلیت دستیابی به آن در یک زمان مشخص وجود داشته باشد. به‌واسطه کارآمد بودن یک الگوریتم، این اطمینان حاصل می‌شود که آیا امکان پیاده‌سازی آن با یک کامپیوتر وجود دارد یا خیر.

٤. **قطعیت (Determinism)**

الگوریتم برنامه نویسی باید قطعی باشد؛ یعنی هربار که یک ورودی مشخصی را دریافت می‌کند، یک خروجی یکسان را تولید کند. به این ترتیب، می‌توانیم به کارکرد آن اطمینان داشته باشیم و به‌راحتی به پیش‌بینی عملکرد آن بپردازیم.

۵. **یکتایی (Uniqueness)**

ممکن است یک مسئله خاص، ازطریق الگوریتم‌های مختلفی قابل حل باشد. در چنین شرایطی، لازم است تمام این الگوریتم‌ها خروجی یکسان و صحیح را برای ورودی یکسان تولید کنند. البته این الگوریتم‌ها می‌توانند از لحاظ کارایی با یکدیگر تفاوت داشته باشند، اما خروجی آن‌ها «باید» یکسان باشد.

٦. **ورودی و خروجی**

الگوریتم برنامه نویسی داده‌های ورودی را دریافت کرده و خروجی تولید می‌کند تا ازطریق دنباله‌ای از گام‌های خاص، ورودی‌ها را به نتیجه دلخواه خود تبدیل کند.

۷. **صحت (Correctness)**

لازم است الگوریتم در برنامه نویسی دقیقاً مسئله‌ای را حل کند که برای آن طراحی شده است. درواقع، الگوریتم باید خروجی‌های مورد انتظار برای تمام ورودی‌های قابل قبول را به‌درستی تولید کند.

۸. **صراحت و درک آسان**

اگر طراحی الگوریتم‌ها به خوبی انجام شده باشد، درک و فهمیدن آن تسهیل می‌یابد. برای افزایش سادگی و قابل درک بودن الگوریتم، لازم است ساختارهای منطقی و نام‌گذاری متغیرها به‌درستی و معنادار انجام شود تا خوانایی آن برای سایر برنامه‌نویسان افزایش پیدا کند.

۹. **بهره‌وری (Efficiency)**

با وجود اینکه نمی‌توان این مشخصه الگوریتم برنامه نویسی را به‌صورت کاملاً دقیق و مشخص بیان کرد، اما معمولاً حالت ایده‌آل یک الگوریتم سودمند این است که بتواند تسک را با سرعت مطلوب و حداقل استفاده از منابع، تکمیل کند.

۱۰. **ماژولاریتی (Modularity)**

می‌توان الگوریتم‌ها را به توابع یا اجزای (Components) کوچک‌تر با قابلیت استفاده مجدد (Reusable) تبدیل کرد تا بتوان قابلیت نگهداری‌پذیری (Maintainability) و امکان استفاده مجدد کدها را افزایش داد.

۱۱. **سازگاری (Adaptability)**

برخی از الگوریتم‌ها به‌صورت سازگار و وقف‌پذیر برای انواع مسائل و مجموعه داده‌ها (Dataset) قابل استفاده هستند و برخی دیگر، به‌طور خاص برای سناریوهای مشخصی کاربرد دارند.

۱۲. **بهینه‌سازی (Optimization)**

در برخی موارد، طراحی الگوریتم برنامه نویسی با هدف بهینه‌سازی معیارهای مشخصی مانند کاهش پیچیدگی زمانی، کاهش مقدار حافظه مورداستفاده و سایر محدودیت‌های منابع انجام می‌شود.

۱۳. **مقیاس‌پذیری (Scalability)**

الگوریتم‌ها باید به‌صورتی طراحی شده باشد که بدون افت در کارایی، امکان رسیدگی به حجم بالایی از ورودی‌ها را داشته باشند و بتواند به‌خوبی، مقیاس‌گذاری شود.

۱۴. **موازی‌شدنی (Parallelizable)**

با پیشرفت چشمگیر در پردازشگرهای چندهسته‌ای (Multicore Processors) و سیستم های توزیع شده (Distributed Systems)، الگوریتمی می‌تواند کارایی بهتری از خود به جا بگذارد که امکان موازی‌سازی آن وجود داشته باشد.

۱۵. **قدرت و استحکام**

طراحی الگوریتم برنامه نویسی باید به گونه‌ای انجام شود که بتواند به‌درستی به ورودی‌های حاوی خطا یا غیرقابل انتظار رسیدگی کند و تا حد ممکن، از مواردی مانند از کار افتادگی کامل سیستم و نتایج نادرست بپرهیزد.

مشخصه‌های فوق این اطمینان را ایجاد می‌کنند که الگوریتم‌ها به‌عنوان یک ابزار قابل اکتفا برای حل مسائل حوزه برنامه‌نویسی و علوم کامپیوتر به کار رود.

مثال الگوریتم در برنامه نویسی

فرض کنید می‌خواهیم الگوریتم جمع دو عدد را بنویسیم.

- **ورودی‌ها:** عدد اول و عدد دوم
- **خروجی:** حاصل جمع عدد اول و عدد دوم

گام‌های موردنیاز عبارتند از:

- خواندن مقدار عدد اول و عدد دوم
- جمع کردن عدد اول و عدد دوم
- ذخیره‌سازی حاصل جمع این دو عدد در یک متغیر جدید به نامSum
- نمایش مقدار متغیر Sum در خروجی

در ادامه، کدهای پایتون این مثال قرار دارند:

```
# Step 1: Read input
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Step 2: Calculate the sum
sum = num1 + num2

# Step 3: Display the result
print(f"The sum of {num1} and {num2} is {sum}")
```

این الگوریتم برنامه نویسی ساده برای اجرای عملیات ریاضی طراحی شده است و یک مثال ساده ولی قابل درک از نحوه استفاده الگوریتم محسوب می‌شود.

مزایای الگوریتم برنامه نویسی

در این بخش، اصلی‌ترین فواید الگوریتم برنامه نویسی را بررسی می‌کنیم:

- افزایش کارایی نرم‌افزار و کاهش زمان اجرا (Execution Time)
- امکان استفاده مجدد از الگوریتم و افزایش ماژولاریتی کد
- افزایش قابل نگهداری بودن کد
- مقیاس‌پذیری و توانایی رسیدگی به داده‌ها در حجم گسترده
- افزایش ثبات و کاهش احتمال بروز خطاهای غیرقابل انتظار
- تجزیه مسائل پیچیده به گام‌های قابل مدیریت و کوچک شده
- قابلیت بهینه‌سازی برمبنای سناریوهای خاص یا محدودیت‌های مشخص
- امکان پیاده‌سازی الگوریتم با **انواع زبان های برنامه نویسی** و افزایش قابل حمل بودن کد
- افزایش توانایی افراد در همکاری با یکدیگر و وجود مستندات ارزشمند
- امکان ایجاد تست به منظور بررسی نحوه عملکرد الگوریتم
- افزایش مهارت حل مسئله

به‌طور کلی، الگوریتم برنامه نویسی ابزارهای ضروری و مهمی در برنامه‌نویسی محسوب می‌شوند که در بهبود مواردی همچون کارایی، کیفیت کدها، قابل نگهداری بودن، مقیاس‌پذیری و قابلیت‌های حل مسئله نقش به‌سزایی دارند. هرچند که استفاده از الگوریتم‌ها مزیت‌های بی‌شماری دارد، اما می‌تواند گاهی با چالش همراه باشد. در ادامه، به آن‌ها می‌پردازیم.

معایب الگوریتم برنامه نویسی

حال می‌خواهیم به کاستی‌های الگوریتم برنامه نویسی اشاره کنیم:

- طراحی و پیاده‌سازی الگوریتم برای مسائل دشواری که با چالش همراه است.
- برخی از الگوریتم‌ها به منابع محاسباتی گسترده نیاز دارند.
- با گذر زمان و توسعه محیط‌های نرم‌افزاری، برخی الگوریتم‌ها به به‌روزرسانی و نگهداری نیاز دارند.

اکنون قصد داریم پرکاربردترین الگوریتم ها در برنامه نویسی را معرفی کنیم.

الگوریتم های برنامه نویسی پرکاربرد

زمانی که درمورد «پرکاربردترین» الگوریتم های برنامه نویسی صحبت می‌کنیم، باید توجه کنیم که هرکدام از آن‌ها برای مسئله خاصی کاربرد دارند؛ بنابراین، انتخاب از میان آن‌ها به نوع مسئله‌ای بستگی دارد که قصد حل آن را داریم. با این دیدگاه، در ادامه، الگوریتم‌هایی را معرفی می‌کنیم که به‌طور گسترده کاربردی هستند و به‌عنوان بخش‌های بنیادی علوم کامپیوتر محسوب می‌شوند.

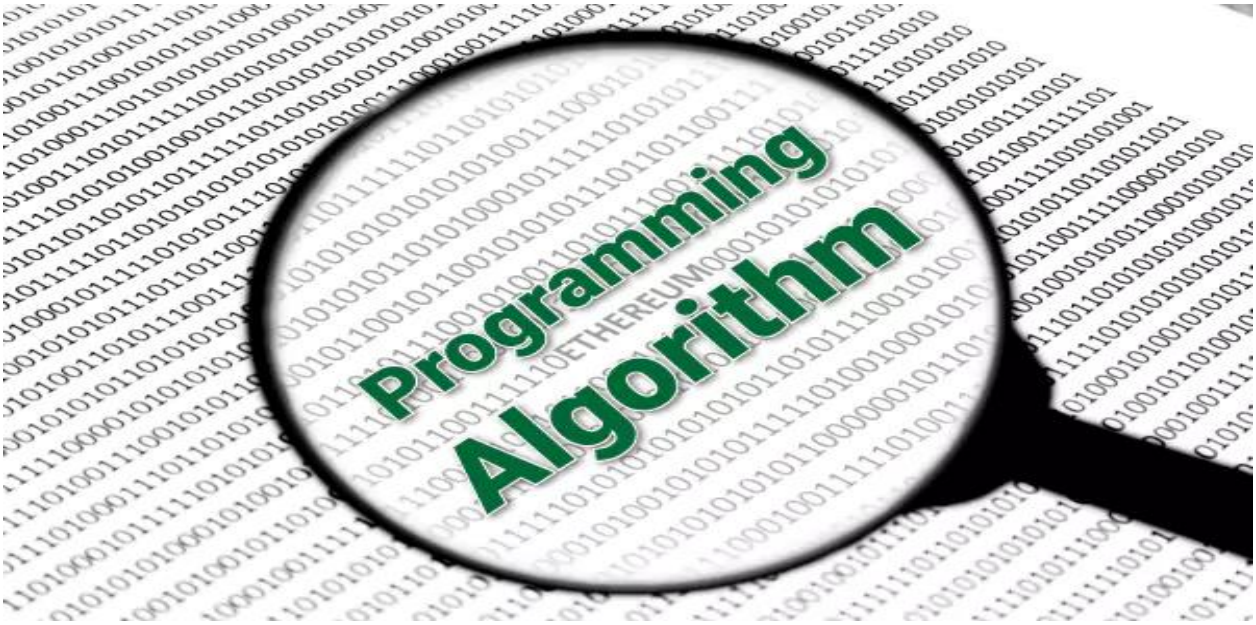
- الگوریتم‌های مرتب‌سازی (Sorting)
- الگوریتم‌های جستجو (Binary)
- الگوریتم‌های گرافی (Graph)
- الگوریتم‌های برنامه‌نویسی پویا (Dynamic Programming)
- ساختمان داده‌ها (Data Structures)
- الگوریتم‌های هوش مصنوعی (AI) و یادگیری ماشین (Machine Learning)
- الگوریتم‌های رمزنگاری (Cryptographic)
- الگوریتم‌های شبکه (Networking) و سیستم عامل (OS)

آموزش نوشتن الگوریتم در برنامه نویسی

نوشتن الگوریتم یک فرآیند ساختارمند و گام‌به‌گام است که برای حل یک مسئله مشخص یا اجرای یک تسک به کار می‌رود. در ادامه، به گام‌های لازم و اساسی برای نوشتن الگوریتم برنامه نویسی اشاره خواهیم کرد:

۱. درک دقیق مسئله
۲. برنامه‌ریزی برای رویکرد خود
۳. تجزیه مسئله به اجزای کوچک‌تر و قابل مدیریت
٤. مشخص کردن ورودی‌ها و خروجی‌های مسئله
٥. نوشتن الگوریتم (شامل مقداردهی اولیه متغیرها، نامگذاری توصیفی، استفاده از ساختارهای کنترلی، رسیدگی به خطاها و استثناهای مسئله، تست و تکرار)
٦. بهینه‌سازی (Optimization)
۷. مستندسازی
۸. بررسی و اصلاح
۹. پیاده‌سازی (Implementation)
۱۰. تست کامل الگوریتم پیاده‌سازی شده
۱۱. تکرار (Iteration)
۱۲. مستندسازی مجدد

توجه کنید که طراحی الگوریتم برنامه نویسی باید به گونه‌ای انجام شود که در آن، خلاقیت و در عین حال، دقت وجود داشته باشد. بسیار مهم است که یک حد توازن میان درک و شفافیت کدها و حل مسئله به‌صورت کارآمد برقرار شود. ازسوی دیگر، توجه کنید که الگوریتم شما در چه حوزه‌ای استفاده خواهد شد و اساساً چه نیازها یا محدودیت‌هایی روی دامنه مسئله‌تان اعمال شده است.



انواع دسته بندی های الگوریتم

به‌طور کلی، می‌توانیم انواع الگوریتم برنامه نویسی را براساس کارایی و ویژگی‌های آن‌ها به‌صورت زیر طبقه‌بندی کنیم:

الگوریتم های مرتب سازی (Sort)

الگوریتم‌های مرتب‌سازی برای چیدن و مرتب‌سازی داده‌ها به ترتیب خاصی (صعودی یا نزولی) استفاده می‌شوند. الگوریتم‌هایی مانند الگوریتم مرتب‌سازی حبابی (Bubble Sort)، مرتب‌سازی ادغامی (Merge Sort) و مرتب‌سازی سریع (Quick Sort) همگی از این طبقه‌بندی هستند.

الگوریتم های جستجو (Search)

الگوریتم برنامه نویسی جستجو برای پیدا کردن یک آیتم خاص درون مجموعه داده استفاده می‌شوند. الگوریتم جستجوی دودویی (Binary) و جستجوی خطی (Linear) دو نوع مشهور از این دسته از الگوریتم‌ها هستند.

الگوریتم های حریصانه (Greedy)

الگوریتم‌های حریصانه (Greedy) در هر گام از اجرا، راه حل‌هایی برای **بهینه محلی** (Local optimum) را با هدف رسیدن به **بهینه سراسری** (global) انتخاب می‌کنند. معمولاً از الگوریتم‌های حریصانه برای مسائلی مانند یافتن کوتاه‌ترین مسیر در یک گراف استفاده می‌شود.

برنامه نویسی پویا (Dynamic Programming)

برنامه‌نویسی پویا روش حل مسئله‌ای است که در آن، مسئله به زیربخش‌های کوچک‌تر تقسیم می‌شود تا بدین طریق، از محاسبات تکراری جلوگیری شود.

تقسیم و غلبه (Divide and Conquer)

این رویکرد، مسئله را به زیرمسائل کوچک‌تر تقسیم و هرکدام را به‌طور مستقل حل می‌کند و درنهایت، پاسخ آن‌ها را با یکدیگر ادغام می‌کند تا مسئله اصلی را حل کند. الگوریتم ادغام (Merge) معروف‌ترین نوع الگوریتم های برنامه نویسی است.

عقبگرد (Backtracking)

الگوریتم عقبگرد با بررسی انتخاب‌های مختلف مسئله، سعی می‌کند جواب‌های مسئله را به‌طور افزایشی (Incremental) پیدا کند و مسیرهایی که به بن‌بست می‌رسد را برگردد و ادامه ندهد. در اغلب موارد، این روش در حل سودوکو (Sudoku) کاربرد دارد.

جمع بندی

در این مطلب، به شرح کاملی از اهمیت طراحی الگوریتم برنامه نویسی در حوزه کامپیوتر پرداختیم و فواید و کاستی‌های آن را بررسی کردیم. استفاده از الگوریتم می‌تواند به‌عنوان یک ابزار کمکی به افزایش کارایی و کیفیت کدهای شما، کاهش احتمال بروز خطاهای غیرقابل انتظار و سایر موارد مهم دیگر منجر شود. بنابراین، پیشنهاد می‌شود با فراگیری نحوه نوشتن این الگوریتم‌ها و مفاهیم پایه مربوط به آن، مهارت‌های کدنویسی خود را افزایش دهید.