

جزوه تفکر الگوریتمی

الگوریتم اصلاً چیست و به چه دردی می‌خورد؟

مقدمه: آیا شما یک برنامه‌نویس به دنیا آمده‌اید؟!

شاید تعجب کنید، اما حقیقت این است که شما سال‌هاست دارید مثل یک کامپیوتر فکر می‌کنید، بدون اینکه حتی یک خط کد نوشته باشید!

صبح که از خواب بیدار می‌شوید تا به مدرسه بروید:

1. **آلارم زنگ می‌زند.**
2. **پتو را کنار می‌زنید.**
3. **مسواک می‌زنید و دست و صورتتان را می‌شویید.**
4. **صبحانه می‌خورید.**
5. **کیف و کتاب‌ها را برمی‌دارید و راهی می‌شوید.**

آیا می‌توانید مرحله ۵ را قبل از مرحله ۱ انجام دهید؟ یعنی در حالی که هنوز خوابید، ناگهان وسط خیابان در حال رفتن به مدرسه باشید؟! معلوم است که نه!

این ترتیب منظم و منطقیِ کارها، دقیقاً همان چیزی است که به آن می‌گوییم: الگوریتم.

تعریف ساده الگوریتم:

الگوریتم یعنی :یک دستورالعمل مشخص، قدم‌به‌قدم و بدون ابهام برای رسیدن از یک «نقطه شروع» به یک «نتیجه مشخص»..

نام الگوریتم از نام دانشمند بزرگ ایرانی، **«محمد بن موسی خوارزمی»** گرفته شده که نخستین بار روش‌های مرحله‌به‌مرحله را برای حل مسائل ریاضی ابداع کرد.

اگر بخواهیم به زبان کامپیوتر بگوییم:

شما به برنامه ورودی (Input) می‌دهید ←← الگوریتم آن را پردازش می‌کند ←← به شما خروجی (Output) تحویل می‌دهد.

مثال واقعی از الگوریتم در برنامه‌هایی که هر روز استفاده می‌کنید:

۱. الگوریتم ورود به حساب کاربری (Login):

وقتی می‌خواهید وارد اکانت شاد، روبیکا، بازی یا ایمیل بشوید:

مرحله ۱: کاربر شماره همراه/ایمیل و رمز عبور را وارد می‌کند.

مرحله ۲: الگوریتم چک می‌کند: آیا رمز وارد شده دقیقاً با رمزی که قبلاً در سیستم ثبت شده یکی است؟

اگر بله: در باز می‌شود و وارد صفحه اصلی می‌شوید.

اگر نه: پیام خطا نمایش داده می‌شود: «رمز اشتباه است، دوباره امتحان کنید».

مرحله ۳: اگر کاربر ۳ بار متوالی اشتباه زد، اکانت را برای ۱ دقیقه قفل کن!

(می‌بینید؟ این دقیقاً یک الگوریتم امنیتی چند مرحله‌ای است).

الگوریتم باختن در بازی (Game Over):

در بازی‌های مثل ماریو، یا بازی‌های اکشن:

مرحله ۱: موقعیت بازیکن روی صفحه را لحظه‌به‌لحظه رصد کن.

مرحله ۲: بررسی کن: آیا بدن کاراکتر به مانع یا تیر دشمن برخورد کرد؟

اگر نه: بازی ادامه دارد و امتیاز اضافه کن.

اگر بله: یکی از جان‌های بازیکن را کم کن.

مرحله ۳: بررسی کن: آیا تعداد جان‌ها به «صفر» رسید؟

اگر بله: صفحه Game Over را نشان بده و صدا را قطع کن.

الگوریتم مسیریاب (نشان / بلد / Google Maps):

- ورودی: مبدا: خانه / مقصد: مدرسه.
- مرحله ۱: تمام مسیرها و کوچه‌های ممکن بین خانه تا مدرسه را لیست کن.
- مرحله ۲: ترافیک هر مسیر را بررسی کن.
- مرحله ۳: مسافت هر مسیر را تقسیم بر سرعت مجاز کن تا «زمان رسیدن» حساب شود.
- مرحله ۴: مسیری که کمترین زمان را دارد روی نقشه به رنگ سبز یا آبی به راننده نشان بده.

ویژگی که هر الگوریتم کامپیوتری باید داشته باشد:

دقت بالا (عدم ابهام): به کامپیوتر نمی‌شود گفت «برو جلو تا به یک جای خوب برسی!». باید دقیقاً بگویید «۱۰ قدم به جلو برو؛ اگر مانع دیدی، ۹۰ درجه به چپ بچرخ».

ترتیب مشخص: در برنامه‌نویسی اگر ترتیب خطوط کد را جابه‌جا بنویسید، بازی خراب می‌شود (مثلاً اول باید چک کند کاربر رمز دارد یا نه، بعد صفحه چت‌ها را باز کند؛ نه برعکس!).

پایان داشتن: یک برنامه اگر پایان یا هدف مشخصی برای یک کار نداشته باشد، گوشی هنگ می‌کند یا داغ می‌شود!

بخش چالش عملی: مثل یک طراح بازی فکر کن!

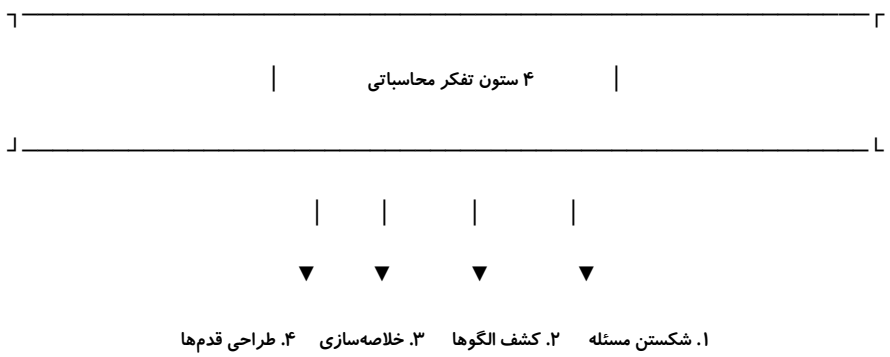
فرض کن می‌خواهی برای یک بازی دوز یا بازی سنگ‌کاغذقیچی ساده الگوریتم بنویسی:

الگوریتم ساده‌ای بنویس که چک کند چه کسی در بازی سنگ-کاغذ-قیچی برنده شده است؟

چهار ستون تفکر محاسباتی (چطور مثل یک برنامه‌نویس فکر کنیم؟)

مقدمه: آیا برنامه‌نویس‌ها نابغه‌اند؟!

- **شاید وقتی یک بازی بزرگ مثل Minecraft یا برنامه‌ای مثل دیجی‌کالا و اسنپ را می‌بینید، پیش خودتان بگویید:**
- **«چطور یک انسان توانسته چنین چیز بزرگ و پیچیده‌ای را بسازد؟ حتماً مغز عجیبی داشته!»**
- **واقعیت این است که برنامه‌نویس‌ها از یک فرمول جادویی استفاده نمی‌کنند؛ آن‌ها از یک مهارت به نام «تفکر محاسباتی» (Computational Thinking) استفاده می‌کنند.**
- **تفکر محاسباتی یعنی تبدیل یک مسئله بزرگ و ترسناک به چند تکه کوچک و قابل حل. این مهارت ۴ ستون اصلی دارد:**



بیایید این ۴ ستون را با یک مثال واقعی از دنیای برنامه‌ها بررسی کنیم:

ستون اول: تجزیه یا شکستن مسئله (Decomposition)

تعریف: یعنی یک پروژه بزرگ را تکه‌تکه کنی تا بتوانی هر بخش را جداگانه حل کنی.

مثال در برنامه‌نویسی:فرض کنید می‌خواهید یک برنامه شبیه «اسنپ‌فود» بسازید. اگر از همان اول بگویید «می‌خواهم کل اسنپ‌فود را بسازم!»، مغزتان قفل می‌شود! اما با ستون اول، آن را می‌شکنید:

- **بخش ۱: ثبت‌نام و ورود کاربر با شماره موبایل.**
- **بخش ۲: نمایش منوی رستوران‌ها و غذاها.**
- **بخش ۳: سبد خرید (اضافه و کم کردن پیتزا و نوشابه).**
- **بخش ۴: پرداخت اینترنتی هزینه.**
- **بخش ۵: ارسال موقعیت پیک موتوری روی نقشه.**
- **حالا حل کردن هر کدام از این تکه‌ها به تنهایی خیلی ساده‌تر شد!**

ستون دوم: الگوخوانی و کشف شباهت‌ها (Pattern Recognition)

تعریف: پیدا کردن بخش‌های تکراری و مشابه در مسئله، تا مجبور نباشیم چرخ را دوباره اختراع کنیم!

مثال در برنامه‌نویسی:در یک بازی جنگی، ۵۰ مدل اسلحه مختلف وجود دارد (تپانچه، کلاشینکف، تک‌تیرانداز و...). آیا برنامه‌نویس برای تک‌تک آن‌ها از صفر کد جداگانه می‌نویسد؟خیر! او دنبال الگو می‌گردد:

همه اسلحه‌ها:

۱. **تیر شلیک می‌کنند.**
۲. **یک تعداد فشنگ دارند (تعداد خشاب).**
۳. **نیاز به «پر کردن مجدد» (Reload) دارند.**
۴. **برنامه‌نویس یک «الگوی کلی برای تفنگ» می‌سازد و فقط برای هر کدام سرعت تیر و شکل ظاهری‌اش را تغییر می‌دهد.**

ستون سوم: خلاصه‌سازی یا تجرید (Abstraction)

تعریف: دور ریختن جزئیات اضافی و بی‌اهمیت، و چسبیدن به اطلاعاتی که واقعاً لازم داریم.

- **مثال در برنامه‌نویسی:وقتی نقشه «نشان» یا «Google Maps» را باز می‌کنید:**
- **نقشه به شما خیابان‌ها، ترافیک و جهت حرکت را نشان می‌دهد.**
- **اما آیا نقشه رنگ دیوار خانه‌ها، مدل درخت‌های کنار خیابان یا رنگ کفش عابران پیاده را هم نشان می‌دهد؟!**
- **قطعا نه! چون این اطلاعات به راننده برای پیدا کردن مسیر کمکی نمی‌کند و فقط گوشی را سنگین و کند می‌کند.یک برنامه‌نویس حرفه‌ای همیشه جزئیات اضافه را حذف (Abstract) می‌کند.**

ستون چهارم: طراحی الگوریتم (Algorithm Design)

تعریف: حالا که مسئله را خرد کردی، الگوها را یافتی و اضافات را حذف کردی، دستورالعمل نهایی را گام‌به‌گام بنویس تا کامپیوتر اجرا کند.

- **مثال در برنامه‌نویسی:طراحی مرحله‌به‌مرحله فرآیند «خرید از فروشگاه بازی»:**
- **کاربر روی آیتم "پوسته‌ی جدید کاراکتر (Skin)" کلیک می‌کند.**
- **برنامه بررسی می‌کند: آیا سکه کاربر ≥ قیمت پوسته است؟**
- **اگر بله: سکه را کسر کن، پوسته را به وسایل کاربر اضافه کن و بنویس «مبارک است!».**
- **اگر نه: پیام بده «سکه‌ات کافی نیست! می‌خواهی سکه بخری؟».**

پروژه:

فرض کنید تیم شما قرار است یک سیستم ساده برای «کتابخانه مدرسه» به عنوان یک برنامه کامپیوتری بنویسد. ۴ ستون را روی آن پیاده کنید:

شکستن مسئله: این برنامه باید به چه بخش‌های کوچکی تقسیم شود؟ (مثلاً: ثبت کتاب جدید، امانت دادن، ...)

الگوخوانی: چه چیزهایی در بین تمام کتاب‌ها مشترک است؟ (مثل: نام کتاب، نویسنده، ...)

خلاصه‌سازی: چه اطلاعاتی از دانش‌آموز یا کتاب کاملاً غیرضروری است و نباید در دیتابیس ثبت شود؟ (مثلاً: رنگ جلد کتاب؟ قد دانش‌آموز؟)

طراحی الگوریتم: الگوریتم ۳ مرحله‌ایِ «امانت دادن کتاب به یک دانش‌آموز» را بنویسید.

آجرهای ساخت الگوریتم (دستورات، شرط‌ها و حلقه‌ها)

مقدمه: دستور زبان یک کامپیوتر چقدر ساده است؟!

تصور کنید می‌خواهید با یک ربات هوشمند یا یک هوش مصنوعی صحبت کنید. شما نمی‌توانید با او به زبان شعر یا طعنه صحبت کنید! کامپیوترها فقط ۳ نوع آجر و دستور اصلی را می‌فهمند. با همین ۳ آجر کوچک، می‌توان بزرگ‌ترین بازی‌های دنیا مثل GTA یا پیچیده‌ترین هوش مصنوعی‌ها را ساخت!

این سه آجر جادویی عبارتند از:

- **دستورات ترتیبی (Sequence): انجام کارها پشت سر هم.**
- **دستورات شرطی (Selection / If-Else): دوراهی‌ها و تصمیم‌گیری («اگر این‌طور شد، فلان کار را بکن»).**
- **حلقه‌ها و تکرارها (Loops): کارهای تکراری که نباید مدل دستی بنویسیم («این کار را ۱۰ بار تکرار کن»).**

بیایید هر کدام را با مثال‌های جذاب بررسی کنیم:

آجر اول: دستورات ترتیبی (Sequence)

این ساده‌ترین آجر است؛ خط‌به‌خط، از بالا به پایین، بدون هیچ بروه‌برگشتی اجرا می‌شود.

مثال در زندگی روزمره: فرمول مسواک زدن (اول خمیردندان روی مسواک ←← بعد مسواک زدن دندان‌ها ←← بعد شستن دهان).

مثال در برنامه‌نویسی (اسکرچ یا پایتون):وقتی روی صفحه دکمه «شروع بازی» را می‌زنید:

- **موزیک پس‌زمینه پخش شود.**
- **امتیاز اولیه روی عدد صفر تنظیم شود.**
- **کاراکتر بازی در وسط صفحه ظاهر شود.**

آجر دوم: دستورات شرطی (اگر... وگرنه / If... Else)

زندگی پر از دوراهی است؛ کامپیوترها هم همین‌طورند! برنامه‌نویس به کامپیوتر می‌گوید اگر شرایط خاصی برقرار بود فلان کار را بکن، وگرنه کار دیگری انجام بده.

ساختار کلی در کدنویسی:

```
text

اگر (شرطی برقرار بود):
    این کار را انجام بده (بخش اول)
وگرنه:
    کار دیگری انجام بده (بخش دوم)
```

- **مثال ۱ (در یک بازی):**
- **اگر امتیاز بازیکن ≥ ۱۰۰ شد:**
- **پیام بده: «تبریک! شما به مرحله بعد رفتید 🚀»**
- **وگرنه:**
- **بازی را ادامه بده و چیزی نگو.**
- **مثال ۲ (در زندگی واقعی / اپلیکیشن هواشناسی):**
- **اگر دمای هوا کمتر از صفر درجه بود:**
- **کلاه پشمی و کاپشن بپوش!**
- **وگرنه:**
- **با تی‌شرت و لباس سبک بیرون برو.**

آجر سوم: حلقه‌ها و تکرارها (Loops)

فرض کنید باید ۱۰۰ بار بگویید: «من برنامه‌نویسی را دوست دارم!». آیا منطقی است که یک جمله را ۱۰۰ بار کپی-پیست کنید؟! معلوم است که نه. در برنامه‌نویسی از حلقه‌ها (Loops) استفاده می‌کنیم تا کارهای تکراری به جای ما انجام شوند.

مثال ۱ (انیمیشن در اسکرچ یا بازی):

به کاراکتر بازی می‌گویید: «۱۰ بار این کار را تکرار کن: ۱۰ قدم به جلو برو و رنگ لباسه‌ات را عوض کن». (به جای اینکه ۱۰ خط کد تکراری بنویسید، یک حلقه می‌گذارید که ۱۰ بار دور بزند).

مثال ۲ (زندگی روزمره):

- **ورزش در باشگاه:**
- **حلقه تکرار (۱۰ بار):**
- **بارفیکس برو.**
- **۱۰ ثانیه استراحت کن.**
- **وقتی حلقه تمام شد، برو سراغ تمرین بعدی!**