Semicolon

ماهنامه علمی سمیکالن

سال اول - شماره ۲ - مهر ماه سال ۱۴۰۳

انجمن علمی علوم کامپیوتر
دانشگاه کاشان

اویلر در حال نواختن ریاضیات

چگونه با استفاده از فراکتال‌ها، ریاضیات را بنوازیم!؟

مشاهده دنیای اطراف از طریق لنز الگوریتم‌ها

نقشه راه برنامه نویسی

مهارت‌هایی مختص برنامه نویس‌ها...

نشریه علمی سمیکالن

سال اول – شماره دوم – آبان‌ماه ۱۴۰۳

صاحب امتیاز ..... انجمن علوم کامپیوتر دانشگاه کاشان

سر دبیر نشریه ..... رامتین قیامی

مدیر مسئول نشریه ..... سینا طیبی

طراح جلد و صفحات داخلی ..... رامتین قیامی

طراح پازل ..... آرزو پروازی

مدیر اجرایی ..... پریا والی زاده

نویسندگان ..... رامتین قیامی

آرزو پروازی ..... محمد رضا لله گانی

الهه جعفری ..... سینا طیبی

پریا والی زاده ..... یاسین رزمی

..... محمد مرادی

◆ با تشکر فراوان از استادان گرامی، خانم دکتر شمائی و آقای دکتر آسعیدی بابت نظارت و راهنمایی‌های ارزشمندشان.

برای کسب اطلاع از آخرین اخبار و دسترسی به آموزش‌ها،
وبسایت و شبکه‌های اجتماعی ما را دنبال کنید...

کانال یوتیوب
اسکن کنید...

کانال اینستاگرام
اسکن کنید...

کانال تلگرام
اسکن کنید...

وبسایت انجمن علوم کامپیوتر
اسکن کنید...

فهرست مطالب

نویسنده: یاسین رزمی

مهارت‌هایی که همه ما برنامه‌نویسان باید داشته باشیم!!!

شما یک برنامه نویس مشتاق هستید یا شاید فقط علاقه مند به یادگیری بیشتر در زمینه برنامه‌نویسی هستید؟ برای کسب اطلاعات بیشتر در مورد مهارت‌های سخت و نرمی که برنامه‌نویسان برای موفقیت به آن نیاز دارند، ادامه مطلب را بخوانید.

فناوری به ستون فقرات زندگی روزمره ما تبدیل شده و به برنامه نویسانی نیاز است تا این فناوری را به جلو ادامه دهند. گزینه‌ها بی‌پایان هستند: یک برنامه‌نویس مشتاق می‌تواند برنامه بعدی گوشی‌های هوشمند که زندگی را تغییر می‌دهد، به دنیا ارائه دهد، جهان‌های جدیدی در بازی سازد، یا نحوه تعامل و ارتباط آنلاین میلیون‌ها نفر در سراسر جهان را ایجاد کند. اینها تنها چند مورد از روش‌هایی است که برنامه نویسان بر دنیای اطراف خود تأثیر می‌گذارند، اما همه برنامه نویسان دارای ویژگی های مشترک هستند. مهارت های سخت و نرم مورد تقاضا که باعث موفقیت شغلی آنها می‌شود.

مهارت های نرم (Soft Skills):

- ارتباط (شفاهی و نوشتاری)
- کار تیمی و حل تعارض
- تطبیق‌پذیری
- مسئولیت‌پذیری
- مدیریت زمان
- حل مسئله
- یکدلی
- شکیبایی
- کنجکاوی

مهارت های سخت (Hard Skills):

- ساختار داده‌ها و الگوریتم‌ها
- ویرایشگرهای متن
- زبان‌های برنامه‌نویسی شی‌گرا (OOP)
- محیط‌های توسعه یکپارچه (IDE)
- پردازش ابری (Cloud computing)
- توسعه وب
- پایگاه داده و SQL
- کنترل نسخه Git
- کانتینرها (Containers)

ساختارها و الگوریتم های داده

بسیاری از برنامه نویسان فکر می کنند که ساختارهای داده و الگوریتم ها (DSA) فقط چیزی هستند که باید در دانشگاه از آن عبور کنید، اما هرگز در زندگی واقعی به آن‌ها نیاز نخواهید داشت. با این حال، وقتی مصاحبه‌های زیادی شامل سوالات DSA می‌شود، شگفت‌زده می‌شوند. دلایل مختلفی وجود دارد که شرکت‌ها به دانش DSA یک کارمند آینده‌نگر علاقه دارند و اینکه چرا برنامه‌نویسان نیز باید به آن علاقه داشته باشند.

برای بسیاری از شرکت‌ها مانند متا، گوگل، مایکروسافت و آمازون، نوشتن کد تنها مرحله نهایی یک فرآیند طولانی است. بیشتر وقت یک برنامه نویس در واقع صرف در نظر گرفتن بهترین راه برای نزدیک شدن به یک پروژه، از جمله بهترین ساختارهای داده و الگوریتم‌های بهینه برای به کارگیری می‌شود. این تصمیمات تأثیر واقعی بر استفاده از منابع و سودآوری شرکت دارند، بنابراین جای تعجب نیست که DSA ها در فرآیند مصاحبه نقش برجسته ای داشته باشند. حتی برای شرکت‌های خارج از دره سیلیکون (silicon valley) ، این سؤالات مهم هستند زیرا دانش اساسی برنامه‌نویس و توانایی‌های حل مسئله را نشان می‌دهند.

هنگامی که یک برنامه نویس این موقعیت را به دست آورد، DSA ها همچنان در کارهای روزمره نقش دارند. به طور خاص، ساختارهای داده یک روش خاص برای سازماندهی داده‌ها هستند تا بتوان از آنها به طور مؤثر استفاده کرد و گزینه های زیادی وجود دارد. یکی از متداول‌ترین ساختارهای داده، آرایه‌ای است که اقلامی از یک نوع داده مانند اعداد صحیح را نگه می‌دارد و فهرست‌بندی می‌کند. انواع دیگری از ساختارهای داده شامل لیست های پیوندی است که داده‌ها را به ترتیب خطی و متوالی سازماندهی می‌کند. و پشته‌ها، که به برنامه‌نویسان اجازه می دهد ابتدا به آیتم‌های اخیراً قرار داده شده، دسترسی پیدا کنند.

در همین حال، الگوریتم‌ها مجموعه‌ای از دستورالعمل‌هایی هستند که برنامه‌نویسان برای حل یک مشکل به رایانه‌ها می‌دهند، دقیقاً مانند دستور العملی که ممکن است به آشپز داده شود. این دستورالعمل‌های گام به گام می‌توانند وظایف مختلفی را انجام دهند، از جمله جستجو و مرتب‌سازی داده‌ها به روشی که منظم و منطقی باشد.علاوه بر این، بسیاری از استارت‌آپ‌ها، و همچنین کارفرمایان FAANG، به دنبال برنامه‌نویسانی هستند که از چابکی در مقیاس‌بندی برنامه‌ها و نوآوری از طریق استفاده از DSA برخوردار باشند.

Language / IDE	C++	Go	HTML	Java	JavaScript	PHP	Python	Ruby	Other
AWS Cloud9	X	X			X	X	X	X	40 languages
Code:: Blocks	X								C
Eclipse				X					
Eclipse Theia				X	X		X		60 languages
GNAT Studio	X						X		Ada, C, & SPARK
IntelliJ IDEA				X	X				Groovy, Kotlin, Scala, TypeScript, & SQL
NetBeans				X	X	X			JavaFX
PyCharm							X		Flask, Django, web2py, Pyramid, & Google App Engine
SlickEdit	X	X	X	X	X	X	X	X	70 languages
Xcode				X			X	X	Swift, React.js, & Applescript
Visual Studio	X	X		X					C, VB, .NET, C#, F#, CSS, Typescript, XML, XSLT
VS Code	X	X	X	X	X	X	X	X	Many languages

پایگاه داده و SQL

یکی از انتظارات اساسی هر برنامه نویس این است که با مفاهیم اصلی پایگاه داده آشنا باشد. در حالی که زبان‌های زیادی برای کار با پایگاه‌های داده استفاده می‌شود، رایج‌ترین آنها SQL است. اگرچه SQL در دهه ۱۹۸۰ توسعه یافت، اما همچنان زبان استانداردی است که برای برقراری ارتباط با پایگاه‌های داده رابطه‌ای استفاده می‌شود و برای برنامه‌نویسان مدرن حیاتی است. در سال‌های اخیر، SQL به‌شدت توسط پایگاه‌های اطلاعاتی کامپیوتری شخصی مورد استفاده قرار گرفته است، زیرا دسترسی به پایگاه‌های اطلاعاتی توزیع‌شده را تسهیل می‌کند (به عنوان مثال، پایگاه‌های داده‌ای که در چندین سیستم رایانه‌ای پخش شده‌اند). به چندین کاربر محلی اجازه می دهد تا به طور همزمان به یک شبکه دسترسی داشته باشند. SQL همچنین ذخیره سازی و سازماندهی آسان داده‌ها را در پایگاه های داده رابطه ای (به عنوان مثال، پایگاه‌هایی که جداول از طریق داده‌های مشترک به یکدیگر مرتبط هستند) امکان پذیر می‌کند.

ویرایشگر های متن

ویرایشگرهای متن برنامه هایی هستند که باز کردن، مشاهده و ویرایش فایل‌های متنی ساده را امکان‌پذیر می‌کنند. از آنجایی که ویرایشگرهای متن، مانند برنامه های پردازش کلمه، قالب بندی را به متن اضافه نمی‌کنند، برنامه‌نویسان می‌توانند از ویرایشگرهای متن برای نوشتن و ویرایش آسان در زبان های برنامه‌نویسی و نشانه‌گذاری استفاده کنند. علاوه بر این، ویرایشگرهای متن به برنامه نویسان کمک می کنند تا فایل‌های اسنادی را ایجاد کرده و فایل های پیکربندی را حفظ کنند.برخی از پرکاربردترین ویرایشگرهای متن عبارتند از:

Sublime Text

UltraEdit++

Visual Studio Code

Notepad

رایانش ابری (Cloud computing)

رایانش ابری رشد بسیار زیادی را تجربه می‌کند، زیرا برای همه کسب‌وکارهایی که می‌خواهند محیط‌ها، فضای ذخیره‌سازی و دارایی‌های دیجیتال خود را به ابر منتقل کنند، به توسعه‌دهندگان ابری نیاز است. در واقع، طبق گزارش LogicMonitor، ۸۷ درصد از تصمیم‌گیرندگان فناوری اطلاعات جهانی موافق هستند که همه‌گیری COVID-۱۹ مهاجرت به ابرها را برای اکثر سازمان‌ها تسریع کرده است. علاوه بر این، پس از مهاجرت، کسب و کارها به برنامه‌نویسانی آشنا با فناوری لازم برای کار موثر با برنامه های کاربردی ابری نیاز خواهند داشت. از آنجایی که مشاغل بیشتر به علم داده، یادگیری ماشین و هوش مصنوعی متکی هستند، کار در فضای ابری اهمیت بیشتری پیدا می‌کند زیرا الگوریتم‌ها و مدل‌ها منابع قابل توجهی را مصرف می کنند. نتیجه این انتقال و نیازهای تجاری این است که مهندسان و توسعه دهندگان ابری متقاضی زیادی دارند.خبر خوب این است که بسیاری از زبان های مورد نیاز برای رایانش ابری در حال حاضر بهترین زبان برای برنامه نویسان هستند، از جمله:

- Python
- Java
- Ruby
- Go

علاوه بر این، ایده خوبی برای برنامه‌نویسان است که با پلتفرم های ابری مانند:

- خدمات وب آمازون (AWS)
- مایکروسافت آژور
- Google Cloud Platform (GCP)

حتی با تمرکز بر روی تنها یک مورد، برای یادگیری عملکردهای کلیدی، به شما کمک می‌کند تا درک بهتری از نحوه کار دیگران به دست آورید و به مجموعه مهارت‌های خود ارزش بیافزایید.

ارتباط

توانایی توضیح ایده‌ها یا روش‌های حل مسئله، پرسیدن و پاسخ مؤثر در یک محیط گروهی، و کمک به کاهش تعارض از طریق گفتگوی محترمانه برای موفقیت در کدنویسی مهم است.

کار تیمی و حل تعارض

به اشتراک‌گذاری سازنده ایده‌ها و حمایت از ایده‌های دیگران به نوبه خود، یک عنصر کلیدی برای موفقیت تیم است. اما آیا شما را شگفت زده خواهد کرد که بدانید توافق مداوم همیشه سودمند نیست؟ در واقع پس‌زمینه‌ها و ایده‌های متفاوتی یک از اعضای تیم کمک می‌کند تا نتایج بهتر نسبت به نتایج فردی داشته باشد.

حل مسئله

مهارت حل مسئله برای برنامه نویسان به اندازه توانایی فنی مهم است. همانطور که دومینیک سیمونو ریچی، مدیر مهندسی Lever برای HackerNoon نوشت: «هر چه ارشدتر باشید، بیشتر از شما انتظار می رود که مسائل پیچیده و بد تعریف شده، اغلب با زمینه بسیار کمی را بپذیرید. راز واقعی افزایش تأثیر شما این است که یاد بگیرید چگونه با یک مسئله در هر اندازه ای مقابله کنید و آن را به قطعات قابل کنترلی تقسیم کنید که می توانید با موفقیت آن را حل کنید.»

مسئولیت‌پذیری

بسیاری به اشتباه مسئولیت‌پذیری را با «سرزنش» مرتبط می‌دانند، اما زمانی که به طور مؤثر مورد استفاده قرار گیرد، در واقع چیزی کاملاً متفاوت است. مسئولیت‌پذیری قبل از تعیین تکلیف یا نوشتن یک خط کد آغاز می‌شود. به سادگی ایجاد اعتماد بین هم تیمی‌ها از طریق بحث عمومی در مورد هدف، طراحی و جدول زمانی است. به طور خاص، این اعتماد به این معنی است که هر یک از هم تیمی‌ها متعهد می‌شوند تا بهترین کار خود را انجام دهند، به سرعت به تیم اطلاع می‌دهند که آیا مانعی پیش‌بینی نشده وجود دارد، و بدانند که هم تیمی‌ها با یکدیگر همکاری خواهند کرد تا مانع را به بهترین شکل ممکن برطرف کنند. با کار شفاف و تعیین اهداف جمعی و جدول زمانی، مسئولیت پذیری یک پشتیبان است، نه یک شمشیر. حرفه‌ای‌ها می‌توانند این مهارت را با حمایت واقعی از هم تیمی‌های خود به شکلی متقابل برای رسیدن به اهداف کلی خود نشان دهند. در واقع، محبوبیت متدولوژی چابک از طریق مدیریت پروژه اسکرام، نمونه‌ای عالی از کاربرد صحیح مسئولیت‌پذیری است.

یکدلی

توانایی درک واقعی افکار، احساسات و تجربیات دیگران، بدون قضاوت، یک مهارت حیاتی برای برنامه‌نویسان است. همدلی با کاربران نهایی برنامه منجر به توسعه نرم افزاری با سطح رضایت بالاتر و پذیرش بهتر کاربر می‌شود. همدلی با اعضای تیم نه تنها ارتباطات تیم را، بلکه فرهنگ اعتماد و کمک متقابل را نیز تقویت می‌کند. جای تعجب نیست که بسیاری از شرکت‌ها همدلی را به عنوان پنج مهارت نرم برتر رتبه‌بندی می‌کنند.

شکیبایی

این یک فضیلت است. اما نه به دلیلی که ممکن است فکر کنید. افراد صبور معمولاً هنگام برخورد با موانع استرس کمتری دارند. مطالعات نشان داده‌اند که کورتیزول (هورمون استرس) بر عملکرد شناختی، ادراک و مهارت‌های سازمانی تأثیر منفی می‌گذارد که برای کدنویسی موفق بسیار مهم هستند. در نتیجه، صبر (یا فقدان آن) می تواند به طور قابل توجهی بر نتایج پروژه و کیفیت کدگذاری تأثیر بگذارد.

کنجکاوی

مدیر عامل CodeFights تیگران اسلویان برای Tech Beacon می نویسد: "بهترین توسعه دهندگان معمولاً افرادی کنجکاو هستند که دوست دارند یاد بگیرند». این مهارت احتمالاً همان چیزی است که کاوش مداوم، آزمایش مکرر ایده‌های مختلف و جستجوی فعال راه‌های جدید برای بهبود را که محرک‌های کلیدی در رشد و موفقیت برنامه‌نویس هستند، هدایت می‌کند.

تطبیق پذیری

اگر یک چیز در برنامه نویسی ثابت باشد، این است که همه چیز تغییر می‌کند. فناوری تکامل می‌یابد و نیازهای مشتریان چند برابر می‌شود یا تغییر می‌کند. به همین دلیل، ضروری است که برنامه نویسان در مواجهه با تغییرات و شکست‌های گاه به گاه سازگار و انعطاف پذیر باشند. داشتن توانایی ارزیابی آرام آنچه باید انجام شود و سازگاری، کلید موفقیت در این زمینه است.

مدیریت زمان

برنامه نویسان باید بتوانند زمان خود را به طور موثر مدیریت کنند. این شامل همه چیز می‌شود، از تخمین زمان تا تکمیل یک کار، کمک به تیم برای توافق بر سر جدول زمانی قابل تحویل، یا تکمیل وظایف فردی به موقع. اولویت قرار دادن مدیریت زمان نه تنها شما را به صورت فردی سازنده‌تر می‌کند، بلکه شما را به عضوی بهتر و قابل اعتمادتر تبدیل می‌کند. در نتیجه، به همین دلیل است که کارفرمایان این مهارت نرم را بسیار مهم می‌دانند.

کنترل نسخه Git

Git یک سیستم کنترل نسخه است که به برنامه نویسان اجازه می دهد تغییرات کد منبع را در طول فرآیند توسعه مدیریت و پیگیری کنند. تصحیح هر گونه خطایی که ممکن است رخ دهد را آسان می‌کند زیرا هر نسخه ذخیره می‌شود و می توان آن را در صورت درخواست فراخوانی کرد. استفاده از کنترل نسخه، برنامه‌نویسان را تشویق می‌کند تا از طریق آزمون و خطا، نوآوری کنند، زیرا آنها نگران از دست دادن تلاش‌های قبلی برای کدنویسی نیستند.

Git پرکاربردترین سیستم کنترل نسخه در بین کارفرمایان است، بنابراین مهم است که در هنگام یادگیری و یا فعالیت در برنامه نویسی، به خوبی از آن آگاه باشید و آماده استفاده از آن باشید.

زبان های برنامه نویسی شی گرا (OOP)

زبان های شی گرا از روشی برای برنامه نویسی پشتیبانی می کنند که بر کلاس ها و اشیاء متکی است. به کلاس‌هایی مانند گروه‌هایی از چیزهای مشابه، مانند میوه‌ها، با اشیایی فکر کنید که در مورد اقلام فردی در آن کلاس اطلاعات بیشتری به ما می‌دهند، مانند سیب. این موضوع در برنامه نویسی مهم است زیرا به برنامه نویسان اجازه می دهد تا به راحتی از کدهای پیچیده در بین برنامه ها استفاده مجدد کنند. برای مثال، اگر من بگویم «سیب من»، لازم نیست تمام ویژگی‌های سیبم را به شما بگویم (یعنی قرمز، گرد، روی درخت رشد کرده، متعلق به من است). به طور مشابه، با استفاده از یک شی (myApple) از یک کلاس (fruit)، یک برنامه نویس می تواند به راحتی دستورالعمل ها یا اطلاعات را در چندین برنامه مرتبط کند و در نتیجه کدنویسی موثرتر و کارآمدتری را ممکن می سازد.

به همین دلیل، زبان های شی گرا مانند جاوا، سی پلاس پلاس، پایتون و پرل برای برنامه نویسان مهم هستند و آنها باید حداقل یکی را در مجموعه مهارت های خود داشته باشند. علاوه بر این، زبان‌هایی مانند جاوا اسکریپت و PHP به خوبی با زبان های OOP جفت می شوند تا کارایی و عملکرد را بیشتر کنند.

محیط های توسعه یکپارچه (IDE)

با ترکیب انواع ابزارهای توسعه دهنده از طریق یک رابط کاربری گرافیکی (GUI، IDE)ها یک میز کار برای برنامه نویسان هستند که در آن همه ابزارهای مورد نیاز آنها چیده شده و برای استفاده آنها آماده است. مانند یک میز کار با اره، مته، میخ. و اگر قصد ساختن خانه پرندگان را داشتید، یک چکش.

IDE ها از این جهت ارزشمند هستند که با یادگیری یک IDE، یک توسعه دهنده می تواند با انواع ابزارهایی که به صورت هم افزایی کار می کنند آشنا شود، به جای اینکه هر ابزار را جداگانه یاد بگیرد و ابزارهای مناسب را برای هر کار کدنویسی جمع کند. علاوه بر این، از آنجایی که همه ابزارها از طریق یک رابط کاربری گرافیکی در دسترس هستند، برنامه‌نویس نیازی به صرف زمان برای جابجایی بین برنامه‌ها ندارد. توجه به این نکته مهم است که یک IDE ممکن است برای کار با یک یا چند زبان برنامه نویسی طراحی شود.

برای موفقیت در حرفه برنامه نویسی، داشتن مجموعه ای متنوع از مهارت‌های فنی (سخت) و مهارت های غیرفنی (نرم) ضروری است. برنامه‌نویسان نه تنها باید از دانش تخصصی در زمینه های مختلف برنامه نویسی مانند ساختار داده ها، پایگاه‌داده، زبان‌های شی گرا و ابزارهای توسعه برخوردار باشند، بلکه باید مهارت‌های ارتباطی، حل مسئله، کار تیمی، انعطاف پذیری و مدیریت زمان را نیز در خود پرورش دهند. این ترکیب منحصربه‌فرد از مهارت های فنی و غیرفنی به برنامه نویسان کمک می‌کند تا در محیط کاری پویای امروز به طور موثر عمل کنند، با تحولات فناوری سازگار باشند و در نهایت به موفقیت شغلی دست یابند.

فراکتال‌ها و کاربرد آن‌ها در موسیقی

چگونه با استفاده از ریاضیات موسیقی دان شویم !!!

نویسنده: پریا والی زاده

مدت‌هاست که تشخیص داده شده که ریاضیات و موسیقی، علی‌رغم شیوه‌های بیان متفاوتشان، همانطور که هلمهولتز می‌گوید «با هم پیوند خورده‌اند». اولین اکتشاف این پیوند معمولاً به فیثاغورث نسبت داده می‌شود، اما احتمالاً کار پیروان ناشناس او بوده است که مقادیر عددی را به نت‌های کنده شده روی λύρα - لیر اختصاص می‌دادند (افرادی بودند که فواصل موسیقی را متناسب با اعداد بیان کرده و ارتباط موسیقی و ریاضیات را توجیه کردند). اقلیدس، بطلمیوس، کپلر، هویگنس، مرسن، دانیل برنولی، فوریه و اویلر در میان بسیاری از ریاضیدانان و فیزیکدانان مشهوری بودند که به نظریه موسیقی کمک کردند. رساله ژان فیلیپ رامو در سال ۱۷۲۲ در مورد هارمونی لقب «آیزاک نیوتن موسیقی» را برای او به ارمغان آورد. دالامبر به قدری تحت تأثیر اصول ریاضی رامو در موسیقی قرار گرفت که خلاصه‌ای از تئوری‌های آهنگسازی خود را نوشت. طرح کلی در مورد این تاریخچه زیبا در مقاله ۱۹۲۴ R. C Archibald ارائه شده است.

در حال حاضر نیز ریاضیات بطور گسترده در تشریح، آنالیز و ساخت موسیقی استفاده می‌شود؛ مثلا از روابط ریاضی در الگوریتم‌های آهنگسازی برای یافتن صداهای مناسب جهت کوک کردن سازها کمک می‌گیریم و یا در جای دیگری برای ساخت یک آهنگ از ریاضی استفاده می‌شود. باید گفت این موارد نمونه های کوچکی از کاربردهای ریاضیات در موسیقی است. سیستم‌هایی که رابطه موسیقی و ریاضی را شرح می‌دهند اساسا در قرن بیستم توسعه یافتند.

از این میان می‌توان به سیستم آهنگسازی جوزف شیلینگر (Joseph Schillinger) و متد ساخت آهنگ از الیور مسیان (Olivier Messiaen) در سال ۱۹۴۰ اشاره کرد. از آن زمان به بعد متدهای جدیدتر در این زمینه بیشتر بر اساس فراکتال‌ها بودند که با استفاده از آنها افق تازه‌ای برای ساخت نغمات جدیدتر برای آهنگسازان ایجاد شد.

تاریخچه‌ای از فراکتال‌ها

واژه Fractal از کلمه لاتین Fractus یعنی (شکسته) گرفته شده و در لغت به معنی سنگیست که به شکلی نامنظم شکسته شده باشد. یکی از مهمترین خصوصیات فراکتال‌ها، خود متشابه بودن آنهاست به این معنی که فراکتال‌ها از اجزایی تشکیل شده‌اند که هر جزء در آنها شبیه به کل شکل می‌باشد؛ شکل گل کلم و سرخس از معروفترین مثال‌هایی است که برای تعریف یک فراکتال ارائه می‌شود.

برای ملموس تر شدن موضوع اجازه دهید کمی از این هندسه زیبا را در اطرافمان بیابیم: ساختارهای فراکتالی در بسیاری از ساختارهای طبیعی مثل ساختمان دانه‌های برف، شکل کوه‌ها، ابرها و شکل ریشه، تنه و برگ درختان، رویش بلورها در سنگ‌های آذرین، شبکه آبراه‌ها و رودخانه‌ها، رسوبگذاری الکتروشیمیایی، رویش توده باکتری‌ها و سیستم عروق خونی و غیره دیده می‌شوند و با آنها می‌توان پدیده‌های طبیعی بسیاری را تشریح، تفسیر و پیش بینی کرد. موارد کاربرد فراکتال‌ها آنچنان زیاد است که حتی نمی‌توان لیستی از آن‌ها ارائه داد.

چند مورد از کاربردهای روزمره با فراکتال‌ها عبارتنداز: مثلا در کامپیوتر (برای فشرده کردن تصاویر یا پردازش تصاویر)، فیزیک (آنتن‌های گوشی موبایل)، پزشکی (برای تفسیر نوار قلبی و پیش بینی رفتار بدن)، معماری و شهرسازی، اقتصاد، شیمی، پیش‌بینی وضعیت آب و هوا، زمین شناسی و حرکت گسل‌ها و صفحات تکتونیکی زمین و غیره.

فراکتال‌ها چه هستند و چه ارتباطی با موسیقی دارند؟

در سال ۱۹۷۰ دو دانشمند به نام‌های ریچارد واس (Richard Voss) و جان کلارک (John Clarke) از دانشگاه کالیفرنیا به این موضوع پی‌بردند که موسیقی با یک حالت الگوریتم مانند (یعنی یک روال گام به گام کلی برای همه نمونه‌ها)، بسط می‌یابد و لذا از یک فرمول ریاضی می‌توان برای ساخت یک آهنگ استفاده کرد. همچنین آنها کشف کردند که انواع مختلف موسیقی از یک الگو همانند هم پیروی می‌کنند. از سوی دیگر، در همان سال (۱۹۷۰) بنویت مندلبرت (Benoit Mandelbrot) نیز شروع به انجام تحقیقات بر روی فراکتال‌ها کرده بود. این مسئله بسیاری از موسیقی‌دان‌ها و ریاضیدانان را ترغیب به مطالعه بر روی این موضوع نمود. در سال ۱۹۷۵ مندلبرت واژه فراکتال را وارد دنیای ریاضی کرد. از دیگر کارهای با ارزش مندلبرت، به تصویر کشیدن فراکتال‌ها با استفاده از کامپیوتر است.

مندلبرت، پدر هندسه فراکتالی، فراکتال را بدین صورت تعریف می‌کند: "یک شکل فراکتالی، مجموعه‌ای از اشکال در هم پیچیده و مجزاست بطوریکه اگر یک قطعه از آن را بزرگ کنیم، شکل حاصل همانند شکل نخستین در خواهد آمد و همچنین با احتمالی اندک بدشکل و بدریخت خواهد شد"

اساس ساخت فراکتال همان چیزی است که در ریاضی به آن تکرار می‌گوییم. هر گاه روی یک معادله غیرخطی، تکرارپذیری صورت گیرد می‌توان به یک شکل فراکتالی رسید. الگوریتم این کار به این صورت مراحل زیر خواهد بود:

۱- یک ورودی (x) را به تابع بدهید.

۲- جواب تابع را بر حسب ورودی داده شده به آن حساب کنید: $(f(x) = y_1)$

۳- جواب بدست آمده از مرحله قبل (y_1) را مجددا به عنوان یک ورودی جدید درون همان تابع جایگزین کنید: $f(y_1 = y_2)$

۴- برای بدست آوردن مقادیر بعدی، مرحله سوم را به ازای اندیس‌های بعدی، تکرار کنید.

تا بدین جا توانستیم به صورت خیلی ساده برای فراکتالی که به روش تکرار روی تابع دلخواه (f) تولید می‌شود، مقادیر خروجی را به دست آوریم. حال اگر قرار باشد این فراکتال را روی صفحه نمایش در یک کامپیوتر داشته باشیم، باید بین خروجی‌های عددی و پیکسل‌های صفحه نمایش، یک تناظر (نگاشت) ایجاد نماییم. در نهایت بسته به دقتی که روی مقادیر ورودی داشته‌ایم، شکلی دقیق‌تری بدست خواهیم آورد؛ در واقع هرچه مقادیر داده شده به تابع در دقتهای کوچکتری باشند، شکل نهایی نیز دقتی بالاتر خواهد داشت.

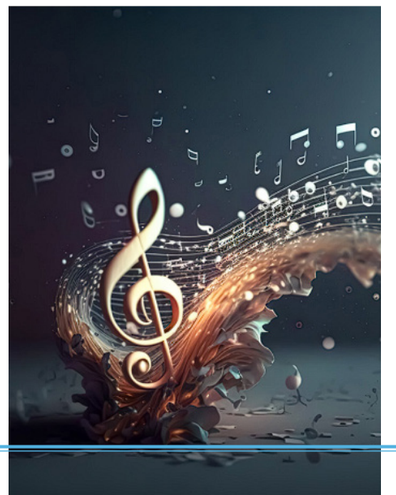
فراکتال‌ها اشکال هندسی عجیبی هستند که قوانین هندسه اقلیدسی را در هم می‌شکنند و نمی‌توان برای توصیف آنها از هندسه اقلیدسی کمک گرفت. در هندسه اقلیدسی اکثر اشکال مثل دوایر، مثلث‌ها، مربع‌ها و دیگر چند ضلعی‌ها دارای محیط و مساحتی مشخص و قابل محاسبه هستند و چنانچه به آنها بنگرید، محیط آنها را یک خط یا منحنی می‌بینید؛ در صورتی که در فراکتال‌ها چنین نیست. در واقع فراکتال‌ها دارای محیط و مساحتی نامتناهی هستند. زیرا هر چه بیشتر روی محیط یک فراکتال دقت کنید، با مقدار بیشتری برآمدگی، منحنی، ناهمواری، شکست و پیچش‌های مختلف روبرو خواهید شد. در نتیجه با وجود این ناهمواری‌ها و منحنی‌های بیشمار، محاسبه فاصله دو نقطه بر روی فراکتال غیرممکن و امری محال است. از آنجایی که اشکال فراکتالی نامتناهی هستند، لذا ترسیم یک فراکتال بصورت کامل و با تمام جزئیات امکان پذیر نمی‌باشد. باید اضافه کرد که با وجود اینکه نمی‌توان شکل دقیقی از یک فراکتال کشید ولی همه ما می‌توانیم فراکتال را بصورت تقریبی رسم کنیم. تقریب استفاده شده در یک شکل فراکتالی به عمق فراکتال برمی‌گردد و در عمق بیشتر، تصویر فراکتالی تقریب زده شده به شکل واقعی نزدیک‌تر خواهد شد.

موسیقی فراکتالی

با استفاده از فراکتال‌ها به طرق مختلف می‌توان به آهنگسازی و صداسازی پرداخته و یا حتی از آنها ایده‌هایی جدیدی برای حلق یک اثر هنری جدید گرفت. راز نهفته در یک موسیقی فراکتالی همان چیزیست که در ریاضیات به آن نگاشت (map) می‌گویند. نگاشت به این معناست که یک ارتباط مستقیم و متناظر بین خروجی‌های عددی (که از معادله حاصل می‌شوند) و پارامترهای خاصی (که برای ساخت آهنگ بکار می‌روند)، ایجاد کنیم. همانطور که برای تصویر کردن یک عکس فراکتالی، خروجی‌های تابع را به پیکسل‌های صفحه نمایش نگاشت می‌کردیم، حال باید پارامترهای ساخت آهنگ را به آن خروجی‌ها نگاشت کنیم.

پارامترهای ساخت آهنگ فراکتالی میتوانند شامل فرکانس‌ها، اوزان، دینامیک و دیگر موارد در آهنگسازی باشند. از آنجا که پارامترهایی که در یک موسیقی فراکتالی بکار می‌روند بیشتر از یک عکس فراکتالی هستند، می‌توان انتظار آهنگ‌هایی متنوع‌تر نسبت به یک عکس فراکتالی را از یک فرمول واحد داشت. در حقیقت عملیات نگاشت، یک ارتباط دادن بین دنیای اعداد و دنیای اصوات است. نگاشت عناصر غیر موسیقایی به نت‌های موسیقی ایده‌ای است که از زمان باستان بوجود آمده است. این تکنیک که به نام ساگتو کاواتو (soggetto cavato) معروف است، حروف مشخصی از کلمات در یک متن نگارش شده را به تعدادی مشخص از نت‌های موسیقی نگاشت می‌کند. جالب است بدانید که این تکنیک بطور گسترده‌ای توسط آهنگسازان تاریخ از جمله یوهان سباستین باخ بکار گرفته شده است. برای مثال Hercules Dux Ferrarie که نام یک موسیقی ساخته شده توسط Josquin des Prez برای هرکول (Hercules) دوک فرارا است از این تکنیک بدین صورت استفاده می‌کند که در آن تم براساس حروف اسم شخص دوک شکل می‌گیرد.

بدین صورت که در ابتدا رشته حروف e-u-e-u-e-a-i-e به شش نت موسیقایی ut, re, mi, fa, sol, la نگاشت می‌شود. یعنی:



*e -> D

*a -> F

*u -> C

*i -> E

و لذا این رشته این صورت خواهد شد:

re, ut, re, ut, re, fa, mi, re. و یا به عبارت دیگر D-C-D-C-D-F-E-D. سپس این دسته نت به عنوان یک تم اصلی در آهنگ مربوطه جای گرفته و با آن کار می‌شود.

در دنیای نگاشت کردن اعداد به عناصر موسیقایی، باید توجه داشت که تنها یک راه واحد و خاص برای بدست آوردن یک خروجی از نگاشت کردن اعداد به عناصر سازنده آهنگ وجود ندارد. بدین معنی که می‌توان نگاشتهای متفاوتی را برای یک خروجی عددی اعمال کرد و به نتایج گوناگونی (آهنگهای متفاوتی) رسید؛ مثلا می‌توان خروجی از یک معادله فراکتال را به چندین مجموعه متفاوت از نت‌ها نگاشت کرد و در نتیجه مدل‌های مختلفی را حاصل کرد. به این آهنگ فراکتالی گوش کنید...

برای توضیح این مطلب که چگونه پروسه نگاشت عمل می‌کند اجازه دهید یکی از الگوریتم‌های معروف را تحلیل کنیم. رشته Morse-Thue این الگوریتم است که اگرچه از نقطه نظر ریاضیات یک سری از اعداد است که ساختار بسیار ساده‌ای را دارند اما بسیاری از خاصیت‌های جالب (همانند خود متشابه بودن) در آن قابل رویت می‌باشد. این رشته اعداد غیر منفی این گونه ساخته می‌شوند که ابتدا اعداد را از ۰ شروع کرده و به ترتیب یکی به آنها افزوده می‌شود (به عبارت بهتر همان شمارش معمولی) و سپس معادل دودویی هر عدد را برای آن قرار می‌دهیم:

۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹, ۱۰, ۱۱...

۰۰۰۰, ۰۰۰۱, ۰۰۱۰, ۰۰۱۱, ۰۱۰۰, ۰۱۰۱, ۰۱۱۰, ۰۱۱۱, ۱۰۰۰, ۱۰۰۱, ۱۰۱۰, ۱۰۱۱...

حال برای هر عدد مجموع آن واحد دودویی را حساب می‌کنیم:

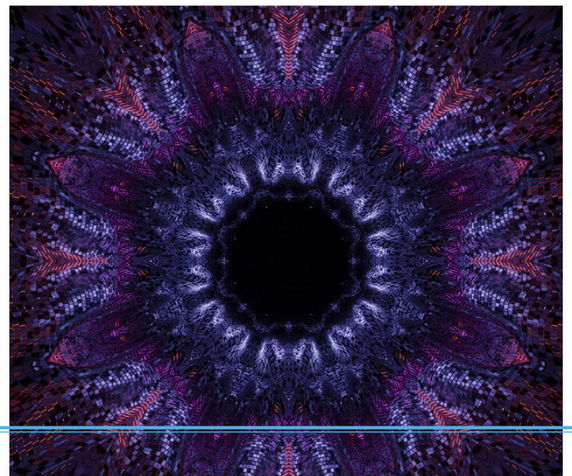
۰, ۱, ۱, ۲, ۱, ۲, ۲, ۳, ۱, ۲, ۲, ۳, ۲, ۳, ۳, ۴ <-

خب، اینجاست که برای تبدیل این اعداد به موسیقی، نگاشت وارد عمل می‌شود. مثلا می‌توانیم اعداد را به یک گام موسیقی نگاشت کنیم و در اینجا گام "دو" یا C، گام دلخواه ما است؛ پس نت C را به عدد ۰ نگاشت خواهیم کرد و نت D را به عدد ۱، E را به ۲، F را به ۳ و G را به ۴ و همینطور الی آخر؛ لذا خواهیم داشت:

C , D , D , E , D , E , E , F , D , E , E , F ....

که این توالی نتها در نهایت همان ملودی فراکتالی ما را می‌سازد. همانطور که مشهود است در این ملودی همواره می‌توان بخش‌هایی را مشاهده کرد که در کل ملودی در حال تکرار هستند و این از همان خاصیت خود متشابه بودن در فراکتال اولیه نشات می‌گیرد و می‌توان این بخش‌ها را خود متشابه ملودی نامید. باید توجه داشته باشید که شما ملزم به انجام تنها یک روش برای نگاشت کردن و حصول یک آهنگ نیستید و مثلا می‌توانید از گام‌های دیگر و یا از هر گروهی از نت‌ها که خود می‌خواهید، نگاشت را انجام دهید. به عنوان یک ایده مثلا شما قادر هستید تا تمام اعداد بدست آمده را در یک عدد صحیح ضرب نمایید تا اعداد متفاوتی بدست آید. یادآوری می‌کنم که شما می‌توانید فراکتال را روی ریتم‌ها، فواصل، دینامیک‌ها و فرکانس‌های متفاوت اعمال کنید.

در حقیقت ساخت ملودی‌های فراکتالی امری کاملا سلیقه‌ای و ذهنی است و چنانچه این ملودی‌ها را تحلیل کنیم، در همه گونه‌های مختلف آنها، خود متشابه بودن و فواصل را در آنها مشاهده خواهید کرد و از آنجایی که محدوده‌ای برای این ساخته‌های ذهنی و تولید آنها وجود ندارد، می‌توان تعداد نامحدودی از این ملودی‌های فراکتالی بوجود آورد. در دیگر الگوریتم‌ها نیز به همین طریق می‌توانید نگاشت به پارامترهای موسیقی را داشته باشید و هر الگوریتمی هم می‌تواند به تنهایی یک مجموعه منحصر بفرد از ملودی‌ها را تولید کند که اثر تولید شده با توجه به نوع نگاشتی که در آن صورت می‌گیرد باز هم متفاوت از دیگری خواهد بود.

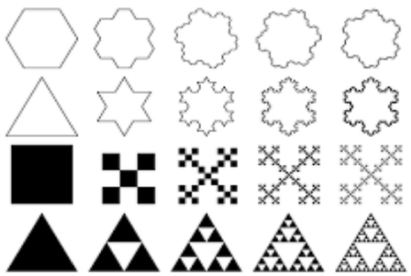


نقطه شروع آهنگسازی با فراکتال‌ها

نرم‌افزارها و برنامه‌های کامپیوتری زیادی برای این منظور وجود دارند که پروسه نگاشت را بطور اتوماتیک محاسبه نمود و نتیجه نهایی را در اختیار آهنگساز قرار می‌دهند و لذا موسیقیدان می‌تواند روی مسائل موسیقیایی متمرکز شود. بدین صورت آهنگساز برای کار کردن با نرم افزار فراکتالی نیاز به اطلاعات چندانی درباره ریاضیات نخواهد داشت. اجازه دهید چند نمونه از این نرم افزارها را ذکر کنیم؛ برای دسته پلتفرم‌های ویندوزی که بیشتر در ایران رایج است می‌توان به نرم افزارهای "FractMus" و "A Musical Generator" و "Art Song and MusicLab I" و "MusiNum" اشاره داشت که از این میان FractMus به دلیل سهولت در استفاده و امکانات گسترده و بهره بردن از الگوریتم‌های متنوع و همچنین قابلیت به تصویر کشیدن یک آهنگ فراکتالی بیشتر مورد توجه قرار گرفته است. برای سیستم‌های مکینتاش نرم‌افزارهای "LoShuMusic and FibonacciBlues" و "Symbolic Composer" را می‌توان نام برد. همچنین زبان برنامه نویسی "Csound" ابزار مناسبی برای نگاشت کردن داده‌ها به پارامترهای صوتی می‌باشد. باید گفت در این سیستم یعنی ساخت آهنگ با استفاده از فراکتال‌ها، همانند دیگر سیستم‌های آهنگسازی، در نهایت ذوق آهنگساز است که کیفیت قطعه موسیقی را تعیین می‌کند و برای خلق یک شاهکار در آن تعداد پارامترهای بکار رفته و پیچیده بودن کار از لحاظ فنی مطرح نیست. در واقع موسیقی فراکتالی برای یک آهنگساز ماهر یک نوع منبع الهام بخش و به عنوان مواد خام است که می‌تواند روی آن بسط و پرداخت صورت دهد تا از درون آن اثری خلق شود.

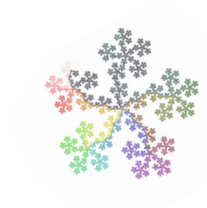
همچنین فراکتال‌ها می‌توانند به روش‌های مختلفی در حوزه موسیقی و صدا اعمال شوند. چند نمونه از این روش‌ها عبارتند از:

۱. ترکیب و ساختار: هندسه فراکتال را می‌توان برای ایجاد آهنگ‌های جدید و منحصر به فرد استفاده کرد. از الگوهای فراکتال می‌توان برای تولید ملودی‌ها، ریتم‌ها و هارمونی‌ها استفاده کرد و در نتیجه موسیقی‌هایی را به نمایش گذاشت که خود شباهتی را



۲. سنتز صدا: فراکتال‌ها را می‌توان برای سنتز صدا نیز به کار برد. با استفاده از الگوریتم‌های فراکتال، می‌توان صداهایی با الگوهای پیچیده و غیرخطی ایجاد کرد که صداهای طبیعی را تقلید می‌کنند یا بافت‌های صوتی کاملاً جدیدی ایجاد می‌کنند. این رویکرد می‌تواند هم در تولید صدای صوتی و هم در تولید صدای الکترونیکی مورد استفاده قرار گیرد که منجر به ایجاد صداهای جدید و مناظر صوتی می‌شود.

۴. اجرا و بداهه نوازی: از الگوهای فراکتال در اجرای زنده و بداهه نیز می‌توان استفاده کرد. نوازندگان و هنرمندان صدا می‌توانند از الگوریتم‌های فراکتال برای ایجاد سیستم‌های مولد استفاده کنند که مواد موسیقایی در حال تکامل را در زمان واقعی تولید، و امکان تولید عبارات صوتی خود به خود و غیرقابل پیش‌بینی را فراهم می‌کنند .



۳. پردازش و تجزیه و تحلیل صدا: تکنیک‌های فراکتال را می‌توان در پردازش و تجزیه و تحلیل صدا به کار گرفت. به عنوان مثال، الگوریتم‌های مبتنی بر فراکتال را می‌توان برای فشرده‌سازی صدا استفاده کرد که امکان ذخیره و انتقال کارآمد داده‌های صوتی را فراهم می‌کند. علاوه بر این، تجزیه و تحلیل فراکتال باعث درک بهتر مناظر صوتی می‌شود که امکان شناسایی الگوها و ساختارها را در ضبط‌های صوتی پیچیده فراهم می‌کند.

۵. تصویرسازی و نصب‌های تعاملی: فراکتال‌ها همچنین برای تجسم موسیقی و ایجاد نصب‌های تعاملی استفاده می‌شوند. با نگاشت پارامترهای موسیقی به الگوهای فراکتال بصری، هنرمندان می‌توانند تجارب بصری همه جانبه‌ای را ایجاد کنند که با محتوای صوتی مطابقت دارد که در نهایت به یک ارتباط ترکیبی بین صدا و تصویر می‌انجامد.

سازه‌هایی که خود تشابهی تقریبی را نشان می‌دهند که در چند مرتبه از مقیاس گسترش می‌یابد، در طبیعت به شکل گیاهان، درختان و خطوط کوهستانی وجود دارند، که شاید دلیل ترجیح زیبایی‌شناختی ما برای آنها در هنرهای تجسمی باشد. الگوهای شبه فراکتال در موسیقی ممکن است به دلایل مشابه جذاب باشند، حتی زمانی که فقط به صورت زودگذر ظاهر شوند. آنها حسی از پیشرفت برازنده را منتقل می‌کنند، گویی هر نت جدید جایگاه طبیعی و مورد انتظار خود را یافته است.

لوی استراوس، که شاید اولین کسی بود که معنای مقیاس فراکتال را در موسیقی توصیف کرد، همچنین آن را به عنوان تکنیکی رد کرد که «از آن نباید چیزی بیش از یک محیط آکوستیک قابل تحمل انتظار داشت».علیرغم اشتیاق او به فراکتال ها در هنر و «پالایش و پیچیدگی» آنها .لوی استروس هشدار داد که «الگوریتم‌های فراکتال، چه در نقاشی و چه در موسیقی، توانایی ایجاد بیش از ژانرهای فرعی را که من آن را تزئینی می‌دانم، ندارند.» شکاف بزرگی این اشیاء اغلب جذاب را از یک نقاشی معتبر یا یک قطعه موسیقی جدا می‌کند.

با این حال، مقیاس بندی قاعده‌مندی‌ها در موسیقی، همانطور که در اینجا درک و نشان داده شده است، ممکن است این ارزیابی اولیه را به چالش بکشد. هنگامی که الگوهای فراکتال به مقدار کم، همراه با سایر ابزارهای موسیقی یا با مجوز هنری به کار می‌روند، ممکن است غنا، بافت و حس اجتناب ناپذیری را به قطعه بیافزایند. براندازی‌های گاه به گاه تقارن‌های ریاضی می‌توانند از ترکیب‌بندی در برابر ظاهری بیش از حد تجویز شده یا مکانیکی محافظت کند. شاهکارهایی که در اینجا تحلیل می‌شوند، مانند بسیاری از نمونه‌های دیگر در ادبیات موسیقی، حاکمیت ذوق بر تکنیک را تأیید می‌کنند: حتی دقیق‌ترین فرمول موسیقی باید برای گوش «خوب» باشد.

نویسنده: رامتین قیامی

هندسه محاسباتی

مشاهده دنیای اطراف از طریق لنز الگوریتم‌ها

هندسه محاسباتی یکی از شاخه‌های جذاب علوم کامپیوتر است که به مطالعه و تحلیل الگوریتم‌ها و داده‌ساختارهایی می‌پردازد که برای حل مسائل هندسی مورد استفاده قرار می‌گیرند. این شاخه به طور خاص به بررسی مسائل مرتبط با اشکال هندسی، نمودارها، و ساختارهای فضایی در فضاهای دو بعدی و سه بعدی می‌پردازد. برای آشنایی بیشتر با دنیای هیجان‌انگیز هندسه محاسباتی و کاربردهای متنوع آن، در این سری و چند سری آینده مجله همراه ما باشید. در این مقالات، به شما نشان خواهیم داد که چگونه این شاخه از علوم کامپیوتر می‌تواند دنیای پیرامون ما را به شیوه‌ای نوین و جذاب تحلیل و بهینه‌سازی کند.

کاربردها و مسائل مهم در هندسه محاسباتی:

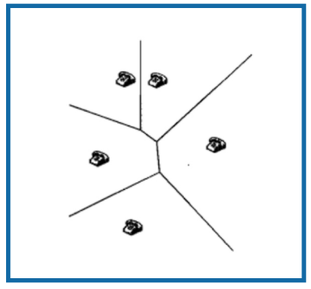
۱. مسیریابی و ناوبری: محاسبه کوتاه‌ترین مسیر بین نقاط مختلف در یک فضا.
۲. تقسیم‌بندی تری‌انگولاسیون: تقسیم‌بندی یک فضا به زیرمجموعه‌های کوچکتر مانند مثلث‌ها.
۳. برخورد و تقاطع: شناسایی نقاط برخورد و تقاطع بین اشیای هندسی مختلف.
۴. پوش‌های محدب (Convex Hulls): یافتن کوچکترین چندضلعی محدبی که مجموعه‌ای از نقاط را دربرمی‌گیرد.
۵. گراف‌ها و شبکه‌ها: تحلیل شبکه‌های جاده‌ای، شبکه‌های الکتریکی، و شبکه‌های اجتماعی.
۶. الگوریتم‌های نزدیک‌ترین همسایه: یافتن نزدیک‌ترین نقطه به یک نقطه مشخص در یک مجموعه از نقاط.
۷. هندسه محاسباتی در گرافیک کامپیوتری: ایجاد تصاویر سه‌بعدی و شبیه‌سازی‌های واقع‌گرایانه.

ابزارها و تکنیک‌ها:

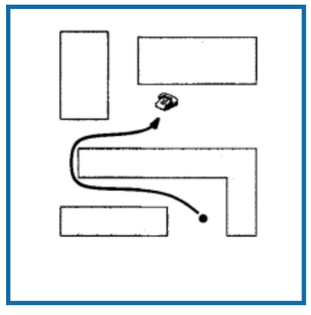
۱. دیاگرام ورونوی (Voronoi Diagram): تقسیم‌بندی فضا به نواحی بر اساس ۱ نزدیکی به مجموعه‌ای از نقاط.
۲. ساختار داده‌های پیشرفته: مانند درخت‌های k-بعدی (k-d trees)، که برای جستجو و مرتب‌سازی در فضاهای چندبعدی استفاده می‌شود.
۳. الگوریتم‌های جستجو و مرتب‌سازی: مانند الگوریتم‌های خط جاروب (Sweep Line) و الگوریتم‌های تقسیم و حل (Divide and Conquer).

هندسه محاسباتی نه تنها در علوم کامپیوتر، بلکه در بسیاری از زمینه‌های مهندسی، رباتیک، جغرافیا، و حتی بیولوژی نیز کاربرد دارد. به عنوان مثال، در رباتیک برای برنامه‌ریزی حرکت ربات‌ها، در جغرافیا برای تحلیل نقشه‌ها و در بیولوژی برای مدل‌سازی ساختارهای مولکولی از آن استفاده می‌شود.

اکنون در حالی که مجله سمیکالن را در دست دارید و احتمالا در محوطه دانشگاه قدم می‌زنید، فرض کنید ناگهان نیاز به برقراری تماس از طریق یک باجه تلفن پیدا کنید. در محوطه دانشگاه تلفن‌های عمومی زیادی وجود دارد و شما می‌خواهید به نزدیک‌ترین باجه تلفن بروید. اما کدام‌یک نزدیک‌تر است؟ نقشه‌ای که محوطه دانشگاه را به مناطقی تقسیم کند و برای هر منطقه نزدیک‌ترین تلفن عمومی را نشان دهد، مفید خواهد بود. این مناطق چه شکلی خواهند داشت؟ و چگونه می‌توانیم آنها را محاسبه کنیم؟ البته پاسخ این نیاز امروزه در جیب شماست اما می‌توان بجای تلفن، نیاز فوری به سرویس بهداشتی یا موارد دیگر را نیز در نظر گرفت :)



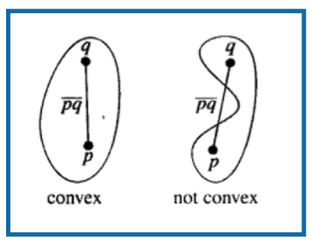
اما در دنیای واقعی کوتاه ترین مسیر بین شما و مقصدتان همیشه خط واصل بین این دو نیست و در نقشه یک محوطه احتمالاً موانع بسیاری در رسیدن به تلفن و یا سرویس بهداشتی بر سر راه ما خواهد بود. اما برنامه‌ریزی یک ربات برای انجام همین کار بسیار دشوارتر است. در این مسئله، مشکل اصلی هندسی است: باید یک اتصال کوتاه بین دو نقطه، بدون برخورد با موانع، پیدا کنیم. حل این مسئله به نام «برنامه‌ریزی حرکت» در رباتیک اهمیت زیادی دارد.



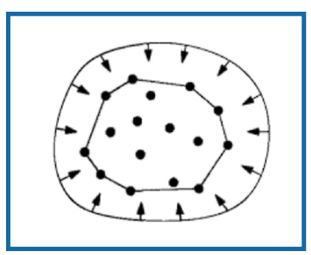
همچنین فرض کنید یک نقشه ندارید بلکه دو نقشه دارید (یکی با توضیحات ساختمان‌ها و دیگری با جاده‌ها). برای برنامه‌ریزی حرکت به تلفن عمومی باید این نقشه‌ها را روی هم قرار دهید و اطلاعات آنها را ترکیب کنید. این کار یکی از عملیات پایه‌ای سیستم‌های اطلاعات جغرافیایی است و شامل مکان‌یابی موقعیت اشیاء از یک نقشه به نقشه دیگر است.

تاریخچه هندسه محاسباتی

در دهه ۱۹۷۰، حوزه هندسه محاسباتی پدید آمد که به این گونه مسائل هندسی می‌پرداخت. می‌توان آن را به عنوان مطالعه سیستماتیک الگوریتم‌ها و ساختارهای داده‌ای برای اشیای هندسی تعریف کرد، با تمرکز بر الگوریتم‌های دقیق که از لحاظ زمانی بهینه هستند. بسیاری از محققان توسط چالش‌های مطرح شده توسط مسائل هندسی جذب شدند. مسیر از فرمول‌بندی مسئله تا راه‌حل‌های کارآمد و زیبا اغلب طولانی بوده و دارای نتایج واسطه‌ای دشوار و غیر بهینه بسیاری است. امروزه مجموعه‌ای غنی از الگوریتم‌های هندسی کارآمد و قابل درک وجود دارد که به سادگی قابل پیاده‌سازی هستند.

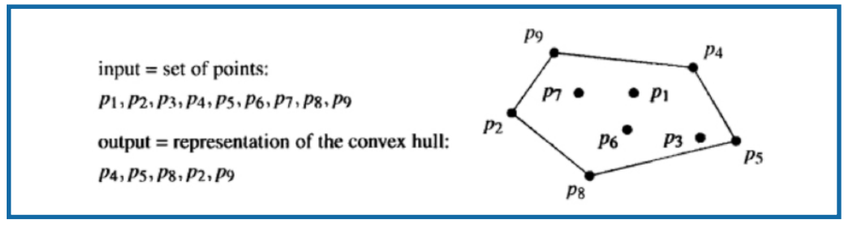


همانطور که از قبل می‌دانید، یک زیرمجموعه S از صفحه، محدب نامیده می‌شود اگر و تنها اگر برای هر جفت نقطه p و q در S، پاره‌خط pq کاملاً در S قرار گیرد به زبانی دیگر پوسته محدب CH(S) که نقاط مجموعه S را در بر می‌گیرد، کوچکترین مجموعه محدبی است که S را شامل می‌شود.

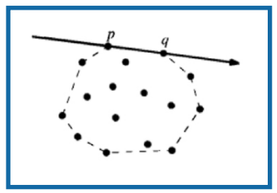


ما می‌خواهیم مسئله‌ی محاسبه‌ی پوسته محدب یک مجموعه متناهی P از n نقطه در صفحه را بررسی کنیم. برای تجسم پوسته محدب، می‌توانیم یک آزمایش ذهنی انجام دهیم. تصور کنید نقاط مانند میخ‌هایی هستند که از صفحه بیرون زده‌اند. حالا یک کش پلاستیکی را بگیرید و دور میخ‌ها بپیچید و رها کنید. کش دور میخ‌ها پیچیده و کوتاه‌ترین مسیر ممکن را ایجاد می‌کند. ناحیه‌ای که توسط نوار لاستیکی محصور می‌شود، همان پوسته محدب P است.

اکنون به این سوال می‌رسیم که چگونه می‌توانیم پوسته محدب را محاسبه کنیم؟ قبل از اینکه به این سوال پاسخ دهیم، باید سوال دیگری بپرسیم: محاسبه پوسته محدب در یک کامپیوتر به چه معناست؟ همانطور که دیدیم، پوسته محدب P یک چندضلعی محدب است. یک روش طبیعی برای نمایش یک چندضلعی این است که رئوس آن را به ترتیب ساعت‌گرد، با شروع از یک نقطه دلخواه، در کامپیوتر با استفاده از روش‌های گوناگون، فهرست کنیم. بنابراین مسئله‌ای که می‌خواهیم حل کنیم این است: با توجه به فهرست $P=\{P_1, P_2, ..., P_n\}$ از نقاط در صفحه، لیستی که حاوی آن نقاط از P است که رئوس CH(P) هستند و به ترتیب ساعت‌گرد فهرست شده‌اند، محاسبه کنیم.



وقتی بخواهیم الگوریتمی برای محاسبه پوسته محدب طراحی کنیم تعریف اول پوسته محدب کمک زیادی نمی‌کند. این تعریف درباره تقاطع رندوم تمام مجموعه‌های محدب که شامل P می‌شوند صحبت می‌کند که تعداد آن‌ها بی‌شمار است. خب مشخص است که استفاده از این روش زمان زیادی از ما خواهد گرفت و پیدا کردن تمام این نقاط و در نهایت رسیدن به بهترین حالت کاملا غیر بهینه است.



اما روش بهینه‌تر چیست؟ برای یافتن پاسخ، بهتر است که به لبه‌های CH(P) نگاهی دقیق‌تر بیندازیم. هر دو راس p و q از یک ضلع، قطعا نقاطی از مجموعه P هستند و اگر خطی که از p و q می‌گذرد را به گونه‌ای جهت‌دهی کنیم که CH(P) در سمت راست قرار گیرد، تمام نقاط P باید در سمت راست این خط باشند. برعکس آن نیز صادق است: اگر تمام نقاط P{p,q} در سمت راست خط جهت‌دار از p و q قرار گیرند، آنگاه pq یک لبه از CH(P) است. حالا که هندسه مسئله را بهتر درک کردیم، می‌توانیم یک الگوریتم توسعه دهیم.

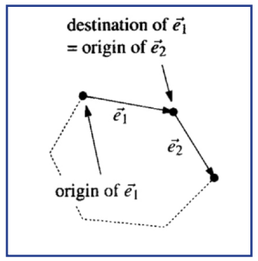
شبه کد الگوریتم کانوکس هال:

```
Algorithm ConvexHull(P):
    Input: A set P of points in the plane
    Output: A list L containing the vertices of CH(P) in clockwise order
E --> NULL
    1. for all ordered pairs (p, q) Exist in P × P with p not equal to q:
            2. do valid  Exist in true
        3. for all points r  Exist in P not equal to p or q:
            4. do if r lies to the left of the directed line from p to q
                    5. then valid  Exist in false
        6. if valid then Add the directed edge pq to E:
    7. From the set E of edges construct a list L of vertices of CH(P), sorted in
8. clockwise order
```

دو خط در شبه کد ما شاید به طور کامل واضح نباشد که در ادامه آنها را توضیح خواهم داد...

اولین مورد خط ۵ است: چگونه می‌توانیم بررسی کنیم که یک نقطه در سمت چپ یا راست یک خط جهت‌دار قرار دارد؟ این یکی از عملیات اولیه مورد نیاز در اکثر الگوریتم‌های هندسی است. (فعلا در این شماره از مجله فرض می‌کنیم که چنین عملیاتی در دسترس و بدیهی است) واضح است که این عملیات می‌تواند در زمان ثابت انجام شود بنابراین پیاده‌سازی واقعی آنها تأثیری بر زمان اجرای اسیمپتوتیک (مرتبه اجرایی)در مرتبه بزرگی نخواهد داشت. این به معنای این نیست که چنین عملیات‌های اولیه‌ای بی‌اهمیت یا ساده هستند. پیاده‌سازی صحیح آنها دشوار است و بر زمان اجرای واقعی الگوریتم تأثیر می‌گذارد. خوشبختانه در اکثر مواقع کتابخانه‌های حاوی چنین عملیات‌های اولیه‌ای، در دسترس هستند.

مرحله دیگر از الگوریتم که نیاز به توضیح دارد، آخرین مرحله است. در حلقه خطوط ۲-۷ مجموعه E از اضلاع پوسته محدب را تعیین می‌کنیم. از E می‌توانیم لیست L را به صورت زیر بسازیم. اضلاع در مجموعه E جهت‌دار هستند، بنابراین می‌توانیم در مورد مبدأ و مقصد یک ضلع صحبت کنیم. از آنجا که اضلاع به گونه‌ای جهت‌دار هستند که نقاط دیگر در سمت راست آنها قرار دارند، مقصد یک ضلع بعد از مبدأ آن قرار می‌گیرد وقتی که رئوس به ترتیب ساعت‌گرد فهرست می‌شوند. اکنون یک ضلع دلخواه e_1 را از E حذف کنید. مبدأ e_1را به عنوان اولین نقطه در L قرار دهید و مقصد آن را به عنوان نقطه دوم. ضلع e۲ را در E پیدا کنید که مبدأ آن مقصد e_1 باشد، آن را از E حذف کنید و مقصد آن را به L اضافه کنید. سپس، ضلع e_2را پیدا کنید که مبدأ آن مقصد e_2 باشد، آن را از E حذف کنید و مقصد آن را به L اضافه کنید. ما به این روش ادامه می‌دهیم تا فقط یک لبه در E باقی بماند. سپس کار تمام است؛ مقصد ضلع باقی‌مانده ضرورتاً مبدأ e_1است که قبلاً اولین نقطه در L بوده است. پیاده‌سازی ساده این فرآیند O(n) زمان می‌برد. این به راحتی می‌تواند به O(n log) n بهبود یابد اما زمان مورد نیاز برای بقیه الگوریتم در هر صورت بر زمان کلی اجرا غالب است.



زمان کلی اجرا

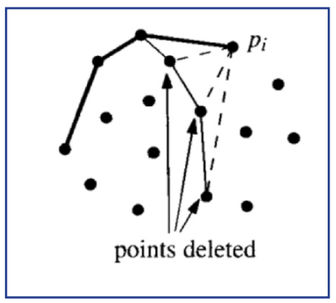
تحلیل پیچیدگی زمانی CONVEXHULL آسان است. ما n^۲ - n جفت نقطه را بررسی می‌کنیم. برای هر جفت به n - ۲ نقطه دیگر نگاه می‌کنیم تا ببینیم آیا همه آنها در سمت راست قرار دارند یا خیر. این در کل O(n^۳) زمان می‌برد. مرحله نهایی O(n^۲) زمان می‌برد، بنابراین زمان کلی اجرا O(n^۳) است. یک الگوریتم با زمان اجرای درجه سه برای استفاده عملی، حتی در کوچک‌ترین مجموعه‌های ورودی خیلی کند است. مشکل این است که ما از هیچ تکنیک طراحی الگوریتمی هوشمندانه‌ای استفاده نکردیم؛ اما آیا روش خلاقانه‌تری هم برای حل وجود دارد؟

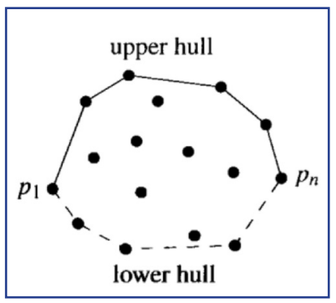
الگوریتم اینکریمنتال (incremental algorithm)

برای رفع مشکلات و بهینه‌تر شدن روش حل، از الگوریتم افزایشی یا incremental algorithm استفاده خواهیم کرد. این روش بدین گونه است که نقاط مجموعه P را یک به یک اضافه کرده و پس از هر افزودن، راه‌حل خود را به‌روزرسانی می‌کنیم. در واقع نقاط را هر بار از چپ به راست به مجموعه خود اضافه می‌کنیم و در نهایت مجموعه‌ای از نقاط (P_1...P_n) که بر اساس مختصات x مرتب شده‌اند، خواهیم داشت. چون ما از چپ به راست کار می‌کنیم، راحت‌تر است که رأس‌های پوسته محدب نیز به ترتیبی که در مرز قرار دارند، از چپ به راست مرتب شوند. اما این همیشه درست نیست. بنابراین، ابتدا فقط رئوسی از پوسته محدب را محاسبه می‌کنیم که در بخش بالای پوسته قرار دارند. برای اینکار بخشی از پوسته محدب را از نقطه چپ گرفته تا نقطه راست (Pn) پیمایش و به لیست اضافه می‌کنیم تا هنگامی که رئوس به ترتیب مخالف مخالف پیمایش شوند. به عبارت دیگر مختصات x نقطه بعد از نقطه فعلی، مقدار کمتری داشته باشد.

اکنون بخش بالایی، حاوی لبه‌های ترازوی محدب است که چندضلعی محدب ما را از بالا محدود می‌کند. در مرحله دوم که از راست به چپ انجام می‌دهیم، قسمت باقی مانده ترازوی محدب یعنی قسمت پایینی را نیز با همین روش محاسبه می‌کنیم و رئوس را به لیست می‌افزاییم.

گام اساسی در الگوریتم افزایشی به‌روزرسانی پوسته بالایی پس از اضافه کردن یک نقطه pi است. وقتی که در مرز یک چندضلعی در جهت عقربه‌های ساعت حرکت می‌کنیم، در هر رأس یک پیچ ایجاد می‌کنیم. برای یک چندضلعی دلخواه، این می‌تواند هم پیچ به راست و هم پیچ به چپ باشد، اما برای یک چندضلعی محدب، هر پیچ باید یک پیچ به راست باشد. این موضوع نحوه اضافه کردن pi را به شکل زیر پیشنهاد می‌دهد. فرض کنیم Lupper یک لیست است که رأس‌های بالایی را به ترتیب از چپ به راست ذخیره می‌کند. ابتدا pi را به Lupper اضافه می‌کنیم. این صحیح است زیرا pi راست‌ترین نقطه از نقاط اضافه شده تاکنون است، بنابراین باید در پوسته بالایی باشد. بعد، بررسی می‌کنیم که آیا سه نقطه آخر در Lupper یک پیچ به راست ایجاد می‌کنند یا نه. اگر این طور باشد، کار بیشتری لازم نیست؛ Lupper حاوی رأس‌های پوسته بالایی p۱، ...، pi است و می‌توانیم به نقطه بعدی، pi+۱ برویم. اما اگر سه نقطه آخر یک پیچ به چپ ایجاد کنند، باید نقطه میانی را از پوسته بالایی حذف کنیم. در این صورت هنوز کار تمام نشده است: ممکن است سه نقطه آخر جدید هنوز یک پیچ به راست ایجاد نکنند، در این صورت باید دوباره نقطه میانی را حذف کنیم. به این روش ادامه می‌دهیم تا سه نقطه آخر یک پیچ به راست ایجاد کنند یا تا زمانی که فقط دو نقطه باقی بمانند.





Algorithm ConvexHull**(**P**):**

Input**:** A set P of points in the plane**.**

Output**:** A list L containing the vertices of CH**(**P**)** in clockwise order**.**

1. Sort the points by x-coordinate**,** resulting in a sequence p**1,** p**2, ...,** pn**.**

2. Put the points p**1** and p**2** in a list L_{upper}**,** with p**1** as the first point**.**

3. For i **=** n**3** to n

4. Append pi to L_{upper}**.**

5. While Lupper contains more than two points and the last three points in L_{upper} do not make a right turn

6. Delete the middle of the last three points from L_{upper}**.**

7. Put the points pn and pn**1-** in a list L_{lower}**,** with pn as the first point**.**

8. For i **=** n**2-** down to **1**

9. Append pi to L_{lower}**.**

10. While L_{lower} contains more than two points and the last three points in L_{lower} do not make a right turn

11. Delete the middle of the last three points from L_{lower}**.**

12. Remove the first and the last point from L_{lower} to avoid duplication of the points where the upper and lower hull meet**.**

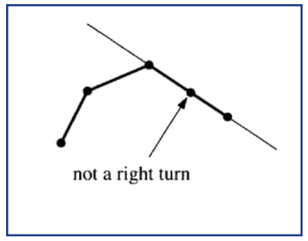
13. Append L_{lower} to L_{upper}**,** and call the resulting list L**.**

14. Return L**.**

اگر دوباره نگاهی دقیق‌تر به این شبه کد بیاندازید، متوجه خواهید شد که الگوریتم بالا به درستی کار نمی‌کند. در اینجا فرض کردیم که هیچ دو نقطه‌ای، x-coordinate یکسانی ندارند. اگر این فرض نادرست باشد، ترتیب نقاط بر اساس x-coordinate به درستی تعریف نمی‌شود. خوشبختانه، این مشکل جدی نیست. ما فقط باید ترتیب‌دهی را به شکل مناسب‌تری تعمیم دهیم: به جای اینکه تنها از x-coordinateاستفاده کنیم، ابتدا نقاط را بر اساس x-coordinate مرتب می‌کنیم و اگر نقاط x-coordinate یکسانی داشته باشند بر اساس y-coordinate مرتب می‌کنیم.

مورد خاص دیگری که نادیده گرفتیم این است که سه نقطه‌ای که باید تعیین کنیم آیا چپ یا راست می‌چرخند ممکن است روی یک خط مستقیم قرار گیرند. در این صورت نقطه میانی نباید در پوش محدب قرار گیرد، بنابراین نقاط هم‌خط باید به گونه‌ای رفتار شوند که گویی یک پیچ چپ می‌زنند. به عبارت دیگر، باید آزمایشی داشته باشیم که اگر سه نقطه یک پیچ راست می‌زنند، درست برگرداند و در غیر این صورت نادرست (توجه کنید که این ساده‌تر از آزمایشی است که در الگوریتم قبلی نیاز بود وقتی که نقاط هم‌خط بودند).

با این اصلاحات، الگوریتم به درستی پوش محدب را محاسبه می‌کند: اولین اسکن پوسته بالایی که اکنون به عنوان قسمتی از پوش محدب تعریف شده است که از کوچک‌ترین رأس بر اساس مختصات x، تا بزرگ‌ترین رأس را اجرا می‌کند، و اسکن دوم قسمت پایینی پوش محدب را محاسبه می‌کند.



حوزه‌های کاربردی

همانطور که قبلاً ذکر شد، ما برای هر مفهوم هندسی، الگوریتم یا ساختار داده‌ای، یک مثال کاربردی انتخاب کرده‌ایم. بیشتر این کاربردها از حوزه‌های گرافیک کامپیوتری، رباتیک، سیستم‌های اطلاعات جغرافیایی و CAD/CAM نشأت می‌گیرند. برای کسانی که با این حوزه‌ها آشنایی ندارند، توضیح مختصری از این حوزه‌ها و برخی از مسائل هندسی که در آن‌ها بوجود می‌آیند، ارائه می‌دهیم.

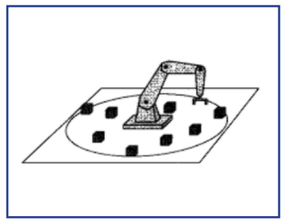
گرافیک کامپیوتری

گرافیک کامپیوتری به ایجاد تصاویر از صحنه‌های مدل‌سازی شده برای نمایش بر روی صفحه نمایش کامپیوتر، چاپگر یا سایر دستگاه‌های خروجی می‌پردازد. این صحنه‌ها از نقاشی‌های دو بعدی ساده که شامل خطوط، چندضلعی‌ها و سایر اشیاء ابتدایی هستند تا صحنه‌های سه بعدی واقع‌گرایانه شامل منابع نوری، بافت‌ها و غیره متغیر هستند. امروزه اینگونه تصاویر می‌توانند به راحتی بیش از چندین میلیون چند ضلعی، شکل و منحنی داشته باشند. از آنجایی که این تصاویر از اشیاء هندسی تشکیل شده‌اند، الگوریتم‌های هندسی نقش مهمی در گرافیک کامپیوتری ایفا می‌کنند. در گرافیک دو بعدی، معمولا تصاویر شامل تقاطع برخی از عناصر ابتدایی، تعیین عنصری که با ماوس اشاره شده است، یا تعیین زیرمجموعه‌ای از عناصر که در یک منطقه خاص قرار دارند، می‌باشد. هنگام برخورد با مسائل سه بعدی، اشکال و تصاویر هندسی پیچیده‌تر می‌شوند. یک مرحله حیاتی در نمایش یک تصویر سه بعدی، حذف سطوح پنهان است؛ یعنی تعیین بخشی از صحنه که از یک دیدگاه خاص قابل مشاهده است یا به عبارت دیگر، حذف بخش‌هایی که پشت سایر اشیاء قرار دارند.

برای ایجاد صحنه‌های واقع‌گرایانه باید نور و پرسپکتیو و یا ژرفنمایی را نیز در نظر بگیریم. این موضوعات مشکلات جدیدی ایجاد می‌کنند، مانند محاسبه سایه‌ها. بنابراین، ایجاد تصویر واقع‌گرایانه نیاز به تکنیک‌های نمایش پیچیده‌ای مانند ردیابی پرتو و رادیوزیتی و یا درخشندگی و .... دارد. در حالتی که با اشیاء متحرک و در برنامه‌های واقعیت مجازی سروکار داریم، شناسایی برخوردهای بین اشیاء بسیار حیاتی است. همه این موارد مشکلات هندسی را در بر دارند.

رباتیک

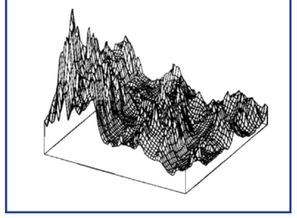
علم رباتیک به مطالعه طراحی و استفاده از ربات‌ها می‌پردازد. زیرا ربات‌ها اشیاء هندسی هستند که در فضای سه‌بعدی - دنیای واقعی - عمل می‌کنند، بنابراین روشن است که در بسیاری از جاها مسائل هندسی پیش می‌آید. در آغاز این مطلب ما مسأله برنامه‌ریزی حرکت را معرفی کردیم که در آن ربات باید مسیری را در محیطی با موانع پیدا کند. برنامه‌ریزی حرکت یک جنبه از مسأله عمومی برنامه‌ریزی وظیفه است. کسی می‌خواهد به ربات وظایف سطح بالا مانند «خانه را جمع و جور کن!» را بدهد و به ربات اجازه می‌دهد تا بهترین راه برای اجرای وظیفه را بیابد. این شامل برنامه‌ریزی حرکات، برنامه‌ریزی ترتیب اجرای زیر وظایف و غیره می‌شود.



مسائل هندسی دیگر در طراحی ربات‌ها و سنسور تشخیص محیط در این ربات‌ها، به وجود می‌آید. اکثر ربات‌های صنعتی بازوهای رباتی با پایه ثابتی هستند. معمولا قطعاتی وجود دارند که ربات باید بتواند به وسیله سنسور خود آن را تشخیص داده و سپس با استفاده از بازوی خود بر روی آنها کاری مانند جابجایی انجام دهد. برخی از قطعات ممکن است بی‌حرکت باشند و برخی نیز ممکن است قبل از اینکه ربات بتواند بر روی آن‌ها کار کند، به جهت شناخته شده‌ای چرخانده شوند. این همه مسائل هندسی هستند،گاهی با مؤلفه سینماتیک.

سیستم‌های اطلاعات جغرافیایی

یک سیستم اطلاعات جغرافیایی یا GIS برای ذخیره داده‌های جغرافیایی مانند شکل کشورها، ارتفاع کوه‌ها، مسیر رودخانه‌ها، نوع گیاهان در مکان‌های مختلف، چگونگی تراکم جمعیت یا بارش مورد استفاده قرار می‌گیرد. GIS همچنین می‌تواند ساختارهای ساخته‌شده توسط انسان مانند شهرها، جاده‌ها، راه‌آهن‌ها، خطوط برق یا لوله‌های گاز را ذخیره کند. یک GIS می‌تواند برای استخراج اطلاعات درباره مناطق خاص و به خصوص برای به دست آوردن اطلاعات درباره رابطه بین انواع مختلف داده‌ها استفاده شود. به عنوان مثال، یک زیست‌شناس ممکن است بخواهد رابطه بین بارش متوسط و وجود گیاهان خاص را بررسی و یک مهندس عمران نیاز داشته باشد تا یک GIS را جستجو کند تا مشخص کند که آیا زیر قطعه زمینی که در آن کاوش می‌کند، حفره گازی و یا تخلخلی وجود دارد یا خیر.



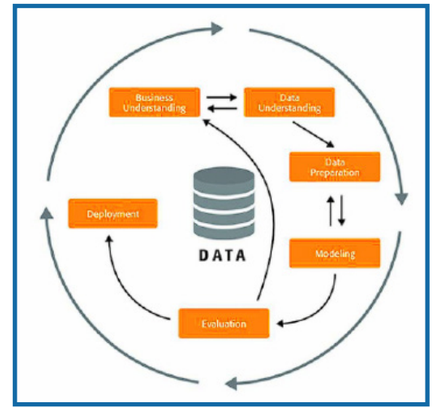
چون بیشتر اطلاعات جغرافیایی مربوط به ویژگی‌های نقاط و مناطق روی سطح زمین هستند، مسائل هندسی به طور فراوان در اینگونه حوزه‌ها پدید می‌آید. علاوه بر این، مقدار داده‌ها به حدی زیاد است که الگوریتم‌های کارآمد ضروری هستند. یکی از اولین سوالات این است که چگونه داده‌های جغرافیایی را ذخیره کنیم. فرض کنید می‌خواهیم یک سیستم راهنمایی خودرو توسعه دهیم که در هر لحظه به راننده نشان دهد که در کجاست. این نیاز به ذخیره نقشه‌ی بزرگ جاده‌ها و دیگر داده‌ها دارد. در هر لحظه باید قادر باشیم موقعیت خودرو را در نقشه تعیین کرده و برای نمایش در کامپیوتر بر روی خودرو، قسمت کوچکی از نقشه را سریع انتخاب کنیم. ساختارهای داده‌ای کارآمد برای این عملیات لازم است.

حتی در برخی از حوزه‌هایی که به نظر می‌آید در آن‌ها مسائل هندسی وجود ندارد، الگوریتم‌های هندسی می‌توانند بهره‌مندی کنند، زیرا اغلب امکان دارد مسئله‌های غیر هندسی را با اصطلاحات هندسی فرموله کرد. به عنوان مثال، در شماره‌های بعدی مجله خواهیم دید که چگونه رکوردها در پایگاه داده می‌توانند به عنوان نقاط در یک فضا، با بعد بالاتر تفسیر شوند، و یک ساختار داده هندسی را ارائه می‌دهیم که امکان پاسخگویی به برخی پرسش‌ها درباره رکوردها را به صورت کارآمد فراهم می‌کند.

امیدواریم که مجموعه مسائل هندسی فوق بتواند روشن سازد که هندسه محاسباتی چه نقش مهمی در بسیاری از حوزه‌های مختلف علوم کامپیوتر و زندگی ما دارد. الگوریتم‌ها، ساختارهای داده و تکنیک‌های توضیح داده شده، ابزارهایی را فراهم خواهند کرد که شما را برای حل مسائل هندسی آماده می‌سازند. این حوزه با استفاده از ترکیبی منحصر به فرد از ریاضیات، الگوریتم‌ها و محاسبات، نقش بسزایی در حل مسائل پیچیده در علوم مختلف ایفا می‌کند. از مسائل متنوعی همچون محاسبه‌ی پوسته محدب، بهینه‌سازی مسیرها و استفاده از اطلاعات جغرافیایی، به عنوان نمونه‌هایی از کاربردهای این حوزه، گفتیم. امیدواریم که این مقدمه‌ی کوتاه، شما را به جذب بیشتر اطلاعات و کاربردهای دیگر هندسه محاسباتی ترغیب کند. در شماره‌های آینده، شما را با مباحث و پژوهش‌های پیشرفته‌تر در این حوزه آشنا خواهیم کرد.

نویسنده: آرزو پروازی

دیتا ماینینگ کند و کاوی در دنیای داده‌ها



داده کاوی یا دیتا ماینینگ چیست؟

داده‌کاوی به صورت کاملا خلاصه فرایند است که داده‌های خام را به اطلاعات مفیدی و قابل بررسی تبدیل می‌کند. الگوها و روابطی را بین دسته داده‌ها پیدا می‌کند و سپس با تحلیل و بررسی آن‌ها، به حل مسائل می‌پردازد. داده کاوی یکی از مراحل اصلی کار با داده‌ها است. اساس علم داده کاوی، بر سه پایه استوار است. متدهای هوش مصنوعی و زیرشاخه‌های آن بخصوص یادگیری ماشین (machine learning)، علم آمار و کار با دسته داده‌ها (جمع‌آوری و مدیریت داده‌ها)

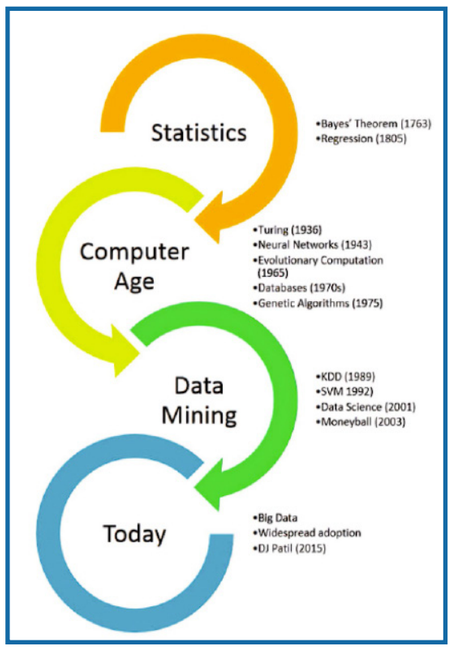
چند مثال برای اینکه درک کنید چرا این علم اهمیت دارد:

همه ما معمولا قبل از رفتن به سفر، آب و هوای مقصد را بررسی می‌کنیم تا از وضعیت آب و هوایی مقصدمان در طی روزهایی که آنجا هستیم خبردار شویم تا مبادا شرایط جوی بد ما را غافلگیر کند. یا مثلا با جمع آوری اطلاعات گوناگون در مورد یک موضوع خاص، خلاصه‌ای دقیق و کاربردی از مطلب مورد نظر خود بتوانیم بنویسیم. دو نوع مختلف داده‌کاوی هم این چنین کمکی به ما می‌کند. همانند مثالی که زده شد، با پیدا کردن روابط و الگوها از طریق داده‌های جمع‌آوری شده، قادر خواهیم بود تا رفتار مشتریان خود را بسنجیم، پیش بینی کنیم و در نهایت محصول را با الگویی که ممکن است بیشتر مورد پسند مشتریان باشد پیدا کنیم و به آن‌ها پیشنهاد دهیم. همچنین داده کاوی ابزاری برای تشخیص آنومالی یا اعمال غیر عادی(خلاف قاعده) است که به ما کمک می‌کند تا خلافکاران را با کمک الگوریتم‌ها و تحلیل‌های رفتاری، راحت‌تر شناسایی و دستگیر کنیم. ما انسان‌ها هر چقدر هم با یکدیگر متفاوت باشیم، الگوهای رفتاری مشابه زیادی بین ما وجود دارد که باعث می‌شود علم داده‌کاوی مهم و همینطور هیجان انگیز باشد. داده کاوی کمک می‌کند تا از بین دسته‌های داده‌ای نامنظم و بهم ریخته، الگویی پیدا کنید و داده‌هایتان را از هرج و مرج نجات دهید؛ مخصوصا وقتی دسته داده‌های کلانی دارید. داده کاوی ابزار خوبی برای مدیران زنجیره‌ی تامین و منابع انسانی است.

> **در شماره قبلی مجله، در مورد دیتا پرایوسی صحبت کردیم. همه ما می‌دانیم که باید قبل از اینکه چوب را ببریم تا قایقی را بسازیم، باید فیزیک قایق و چگونگی متعادل کردن آن روی آب را بدانیم. حال که از چگونگی حفظ حریم خصوصی داده‌ها اطلاع پیدا کردیم، می‌توانیم بالاخره کار با داده‌ها را شروع کنیم.**

تاریخچه داده کاوی

داده کاوی علم نسبتاً نوینی است تا جایی که تا قبل از ۱۹۹۰ حتی اسمی از آن نیز آورده نشده است. البته ما انسان‌ها همیشه در تاریخ مشغول این عمل به ظاهر ساده ولی مهم در روش‌های گوناگونی بوده‌ایم. هر تحلیل و پیش بینی که طبق متدهای آماری صورت گیرد، می‌تواند به عنوان کاوش داده در تلقی شود. گرگوری پیاتتسکی-شاپیرو در سال ۱۹۸۹ اصطلاح «کشف دانش در پایگاه‌های داده» (KDD) را ابداع کرد. با این حال اصطلاح «داده‌کاوی» در تجارت و مطبوعات رایج و محبوب‌تر شد.
این نمودار کمک می‌کند تا ببینید این علم چطور در طی زمان شکل گرفته است:



فرآیند داده‌کاوی می‌تواند برای هر مسئله، با توجه به ذات مسئله مراحل مختلفی را داشته باشد ولی بصورت کلی به ۴ زیرمرحله تقسیم می‌شود:

۱. جمع آوری: جمع آوری داده می‌تواند از منابع مختلف صورت گیرد ولی هماهنگ کردن داده‌هایی که از منابع مختلف جمع‌آوری شده باشند، کار سختی است. امروزه یکی از مهم‌ترین مسائلی است که یک دانشمند داده با آن دست و پنجه نرم می‌کند(handling multiple data source) این مسئله است.

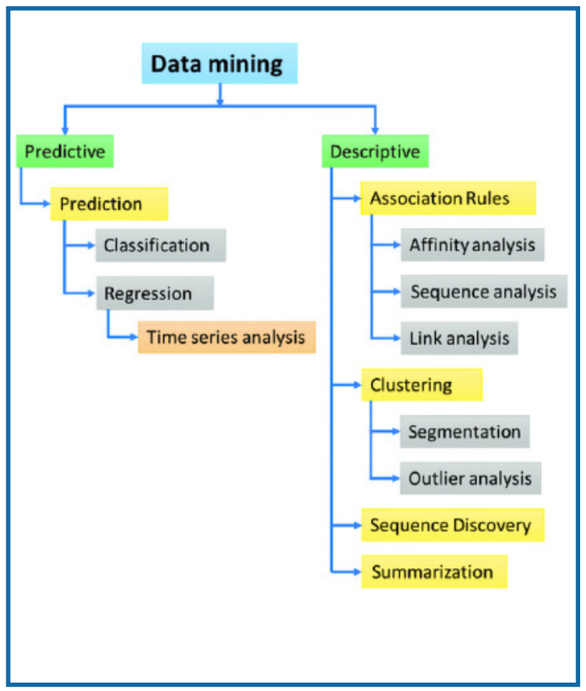
۲. آماده سازی: داده‌ها برای تحلیل باید پاکسازی شوند. ستون‌های خالی و اطلاعات بی‌فایده حذف و اصلاح شوند و اگر لازم بود، بخش‌های خالی با عددهای منطقی (اما نه واقعی) پر شود (در مواقعی که داده‌های بسیار کمی موجود و در اختیار ما است بسیار مفید واقع می‌شود) یا بصورت دستی مقادیری به داده اضافه یا کم شود. (مخصوصا اگه تفاوت واحد اندازه گیری وجود داشته باشه) این مراحل و ملاحظات همه داده‌ی را برای بررسی آماده می‌کنند.

۳. تجزیه و تحلیل بر پایه‌ی الگوریتم: هنگامی که داده‌ها آماده شدند، یک دانشمند داده، تکنیک داده‌کاوی مناسب را انتخاب و سپس یک یا چند الگوریتم را پیاده‌سازی می‌کند. به عنوان مثال، این تکنیک‌ها می‌توانند الگوها و همبستگی‌ها (correlations) را شناسایی کنند. در یادگیری ماشین، قبل از اینکه برنامه روی کل مجموعه داده (dataset) اجرا شود، معمولا الگوریتم‌ها بر روی مجموعه داده‌های نمونه train می‌شوند. (توضیح دادنِ نحوه‌ی کارِ این نوع از یادگیری ماشین، خود نیاز به مطلب جداگانه‌ای دارد)

۴. تفسیر و نمایش نتایج: نمایش نتایج بدست آمده و تفسیر آن‌ها بر اساس اهمیت، کار حیاتی است که توسط دانشمند داده به همراه چند نفر از مدیران اجرایی انجام می‌گیرد تا نتایج معنادارتر و مفیدتر شوند. همچنین باید از ابزارهای نمایش داده‌ای مناسبی بهره برد تا کارایی نتایج دوچندان شود.

در علم داده کاوی باید از چه ابزارهایی استفاده کنیم؟

در مورد مرحله سوم کار با داده یعنی تجزیه و تحلیل، ابزارها و الگوریتم‌هایی را معرفی کرده و صرفا نام می‌بریم. توضیح بقیه مراحل را به قسمت‌های دیگر موکول می‌کنیم.
این الگوریتم‌ها به دو بخش الگوریتم‌ها بر پایه‌ی توصیف (descriptive) و الگوریتم‌های پیش‌بینی کننده (predictive) تقسیم می‌شوند:
داده‌کاوی توصیفی، داده‌ها را برای ارائه آخرین اطلاعات در مورد رویدادهای گذشته تجزیه و تحلیل می‌کند، در حالی که داده کاوی پیش‌بینی‌کننده پاسخ پرس و جوهای آینده را ارائه می‌دهد. زبان‌های برنامه نویسی پایتون، R و SQL به ترتیب بیشترین میزان استفاده را در بین دانشمندان داده دارا هستند. برای جمع آوری و پاکسازی داده‌ها می‌توان از ابزارهایی چون MySQL و SQL، SQlite، MariaDB، PostgreSQL، اوراکل، cassandra، MongoDB بهره برد. همگی این ابزارها را می‌توان به جایِ یکدیگر استفاده کرد. (بسته به تعداد عملیات read-write، نوع دیتابیس و حجم داده)



این چارت‌ها می‌توانند به شما کمک کنند تا راحت‌تر بتوانید برای مسئله‌ی خود DBMS* پیدا کنید.

RELATIONAL VS NON-RELATIONAL DATABASES

Feature	Relational databases (SQL)	Non-relational databases (NoSQL)

Data structure	• Organize data into tables with rows and columns
	• Strict schema
	• Data resides in records and attributes
		• Use different data models like
		  ○ document-oriented,
		  ○ key-value,
		  ○ graph,
		  ○ wide-column
		• Store unstructured data
		• No fixed schema

Language	Structured Query Language (SQL)	Various query languages depending on the data model
Scalability	• Scale vertically (more computer power to a single server) • Horizontal scaling is challenging and requires additional effort	• Scale horizontally (add more servers) • Share data between servers, decreasing the request-per-second rate in each server
Performance	• Perform well with intensive read/write operations on small to medium datasets • Can suffer when data and user requests grow	• High performance with distributed design • Provide simultaneous access to a large number of users • Store unlimited data sets in various formats
Security	• The integrated structure provides better security • ACID compliance is preferred for applications where database integrity is critical	• Generally weaker security • ACID guarantees are often limited to a single database partition • Some DBMSs offer advanced security features for compliance.
Use cases	Complex software solutions, eCommerce, financial applications	Storing and scaling unstructured data, MVPs for startups, sprint-based Agile development

پس یادگیری، آنها را هر چه سریع‌تر استارت بزنید و جلو بروید :)

DATABASE MANAGEMENT SYSTEMS COMPARISON

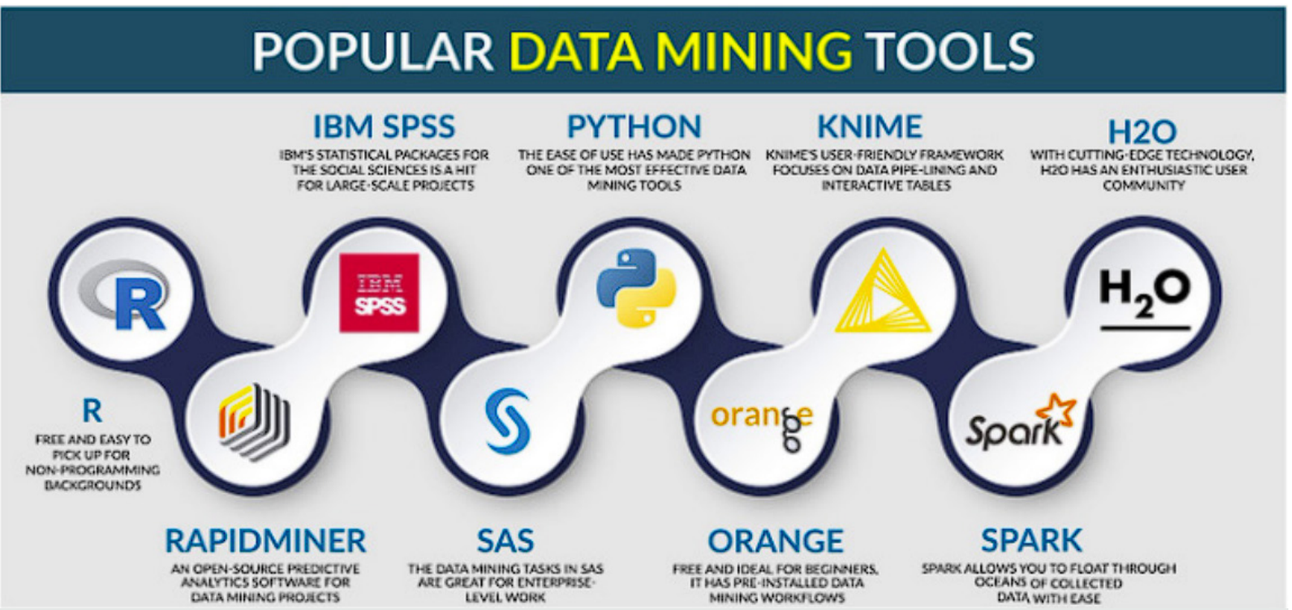
	Database Type	Licensing	Scalability	Data Types Supported	Learning Curve
MySQL	SQL	GNU Generally Public License	Vertical, complex	Structured, semi-structured	Mild
MariaDB	SQL	GNU Generally Public License	Vertical	Structured, semi-structured	Mild
Oracle	Multi-model, SQL	Proprietary	Both (Vertical & Horizontal)	Structured, semi-structured, unstructured	Hard
PostgreSQL	Object-relational, SQL	Open-source	Vertical	Structured, semi-structured, unstructured	Hard
MSSQL	T-SQL	Proprietary	Vertical, complex	Structured, semi-structured, unstructured	Hard
SQLite	SQL	Public domain	Vertical	Structured, semi-structured, unstructured	Mild
MongoDB	NoSQL, document-oriented	SSPL	Horizontal	Structured, semi-structured, unstructured	Mild
Redis	NoSQL, key-value	Open-source, BSD 3-clause	Horizontal	Structured, semi-structured, unstructured	Mild
Cassandra	NoSQL, wide-column	Open-source	Horizontal	Structured, semi-structured, unstructured	Hard
Elasticsearch	NoSQL, document-oriented	Open-source	Horizontal	Structured, semi-structured, unstructured	Hard
Firebase	NoSQL, real-time database	Open-source	Horizontal	Structured, semi-structured, unstructured	Mild
Amazon DynamoDB	NoSQL, key-value	Proprietary	Horizontal	Structured, semi-structured, unstructured	Mild

مزایا: مزایای استفاده از داده‌کاوی در هر زمینه‌ی علوم کامپیوتر، آنقدر مشهود است که این بخش صرفا به بیان بدیهیات می‌پردازد. داده کاوی در امنیت و شبکه، به متخصص در تشخیص بدافزار، نفوذ سیستم و شبکه، حملات خودی و بسیاری از تهدیدات امنیتی دیگر کمک می‌کند. به طراح سایت و توسعه دهنده نرم‌افزار این قابلیت را می‌دهد تا فروش را به کمک پیش‌بینی رفتار مشتری بالا ببرد و تضمین کند که اطلاعات مفید را می توان از داده های خام استخراج کرد و از آن برای منافع سازمان و مشتریانش استفاده کرد. متخصصان هوش مصنوعی به کمک داده‌کاوی، می‌توانند داده‌ها را سریع‌تر از روش‌های دستی پردازش کنند و درخواست‌ها را در مقیاس بزرگ، به طور موثر مدیریت کنند. در صنعت بازی سازی، توسعه‌دهندگان می‌توانند از این داده‌ها برای شناسایی الگوها و اولویت‌ها استفاده کنند، و آنها را قادر می‌سازد تا مکانیک بازی را دقیق تنظیم کنند، سطح دشواری را متعادل نموده و رضایت کلی بازیکنان را بهبود بخشند. در هر صورت داده کاوی دانستن به پیشبرد اهداف شرکت ها کمک می‌کند.

معایب: اولین و مهمترین دغدغه‌ی دانشمندان داده در استفاده از متدهای داده‌کاوی، امنیت داده است. مشکل بعدی در کمبود دقت(underfit) یا دقت بیش از حد خرج کردن در کار با داده‌های جمع آوری شده است(overfit). همچنین مراحل ابتدایی داده‌کاوی ممکن است هزینه‌بر باشند.

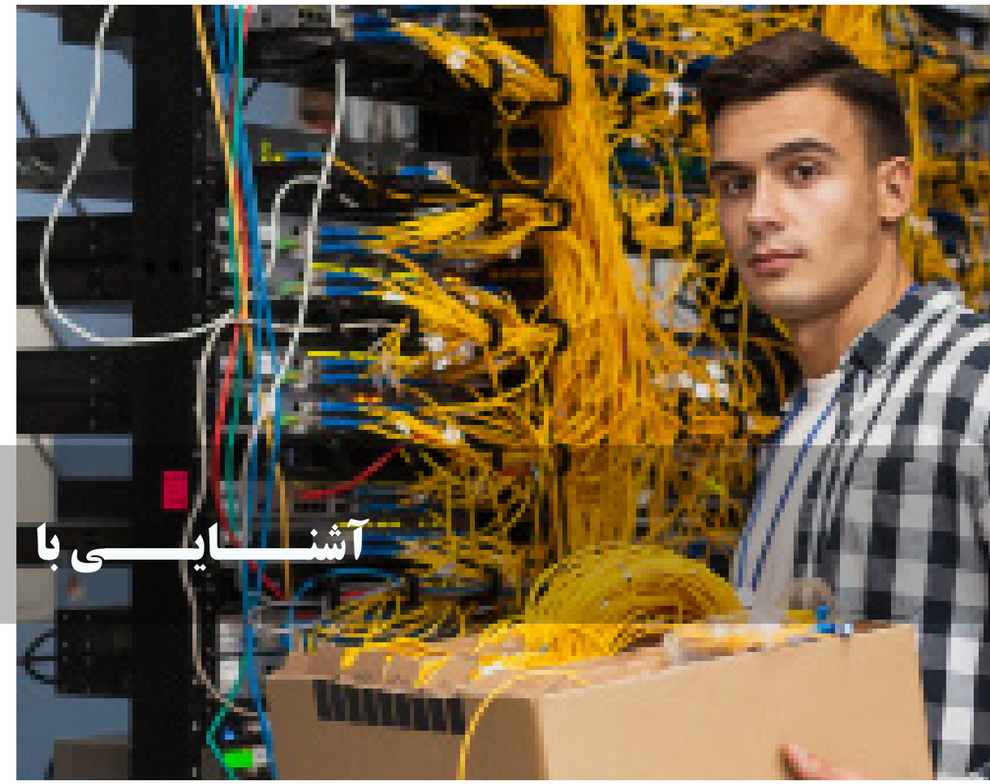
ابزار ماشین لرنینگ

برای قسمت یادگیری ماشین، این ابزارها امروزه بیشترین استفاده را دارند که با توجه به متن‌باز بودن یا نبودن، مسئله و حجم داده، میزان تخصص شما، بودجه تخصیص داده شده و پارامترهای دیگر، باید دائما در جست و جوی مورد مناسب و یادگیری باشید.



نویسنده: سینا طیبی

آشنایی با DHCP



به هر هاست موجود در شبکه باید یک آدرس IP اختصاص داده شود چه به صورت دستی و چه به صورت خودکار! معمولا در شبکه‌های خانگی که ۲ یا ۳ کامپیوتر و تلفن همراه وجود دارد، به راحتی می‌توان یک آدرس IP به آن دستگاه‌ها اختصاص داد. حالا تصور کنید که یک شبکه بزرگ وجود دارد که هزاران کامپیوتر به آن متصل هستند، در نتیجه تنظیم دستی همه این IPها برای سیستم زمان خیلی زیادی می‌برد و ممکن است برای یک شبکه کار به یک کابوس تبدیل شود. هیچ دو سروری نمی‌توانند آدرس IP یکسانی را برای مشتری‌هایشان داشته باشند و اگر به‌صورت دستی دو آدرس رو شبیه هم کنید، باعث ایجاد مشکلات متعددی می‌شود.

و اما راه حل این مشکل چیست ؟

Dynamic Host Configuration Protocol یا همان DHCP، اگر کمی در مورد شبکه‌های محلی و جهانی خوانده و یا کمی در تنظیمات ویندوز کامپیوتر خود سرک کشیده باشید قطعا تا به حال به کلمه DHCP برخورده کرده‌اید. به صورت خلاصه عملکرد DHCP بدین صورت است که در آن پروتکل‌هایی برای تخصیص پارامترهای پیکربندی TCP/IP به کامپیوترها مورد استفاده قرار میگیرد.حال که کمی با این پروتکل آشنا شدید، بیایید کمی جزئی تر به آن نگاه کنیم.

DHCP چیست ؟

DHCP مخفف Dynamic Host Configuration Protocol و به معنای "پروتکل پیکربندی پویای میزبان" است. این پروتکل قادر است سرور را فعال کند تا به طور خودکار به کامپیوتری که در محدوده اعداد مشخص‌شده برای کانفیگ شبکه است ، آدرس IP اختصاص دهد. علاوه بر این، DHCP یک وظیفه مهم دیگر نیز دارد. مدیریت پیکربندی شبکه برای default ، subnet mask Gateway و همچنین سرویس DNS.

اجازه دهید عملکرد DHCP را برای شما با یک مثال توضیح دهم:

شبکه کامپیوتری خود را شهری در نظر بگیرید و DHCP را اداره پست در نظر بگیرید که آدرس های پستی را برای مدت معینی اجاره می دهد. در بحث شبکه، این آدرس‌های پستی همان آدرس‌های IP اختصاص یافته به هر دستگاه متصل به LAN شما هستند. هر کدام از این دستگاه‌ها، دارای یک آدرس منحصر به فرد است که بتواند داده‌های مختلفی را دریافت و ارسال کنند. زمانی که فردی به تازگی وارد این شهر می‌شود، به یک آدرس پستی اختصاصی نیاز دارد تا بتواند با دوستان و خانواده خود در شهرهای دیگر ارتباط برقرار کند. مسلماً اگر آدرس پستی اختصاص یافته به این فرد با فرد دیگری در این شهر یکسان باشد، پست‌های ارسالی نیز با مشکل مواجه خواهند شد؛ پس اگر یک سازمان بخواهد این عمل اختصاص دادن آدرس‌های پستی را دستی انجام دهد، باید چندین نفر را استخدام کرده و با صرف زمان و هزینه بسیار، امور را پیش ببرد. اما بجای این کار می‌تواند از یک پروتکل یا هر ابزار دیگری استفاده کند که با افزوده شدن افراد جدید به این شهر، آن پروتکل به طور پویا و خودکار برای آنها آدرس پستی منحصر به فرد تولید کند که برای این کار باید تمام آدرس‌های اختصاص یافته را در لیست خود چک کند تا تکراری نباشد. از مزایای این روش این است که اگر فردی به طور موقت وارد شهر شده و بخواهد برای چند ماه آدرس پستی اجاره کند، سیستم به طور خودکار پس از خروج فرد از شهر، آدرس را از وی گرفته و به دیگری می‌دهد. کل فرآیندی که توضیح داده شد توسط DHCP انجام می‌گیرد اما بجای شهر و آدرس پستی، در شبکه و با آدرس های IP محقق می‌شود.DHCP ابزاری مفید و کاربردی است که در صرف زمان و هزینه‌های شما صرفه جویی به عمل آورده و امنیت را افزایش می‌دهد.

حال که به صورت کامل به عملکرد DHCP واقف شدید بیایید به سراغ بخش های مختلف تشکیل دهنده DHCP برویم:

سرور DHCP: یک سرور DHCP توانایی این را دارد که بسته به نیاز یک سرور، کامپیوتر و یا روتر باشد که اطلاعات پیکربندی شبکه از جمله آدرس‌های IP را مدیریت می‌کند.

DHCP client: یک کلاینت DHCP در اصل دستگاهی درون شبکه است که با سرور DHCP در تعامل بوده و اطلاعات پیکربندی را از آن دریافت می‌کند.

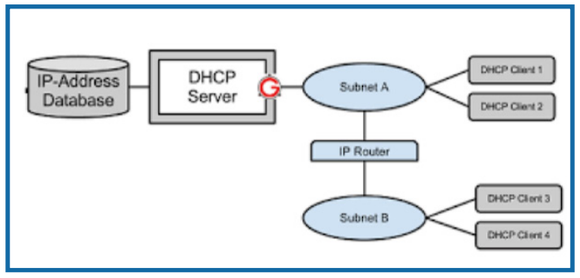
DHCP relay agent: دستیار DHCP یک هاست یا روتر است که درخواست‌های بین کلاینت‌های DHCP درونی و سرور از راه دور را ارسال کرده و به آنها پاسخ می‌دهد. زمانی که تنها یک سرور DHCP برای چندین LAN در دست باشد، این دستیار تمام درخواست‌های شبکه را مدیریت می‌کند.

آدرس دروازه پیش فرض (Default gateway address): یک دروازه پیش فرض، که به آدرس gateway نیز معروف است، گره‌ای است که اطلاعات را بین شبکه‌های لوکال یا ساب‌نت‌های داخلی و اینترنت جا به جا می‌کند.

استخر آدرس (IP address pool): منظور لیستی از تمام آی‌پی‌هایی است که برای توزیع آماده و در دسترس هستند.

زمان اجاره (Lease time): این زمان نشان دهنده دوره‌ای است که یک کلاینت می تواند در آن بازه از آدرس IPای که به آن اختصاص داده شده، استفاده کند.

منظور از زمان اجاره زمانی است که یک سرور DHCP پس از عضویت یک دستگاه به شبکه، آدرس IP خاصی را به آن اختصاص می‌دهد، آن دستگاه می‌تواند برای یک مدت معین از آدرس استفاده کند. این مدت زمان به مدت زمان اجاره یا DHCP lease معروف است. اگر کلاینت بخواهد که اجاره IP خود را تمدید کند، باید یک درخواست به سرور ارسال کند؛ در غیر این صورت، سرور پس از اتمام مدت زمان اجاره، یک فرآیند جداسازی آدرس IP را پیش خواهد گرفت.



چطور عمل می‌کند؟

زمان اختصاص دادن آدرس های IP به دستگاه‌های درون شبکه، پروتکل پیکربندی خودکار هاست (DHCP) چند مرحله اجرایی را طی می‌کند. این مراحل به DORA شناخته می‌شوند، که خلاصه‌ای از Discovery, offer, request, acknowledgment می باشد. بنابراین این پروسه شامل، کشف، پیشنهاد، درخواست و تاییدیه می باشد. بگذارید این مراحل را کمی بیشتر بررسی کنیم...

◆ Server discovery یا یافتن سرور

کلاینت قبل از دسترسی به خدمات شبکه، ابتدا پیغام‌های DHCP را ارسال کرده تا سرورهای معتبر DHCP را پیدا و کشف کند. اگر یک کلاینت و سرورهای DHCP بر روی LANهای متفاوتی باشند، مشاور یا دستیار DHCP پیغام‌هایی را رد و بدل می‌کند تا ارتباط را هموار کند.

IP address lease offer **یا پیشنهاد اجاره آدرس** IP

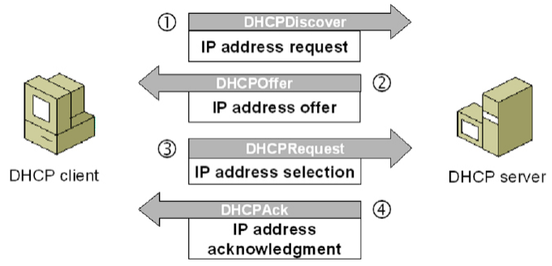
DHCP به پیغام کشف سرور پاسخ داده و یک پیام پیشنهادی پخش می‌کند. این پیام پیشنهادی شامل اطلاعات پیکربندی مثل آدرس MAC کلاینت، آدرس IP پیشنهادی، Subnet mask، آدرس IP دروازه پیش فرض، آدرس IP سرور DNS، مدت زمان اجاره و آدرس IP سرور DHCP خواهد بود. تمام این اصطلاحات را در بالا معرفی و واژگان انگلیسی آنها را ارائه کردیم. (توصیه می‌کنیم حتما با اصطلاحات انگلیسی آشنا باشید)

IP lease request **یا درخواست اجاره** IP

پس از آنکه کلاینت پیام پیشنهادی شامل اطلاعات پیکربندی را دریافت کرد، یک پیغام درخواست DHCP برای آدرس IP پیشنهاد شده ارسال می‌کند. اگر کلاینت با پیشنهادهای متفاوتی از چندین سرور DHCP مواجه شده باشد، تنها یکی از آنها را خواهد پذیرفت. سپس کلاینت یک درخواست Address Resolution Protocol یا به اصطلاح ARP تستی ارسال کرده تا ببیند که آیا هیچ هاست دیگری از این آدرس IP استفاده می‌کند یا نه! در ادامه کلاینت باید مشخص کند که کدام پیشنهاد را انتخاب کرده و به سایرین اطلاع دهد که پیشنهادشان را پس گیرند.

IP lease acknowledgment **یا تاییده اجاره** IP

در آخرین مرحله، DHCP به درخواست پاسخ داده و اطلاعات آدرس IP را به کلاینت تحویل می‌دهد. سپس کلاینت پیکربندی های لازم را انجام داده و می تواند با آدرس IP خاص خود به اینترنت متصل شود.



مزایای DHCP

DHCP مزایای بسیاری دارد، این پروتکل پویا می تواند:

مدیریت IP ها را ارتقا دهد.

از تداخلات بین IP ها جلوگیری کند.

اثربخشی را افزایش دهد.

در هزینه و زمان صرفه جویی به عمل آورد.

و اما عیب بزرگ DHCP

با اینکه DHCP با تغییر پویای آدرس‌های IP، امنیت را ارتقا می‌دهد، لازم است بدانید که خود این پروتکل دارای نقطه ضعف امنیتی است. این پروتکل به دستگاه‌های جدید اجازه می‌دهد تا بدون گذر از فرآیندهای اعتبار سنجی وارد شبکه شوند. به عبارت دیگر، اطمینان از امنیت سرور DHCP در شبکه مشخصی که یک دستگاه به آن متصل می‌شود، می‌تواند کار دشواری باشد. این موضوع احتمال وقوع مشکلات امنیتی را افزایش می‌دهد. برای مثال، نبود اعتبارسنجی می‌تواند منجر به خالی شدن ظرف IPها شود، چرا که دستگاه‌های غیر مجاز نیز می‌توانند به آدرس IP مجهز شوند. به علاوه، DHCP در معرض حملات سایبری DoS یا MITM است.

یکی از اصلی‌ترین دلایلی که ادمین‌های شبکه از DHCP استفاده می‌کنند، مدیریت آسان‌تر آدرس‌های IP است. زمانی که یک دستگاه جدید وارد شبکه می‌شود، این DHCP به طور خودکار یک آدرس آی پی به آن اختصاص می‌دهد، این بدین معناست که دیگر نیازی نیست تا ادمین شبکه برای هر دستگاه، به طور دستی موارد پیکربندی را اجرا کند. این موضوع باعث عدم اتلاف زمان بخصوص برای سازمان‌های بزرگ‌تر شده و از آنجایی که به متخصص اضافی شبکه برای پیشبرد این امور نیازی نخواهد بود، در هزینه‌های سازمان نیز صرفه جویی خواهد شد. بعلاوه زمانی که این امور را بطور دستی پیکربندی کنید، احتمال وقوع تداخل و اشتباهات نیز وجود خواهد داشت. ممکن است به اشتباه یک آدرس IP که قبلا اختصاص یافته را به دستگاه جدیدی تحویل دهید، که همانطور که گفته شد هیچ دو هاستی نمی‌توانند از یک آدرس IP یکسان استفاده کنند. اما به لطف DHCP درخواست‌هایی برای حصول اطمینان از عدم وجود conflict یا تعارض بین IPها توسط این پروتکل ارسال خواهند شد. علاوه بر این DHCP می‌تواند آدرس‌های IP داینامیک را اجرا کند؛ پس اگر در آدرس IP خاصی تغییری ایجاد شود و یا دیگر نخواهید از آن استفاده کنید، DHCP به طور خودکار آن را تغییر می‌دهد؛ به همین دلیل است که امنیت این پروتکل از حالت استاتیک بالاتر است. با اینکه بیشتر روترها و کامپیوترهای مدرن دارای قابلیت DHCP به صورت پیش فرض هستند، برخی از سازمان‌ها تمایل دارند که از یک سرور DHCP اختصاصی استفاده نمایند. این انتخاب می‌تواند به دلیل تخصیص بهتر آی پی‌ها و افزایش ویژگی شخصی سازی باشد. به این سرورهای DHCP اختصاصی، معمولاً Centralized DHCP server یا سرور DHCP متمرکز گویند. البته برای استفاده از اینترنت شبکه خانگی، سرور DHCP نیازی نیست.

آشنایی با Node-red

آیا تا حالا شده که ایده‌ای با قابلیت اجرای آسان به ذهنتان برسد و نیاز به دانش فنی سمت سرور و حتی سمت کاربر داشته باشید اما یادگیری این دانش‌ها زمان از شما ببرد و به همین دلیل از خیر کل ایده و پروژه بگذرید؟ اینجاست که Node-red برای شما جادو می‌کند. یک ابزار برنامه‌نویسی محبوب در حوزه اینترنت اشیا است که کاربرد آن صرفا به این حوزه محدود نشده و به راحتی می‌توان از آن در جاهای مختلفی استفاده نمود.

چگونگی استفاده

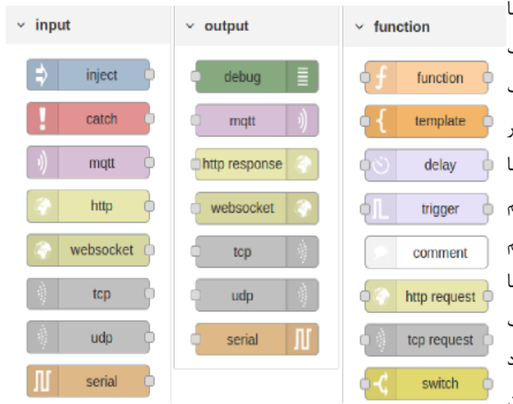
این ابزار شامل یک محیط گرافیکی تحت وب است که فرآیند تولید و طراحی را با نمودار و فلوچارت آسان می‌کند. در صورتی که نمودارها نیز کافی نباشد می‌توان با استفاده از زبان جاوا اسکریپت درون محیط آن کد نویسی انجام دهید. به دلیل متن باز بودن، جامعه توسعه دهندگان آن رقم قابل توجهی است و در صورت نیاز می‌توان در فروم‌های مختلفی درخواست کمک از دیگران کرد.

محیط

برای شروع نیاز است یک گره (node) ایجاد کنید که از پنل کناری یا با کلیک راست روی صفحه می‌توان به سادگی تمام این کار را انجام داد. شاید بپرسید چرا باید یک گره ایجاد کنم؟ واضح است همانطور که در فلوچارت با استفاده از مستطیل لوزی دایره و المان‌های دیگر طراحی خود را انجام می‌دادید اینجا نیز با استفاده از گره‌ها الگوریتم خود را پیاده سازی می‌کنید.

گره‌ها – node

بطور کلی سه دسته گره در این محیط وجود دارد: گره‌های آغازین، میانی و پایانی که هر کدام کاربرد خود را دارند؛ گره‌های آغازین معمولا منتظر دریافت دستور یا رخداد (event) هستند که این خود می‌تواند یک درخواست شبکه یا یک تغییر در ساختار سیستم باشد. برای مثال اگر یک دستگاه آردوینو به آن متصل کنید هر چیزی که در سریال چاپ شود می‌تواند یک رخداد باشد تا شما با بررسی آن رخداد، ادامه الگوریتمتان را طراحی کنید یا قادر هستید که یک تونل وب سوکت ایجاد و با سیستم دیگری در ارتباط باشید و به راحتی با دستورهای از پیش تعیین شده به سرور خود دستور دهید. حال وقت آن است که پردازشی روی رخ داد دریافت شده صورت بگیرد می تواند تغییر در اطلاعات دریافت شده صورت بگیرد یا حتی داده جدیدی با توجه به داده دریافت شده تولید شود، در این مرحله شما می توانید از توابع از پیش نوشته شده استفاده کنید یا خودتان با جاوا اسکریپت یک تابع جدید طراحی کنید؛ منطق پشت این سیستم به این صورت است که شما یک بسته دریافت می کنید و در هر مرحله بلایی به سر آن می‌آورید، این بسته‌ها در این سیستم msg یا پیام نامیده می‌شود که یک شی یا obj است. معمولا بسته‌ها شامل یک سری اطلاعات (payload) و یک موضوع (topic) هستند. البته می توانیم از نام های دیگری هم استفاده نماییم. در واقع هر مقداری که دوست داشته باشیم میتوانیم شخصی سازی‌های دلخواه خود را اعمال کنیم. در مرحله پایانی می‌توان با گره‌های پایانی، دیتای پردازش شده را در خروجی یک کنسول و یا در یک صفحه وب و هر جای دیگری قابل نمایش خواهد بود که این خود کاملا وابسته به گره مورد استفاده شما دارد، در تصویر روبرو می توانید به ترتیب از راست به چپ گره آغازین (ورودی)، پایانی (خروجی) و گره میانی (توابع) را مشاهده کنید.



msg ها

بسته‌هایی که در این سیستم در حال جابجایی می‌باشند، همگی یک نوع شی هستند و دارای ویژگی‌های خاص خود هستند. در واقع ما با آنها مثل یک فایل json برخورد می‌کنیم و وظیفه انتقال اطلاعات بین node ها را به عهده آنها می‌گذاریم. از آنجایی که جامعه این پلتفرم جامعه بسیار وسیعی می‌باشد، کتابخانه‌های متعددی نیز برای آن توسعه داده شده است که هر کدام کاربرد مخصوص خود را دارند و هر کدام قابلیتی به سیستم می‌افزایند. برای مثال برای ارتباط با "پورت سریال برد آردوینوی" خود روی سیستم شخصی باید کتابخانه node-red-node-serialport را از منوی همبرگر سمت راست بالای صفحه قسمت manage palette نصب کنید.

چگونه node–red را نصب کنیم؟

همانطور که می دانید همیشه بهترین منبع برای شروع، وبسایت رسمی توسعه دهنده اصلی آن محصول است. برای رفتن به این وبسایت باید به آدرس nodered.org در برویم. پس از ورود به سایت از منوی documentation و سپس get started می‌توان سیستم‌هایی که این نرم‌افزار از آن ساپورت می‌کند را مشاهده کرد. پیشنهاد می‌شود در صورتی که این نرم‌افزار را برای کارهای تحقیقاتی استفاده نمی‌کنید، آن را بر روی یک سرور لینوکس نصب و اجرا کنید تا همیشه در دسترس باشد. همچنین می‌توان این نرم‌افزار را بر روی برد رزبری پای، تلفن‌های همراه و خیلی از محیط‌های دیگری که در داخل سایت ذکر شده، نصب کرد. از آن جایی که این نرم‌افزار تحت مرورگر وب و تنها بر روی یک پورت اجرا می‌شود، با تمام دستگاه‌های متصل به شبکه دستگاهی که روی آن نصب شده است، می‌توان از آن استفاده نمود.

نصب روی سیستم خانگی

تقریبا از اسم نرم افزار نیز می توان حدس زد که محیط اجرای آن node js می‌باشد برای همین قبل از هر اقدامی نیاز است تا ابتدا node js را بر روی سیستم خود نصب کنید که از طریق nodejs.org می‌توان نسخه مربوط به سیستم عامل مورد نیاز را پیدا و نصب نمود. سپس با سیستم مدیریت بسته (package manager) معروف node js یعنی npm می توانید با دستورات زیر نصب

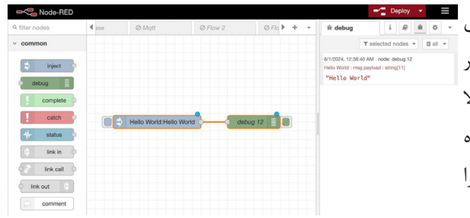
```
npm install node-red
```

و سپس اجرا کرد:

```
node-red
```

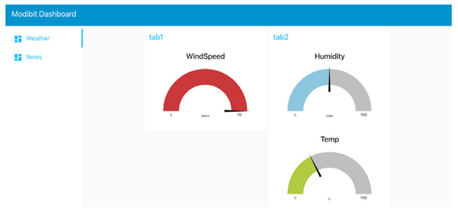
اجرا (hello world)

پس از اجرا با دستور node-red داخل ترمینال حال از طریق آدرس ۱۲۷.۰.۰.۱:۱۸۸۰ می توانید وارد محیط نود رد شوید سمت چپ گره‌ها یا نودهارا مشاهده می‌کنید و سمت راست نیز ابزارها و محیط کنسول و دیباگ برای شروع می توانید یک گره inject به وسط صفحه بکشید و یک گره debug روبروی آن و به یکدیگر متصل کنید حال داخل گره inject پیامی وارد کنید (در payload) و از سمت راست بالا deploy کنید. حال هر وقت روی inject ضربه بزنید پیام شما در محیط دیباگ نمایش داده می‌شود. تبریک! شما اولین طراحی خود را با این نرم افزار انجام دادید. البته راه درازی را در پیش دارید...

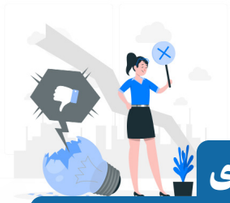


طراحی frontend

در این سیستم فوق العاده حتی به محیط product نیز فکر شده و شما می توانید یک سرویس تجاری روی آن پیاده سازی و برای کاربران خود پنل طراحی کنید. برای اینکار ابتدا باید کتابخانه dashboard را نصب و سپس با استفاده از گره‌های آن اقدام به طراحی پنل کاربری کنید. برای مثال در این تصویر برای چند ماژول دما سرعت باد و میزان رطوبت هوا شمارنده هایی برای نمایش به کاربر تعبیه شده. با خلاقیت و دانش بیشتر می‌توان یک اپلیکیشن کامل را نیز طراحی نمود. این داشبورد با استفاده از فریم ورک انگولار توسعه داده شده و قابلیت‌های زیادی در آن وجود دارد. همانطور که گفته شد می‌توان با نصب کتاب خانه‌های بیشتر، از قابلیت‌های بی‌شماری برخوردار شد. امیدواریم که تا اینجای کار این معرفی و آموزش ابتدایی براتون مفید بوده باشه و یادتون نره که حتما دست به کد بشید، شاید روزی با توجه به پتانسیل‌های بیشمار این حوزه، :)



نویسنده: محمد رضا لله گانی



غول‌های فناوری و پروژه‌های شکست خورده

شرکت‌های بزرگ فناوری اغلب با محصولات اصلی و شناخته‌شده‌شان مانند گوشی‌های هوشمند، سیستم عامل‌ها و سرویس‌های ابری به یاد آورده می‌شوند. با این حال، این شرکت‌ها گاهی محصولات غیر عادی و غیر متعارفی نیز توسعه می‌دهند که شاید کمتر به گوش رسیده باشند. در این مقاله به بررسی محصولات خاص و غیر عادی از ۸ کمپانی بزرگ فناوری شامل اپل، گوگل، مایکروسافت، آمازون، سامسونگ، فیسبوک، تسلا و هواوی خواهیم پرداخت.

اپل نیوتن (Apple Newton)

یکی از اولین دستگاه‌های دیجیتال دستیار شخصی (PDA) بود که توسط شرکت اپل در سال ۱۹۹۳ معرفی شد. این دستگاه که به‌طور رسمی با نام «Newton MessagePad» شناخته می‌شد، یکی از پیشروان در زمینه دستگاه‌های همراه هوشمند به‌شمار می‌رفت و برخی از ویژگی‌هایی که بعدها در گوشی‌های هوشمند و تبلت‌ها رایج شدند را برای اولین بار معرفی کرد.

ویژگی‌ها و نوآوری‌ها: تشخیص دست‌نوشته: یکی از برجسته‌ترین و منحصربه‌فردترین قابلیت‌های نیوتن، سیستم تشخیص دست‌نوشته بود که به کاربر امکان می‌داد نوشته‌های دستی خود را به متن دیجیتال تبدیل کند. هرچند این ویژگی در ابتدا بسیار مورد توجه قرار گرفت، اما به دلیل دقت پایین و ناپایداری عملکرد، با انتقاداتی مواجه شد.

قابلیت‌های سازماندهی: نیوتن می‌توانست کارهایی مانند مدیریت مخاطبین، تقویم، یادداشت‌برداری و لیست کارها را انجام دهد. این ویژگی‌ها به کاربران کمک می‌کرد تا بتوانند امور روزمره خود را به صورت دیجیتال سازماندهی کنند.

اتصالات و گسترش‌پذیری: نیوتن دارای درگاه‌های اتصال به دستگاه‌های دیگر و همچنین قابلیت گسترش از طریق کارت‌های حافظه بود.

دلایل عدم موفقیت: یکی از عوامل مهم در عدم موفقیت نیوتن، قیمت بالای آن بود که باعث می‌شد برای بسیاری از کاربران غیرقابل دسترس باشد. دقت پایین و عدم ثبات در تشخیص دست‌نوشته، یکی از نقاط ضعف اصلی نیوتن بود که بر تجربه کاربری تأثیر منفی گذاشت. در زمان معرفی نیوتن، بازار PDAها به سرعت در حال توسعه بود و دستگاه‌های مشابه دیگری با قیمت پایین‌تر و عملکرد بهتر به بازار عرضه می‌شدند. در نهایت، اپل در سال ۱۹۹۸ تولید نیوتن را متوقف کرد.

اگرچه نیوتن به عنوان یک پروژه تجاری موفقیت‌آمیز نبود، اما ایده‌ها و فناوری‌های آن در توسعه محصولات بعدی اپل و همچنین دیگر دستگاه‌های دستی هوشمند تأثیرگذار بود.

اپل پیپین (Apple Pippin)

اپل پیپین (Apple Pippin) یک کنسول بازی و کامپیوتر چندرسانه‌ای بود که در دهه ۱۹۹۰ توسط اپل و با همکاری شرکت ژاپنی بانـدای (Bandai) تولید شد. این دستگاه که در سال ۱۹۹۶ به بازار عرضه شد، تلاشی از سوی اپل برای ورود به بازار کنسول‌های بازی و سیستم‌های چندرسانه‌ای خانگی بود.

ویژگی‌ها و قابلیت‌ها: سیستم عامل: پیپین از نسخه‌ای تعدیل شده از سیستم عامل Mac OS استفاده می‌کرد که آن را به نوعی کامپیوتر خانگی ساده‌شده تبدیل می‌کرد. کاربران می‌توانستند بازی‌ها و نرم‌افزارهای چندرسانه‌ای را از طریق CD-ROM اجرا کنند.

پلتفرم باز: اپل پیپین به عنوان یک پلتفرم باز طراحی شده بود که توسعه‌دهندگان و تولیدکنندگان دیگر می‌توانستند نرم‌افزارها و محتواهای خود را برای آن تولید کنند. این ویژگی در تلاش بود تا محیطی باز و انعطاف‌پذیر برای کاربران و توسعه‌دهندگان ایجاد کند.

اتصالات: پیپین به اینترنت متصل می‌شد و قابلیت پخش محتواهای چندرسانه‌ای داشت. همچنین دارای پورت‌های مختلفی برای اتصال به دستگاه‌های جانبی مانند کیبورد و ماوس بود.

دلایل عدم موفقیت: رقابت شدید: در زمان عرضه پیپین، بازار کنسول‌های بازی به شدت تحت تسلط شرکت‌هایی مانند سونی (با پلی‌استیشن)، نینتندو (با نینتندو ۶۴) و سگا (با سگا ساترن) بود. این رقابت شدید باعث شد که پیپین نتواند جایگاه مناسبی در بازار پیدا کند.

قیمت بالا: پیپین با قیمتی در حدود ۶۰۰ دلار به بازار عرضه شد که نسبت به رقبا بسیار گران‌تر بود و این موضوع باعث شد تا نتواند مخاطبان زیادی جذب کند.

محدودیت محتوا: تعداد کمی از بازی‌ها و نرم‌افزارهای کاربردی برای پیپین توسعه داده شد که این موضوع نیز به کاهش جذابیت آن در بازار منجر شد.

با وجود تلاش‌های اپل و بانـدای، اپل پیپین نتوانست موفقیت قابل توجهی در بازار کسب کند و تولید آن در سال ۱۹۹۷ متوقف شد. تنها حدود ۴۲ هزار واحد از این دستگاه به فروش رفت. اگرچه پیپین به عنوان یک کنسول بازی موفق نبود، اما ایده‌ها و تجربیاتی که از این پروژه حاصل شد، به نوآوری‌های بعدی اپل در زمینه‌های دیگر کمک کرد.

گوگل گلس (Google Glass)

یک عینک هوشمند پیشرو بود که توسط گوگل در سال ۲۰۱۳ معرفی شد. این دستگاه به عنوان یکی از اولین تلاش‌ها برای ترکیب فناوری‌های پوشیدنی با واقعیت افزوده (AR) شناخته می‌شود. گوگل گلس به کاربران امکان می‌داد تا اطلاعات دیجیتال را بدون نیاز به استفاده از دست، در مقابل چشمان خود مشاهده کنند.

ویژگی‌ها و قابلیت‌ها: صفحه نمایش کوچک:** گوگل گلس دارای یک نمایشگر کوچک شفاف بود که در گوشه بالای راست عینک قرار داشت. این نمایشگر می‌توانست اطلاعاتی مانند نقشه‌ها، پیام‌ها، و نوتیفیکیشن‌ها را در دید کاربر نمایش دهد. کاربران می‌توانستند از طریق فرمان‌های صوتی، مانند گفتن «OK Glass»، دستگاه را کنترل کرده و از قابلیت‌های مختلف آن استفاده کنند. گوگل گلس دارای یک دوربین ۵ مگاپیکسلی بود که امکان گرفتن عکس و ضبط ویدئو را فراهم می‌کرد. این دستگاه به اینترنت متصل می‌شد و می‌توانست از طریق وای‌فای یا بلوتوث به گوشی هوشمند کاربر متصل شود. یکی از قابلیت‌های برجسته گوگل گلس، امکان استفاده از نقشه‌ها و راهنمای مسیر بود که به کاربران در پیدا کردن مسیر کمک می‌کرد.

چالش‌ها و موانع: یکی از بزرگ‌ترین موانع برای پذیر ش عمومی گوگل گلس، نگرانی‌های مربوط به حریم خصوصی بود. وجود دوربین در عینک و قابلیت ضبط بدون جلب توجه، باعث شد تا افراد زیادی احساس نگرانی کنند. گوگل گلس با قیمتی در حدود ۱۵۰۰ دلار به بازار عرضه شد که برای بسیاری از کاربران معمولی، بسیار گران‌قیمت بود. با وجود ویژگی‌های جالب، گوگل گلس به دلیل محدودیت‌های سخت‌افزاری و نرم‌افزاری نتوانست تجربه کاربری کاملی ارائه دهد.

در نهایت در سال ۲۰۲۱، گوگل تصمیم گرفت تا پروژه گوگل گلس را به طور کامل متوقف کند و تمرکز خود را بر روی پروژه‌های دیگر واقعیت افزوده و فناوری‌های پوشیدنی بگذارد. با این حال، گوگل گلس به عنوان یکی از نخستین نمونه‌های فناوری پوشیدنی، تأثیر مهمی بر روی تحقیقات و توسعه‌های آینده در این حوزه داشت.

پروژه گوگل لوون (Google Loon)

یکی از پروژه‌های نوآورانه و بلندپروازانه شرکت آلفابت (شرکت مادر گوگل) بود که در سال ۲۰۱۳ به عنوان بخشی از پروژه‌های تحقیق و توسعه گوگل راه‌اندازی شد. هدف اصلی این پروژه ارائه اینترنت به مناطق دورافتاده و کم‌دسترس با استفاده از بالن‌های بالای جوی (stratospheric balloons) بود. این بالن‌ها که در ارتفاع حدود ۱۸ تا ۲۵ کیلومتری از سطح زمین پرواز می‌کردند، مجهز به تجهیزات ارتباطی بودند و می‌توانستند امواج اینترنت را به زمین انتقال دهند. بالن‌های گوگل لوون به‌طور پیوسته در حال حرکت بودند و با همکاری یکدیگر شبکه‌ای از اتصال اینترنتی را به وجود می‌آوردند. این فناوری به‌ویژه برای مناطق روستایی، مناطق آسیب‌دیده از بلایای طبیعی، و کشورهای در حال توسعه که زیرساخت‌های اینترنتی ضعیفی دارند، بسیار مفید بود.

با وجود موفقیت‌هایی که این پروژه در ارائه اینترنت به برخی مناطق داشت، از جمله کمک به برقراری ارتباطات در مواقع اضطراری مانند زلزله یا طوفان، اما با چالش‌های زیادی روبرو شد. این چالش‌ها شامل هزینه‌های بالا، مشکلات فنی، و سختی‌های مدیریت یک شبکه‌ی پایدار از بالن‌ها بود. در نهایت، گوگل در ژانویه ۲۰۲۱ اعلام کرد که پروژه گوگل لوون را به‌دلیل عدم پایداری اقتصادی و چالش‌های دیگر متوقف می‌کند.

هرچند پروژه گوگل لوون به پایان رسید، اما تجربیات و فناوری‌های توسعه‌یافته در این پروژه به بهبود و پیشرفت در زمینه‌های دیگر، از جمله پروژه‌های مشابه و توسعه‌یافته توسط سایر شرکت‌ها و نهادها، کمک کرده است.

آمازون فایر فون (Amazon Fire Phone)

یک گوشی هوشمند بود که توسط آمازون در ژوئن ۲۰۱۴ معرفی شد. این گوشی به عنوان تلاش آمازون برای ورود به بازار گوشی‌های هوشمند و گسترش اکوسیستم خود عرضه شد. فایر فون دارای چند ویژگی منحصر به فرد و نوآورانه بود، اما نتوانست به موفقیت تجاری دست یابد و در نهایت تولید آن متوقف شد.

ویژگی‌ها و قابلیت‌ها: فناوری Dynamic Perspective: یکی از برجسته‌ترین ویژگی‌های آمازون فایر فون، فناوری Dynamic Perspective بود که از چهار دوربین جلویی برای ردیابی حرکت سر کاربر استفاده می‌کرد. این فناوری امکان مشاهده تصاویر و رابط کاربری سه‌بعدی را فراهم می‌کرد و کاربران می‌توانستند با حرکت دادن گوشی، زاویه دید خود را تغییر دهند.
Firefly: این ویژگی دیگر فایر فون بود که به کاربران امکان می‌داد تا اطلاعات مختلفی مانند محصولات، متن‌ها و کدهای QR را شناسایی کنند. این قابلیت به شدت با فروشگاه آمازون یکپارچه شده بود و کاربران می‌توانستند به سرعت محصولات را شناسایی و خریداری کنند.
سیستم عامل Fire OS: فایر فون از سیستم عامل Fire OS استفاده می‌کرد که نسخه‌ای اصلاح‌شده از اندروید بود. این سیستم عامل به شدت به اکوسیستم آمازون متصل بود و شامل دسترسی آسان به خدماتی مانند Amazon Prime و Kindle بود.
چالش‌ها و موانع: قیمت بالا: آمازون فایر فون با قیمتی مشابه گوشی‌های پرچمدار رقیب مانند آیفون و گوشی‌های اندرویدی عرضه شد، اما ویژگی‌ها و قابلیت‌های آن نتوانستند قیمت بالا را توجیه کنند.
محدودیت‌های نرم‌افزاری: یکی از مشکلات اصلی فایر فون، عدم دسترسی به فروشگاه Google Play و برنامه‌های اندرویدی محبوب بود. این محدودیت باعث شد که کاربران نتوانند به راحتی به برنامه‌ها و سرویس‌هایی که عادت به استفاده از آن‌ها داشتند، دسترسی داشته باشند.
تمرکز بیش از حد بر فروشگاه آمازون: فایر فون به شدت به خدمات و فروشگاه آمازون وابسته بود و به نظر می‌رسید که بیشتر به عنوان ابزاری برای خرید از آمازون طراحی شده است تا یک گوشی هوشمند کامل.
نتیجه و توقف پروژه: فایر فون نتوانست توجه کافی از سوی مصرف‌کنندگان جذب کند و فروش آن بسیار کمتر از انتظارات آمازون بود. در نهایت، آمازون در سال ۲۰۱۵ تصمیم گرفت تولید فایر فون را متوقف کند و باقی‌مانده دستگاه‌ها را با تخفیف‌های بزرگ به فروش برساند. پروژه فایر فون به عنوان یک شکست تجاری بزرگ برای آمازون به یاد می‌آید، اما این شرکت از تجربیات حاصل از این پروژه برای بهبود محصولات دیگر خود، مانند دستگاه‌های Echo و تبلت‌های Fire، استفاده کرد.

آمازون داش (Amazon Dash)

مجموعه‌ای از دستگاه‌های کوچک و خدماتی بود که توسط آمازون برای ساده‌سازی و تسهیل فرآیند خرید محصولات روزمره و خانگی طراحی شده بود. این دستگاه‌ها و خدمات در چندین شکل و با ویژگی‌های مختلف عرضه شدند و هدف آن‌ها ارائه تجربه‌ای ساده و سریع برای سفارش کالاها بود.

ویژگی‌ها و انواع
آمازون داش:Dash Button یک دکمه کوچک قابل برنامه‌ریزی بود که کاربران می‌توانستند آن را برای سفارش فوری محصولات خاص، مانند مواد شوینده، نوشیدنی‌ها، و دیگر لوازم خانگی، تنظیم کنند. با فشردن دکمه، محصول انتخاب شده به‌صورت خودکار از طریق حساب کاربری آمازون سفارش داده می‌شد و به آدرس تعیین شده ارسال می‌گردید. هدف از طراحی Dash Button این بود که کاربران بتوانند به راحتی و با یک کلیک ساده، اقلام ضروری خود را بدون نیاز به وارد شدن به وب‌سایت یا اپلیکیشن آمازون سفارش دهند. Dash Wand یک دستگاه دستی کوچک بود که مجهز به اسکنر بارکد و دستیار صوتی الکسا بود. کاربران می‌توانستند بارکد محصولات خود را اسکن کرده یا از طریق دستورهای صوتی، محصولات را به سبد خرید خود اضافه کنند. Dash Wand برای افرادی طراحی شده بود که به راحتی می‌خواستند لیست خرید خود را مدیریت کنند و اقلام مورد نیاز را به سادگی و سریع به سفارشات خود اضافه کنند.
(Dash Replenishment Service (DRS:این سرویس به شرکت‌های تولیدکننده دستگاه‌ها امکان می‌داد که دستگاه‌های خانگی خود را به اینترنت متصل کنند و هنگامی که یک محصول، مانند فیلتر آب یا جوهر پرینتر، به سطح پایینی می‌رسید، به‌طور خودکار سفارش مجدد آن از آمازون انجام شود.
هدف: هدف DRS این بود که فرآیند سفارش محصولات ضروری به‌طور خودکار انجام شود، بدون اینکه کاربران نیاز به یادآوری یا اقدام خاصی داشته باشند.
چالش‌ها و توقف پروژه: محدودیت‌های عملیاتی: یکی از چالش‌های اصلی Dash Button این بود که هر دکمه تنها برای یک محصول خاص تنظیم می‌شد و این محدودیت باعث می‌شد تا کاربران نتوانند از یک دکمه برای چندین محصول مختلف استفاده کنند. با وجود اینکه آمازون داش به عنوان یک نوآوری جالب مطرح شد، اما استقبال از آن به اندازه‌ای نبود که این پروژه بتواند به موفقیت بزرگی دست یابد. بسیاری از کاربران ترجیح می‌دادند تا از طریق اپلیکیشن‌های موبایلی یا وب‌سایت خریدهای خود را انجام دهند.

ظهور دستیارهای صوتی و خانه‌های هوشمند: با رشد و گسترش دستیارهای صوتی مانند الکسا و دستگاه‌های هوشمند خانگی، استفاده از دکمه‌های فیزیکی برای سفارش محصولات کمتر جذاب شد. در آوریل ۲۰۱۹، آمازون اعلام کرد که فروش Dash Button را متوقف می‌کند و در سال ۲۰۲۰، پشتیبانی کامل از آن‌ها را نیز پایان داد. با این حال، Dash Replenishment Service همچنان به عنوان بخشی از اکوسیستم آمازون فعال است و با دستگاه‌های هوشمند مختلف کار می‌کند. پروژه آمازون داش به عنوان یک تجربه جالب در جهت ساده‌سازی خرید روزمره یاد می‌شود، هرچند که نتوانست به موفقیت تجاری بلندمدت دست یابد.

سامسونگ گالکسی بیم (Samsung Galaxy Beam)

یک گوشی هوشمند منحصربه‌فرد بود که در سال ۲۰۱۲ معرفی شد و دارای یک پروژکتور داخلی بود. این گوشی بخشی از سری گالکسی سامسونگ بود و به‌عنوان یک محصول نوآورانه در بازار گوشی‌های هوشمند مطرح شد.

ویژگی‌ها و قابلیت‌ها: سامسونگ گالکسی بیم مجهز به یک پروژکتور پیکو (Pico Projector) داخلی بود که می‌توانست تصاویر و ویدئوها را با وضوح ۶۴۰x۳۶۰ پیکسل بر روی دیوار یا سطوح دیگر نمایش دهد. پروژکتور می‌توانست تصویری با اندازه حداکثر ۵۰ اینچ ایجاد کند. این ویژگی به کاربران امکان می‌داد تا به راحتی محتواهای چندرسانه‌ای خود را در جلسات کاری، نمایش‌های خانگی، یا حتی برای تماشای فیلم و ویدئو با دوستان به اشتراک بگذارند.

سخت‌افزار و نرم‌افزار: صفحه نمایش: این گوشی دارای یک صفحه نمایش ۴٫۰ اینچی با وضوح WVGA (۸۰۰x۴۸۰ پیکسل) بود. پردازنده و حافظه: سامسونگ گالکسی بیم با پردازنده دو هسته‌ای ۱ گیگاهرتزی و ۷۶۸ مگابایت رم عرضه شد، که در آن زمان قدرت پردازشی مناسبی محسوب می‌شد.

سیستم عامل: این گوشی از سیستم عامل اندروید ۲٫۳ (Gingerbread) بهره می‌برد.

باتری: باتری ۲۰۰۰ میلی‌آمپر ساعتی این گوشی به‌خصوص برای پشتیبانی از پروژکتور در نظر گرفته شده بود.

چالش‌ها و موانع: عمر باتری: استفاده از پروژکتور داخلی، مصرف زیادی از باتری را به همراه داشت و به همین دلیل، عمر باتری در هنگام استفاده از پروژکتور بسیار محدود بود.

کیفیت پروژکتور: پروژکتور گالکسی بیم دارای وضوح و روشنایی محدودی بود که باعث می‌شد کیفیت تصویر در محیط‌های با نور کم قابل قبول باشد، اما در محیط‌های روشن‌تر یا بر روی سطوح بزرگ‌تر، کیفیت تصویر به وضوح کاهش می‌یافت.

رقابت: در زمان معرفی گالکسی بیم، گوشی‌های هوشمند با تمرکز بر عملکرد پردازشی و دوربین‌های قوی‌تر در بازار محبوب بودند و ویژگی پروژکتور نتوانست به اندازه‌ای جذابیت داشته باشد که رقابت‌پذیر باشد.

سامسونگ گالکسی بیم به دلیل نوآوری در طراحی و ادغام پروژکتور در یک گوشی هوشمند، توجه زیادی را به خود جلب کرد، اما نتوانست به موفقیت تجاری چشمگیری دست یابد. محدودیت‌های فنی مانند کیفیت پایین پروژکتور و مصرف بالای باتری، به همراه رقابت شدید در بازار گوشی‌های هوشمند، باعث شدند که این محصول تنها به عنوان یک نوآوری جالب شناخته شود و به زودی از بازار خارج شود. سامسونگ تلاش کرد تا در سال ۲۰۱۴ با معرفی مدل به‌روز شده‌ای از این گوشی (Galaxy Beam ۲) این ایده را بهبود دهد، اما آن هم نتوانست به موفقیت قابل توجهی دست یابد.

سامسونگ گالکسی راند (Samsung Galaxy Round)

یکی از اولین گوشی‌های هوشمند با صفحه نمایش منحنی بود که در اکتبر ۲۰۱۳ معرفی شد. این گوشی به عنوان یکی از نوآوری‌های سامسونگ در زمینه طراحی و فناوری صفحه نمایش عرضه شد و توجه زیادی را به خود جلب کرد.

ویژگی‌ها و قابلیت‌ها: گالکسی راند دارای یک صفحه نمایش ۵٫۷ اینچی Full HD Super AMOLED با انحنای افقی بود که به صورت منحنی از لبه به لبه دیگر گوشی امتداد داشت. این طراحی منحصر به فرد به منظور ارائه تجربه کاربری متفاوت و بهبود ارگونومی دستگاه طراحی شده بود.

تجربه کاربری: سامسونگ برخی ویژگی‌های نرم‌افزاری خاص را برای بهره‌گیری از این طراحی منحنی اضافه کرده بود. به عنوان مثال، با تکان دادن گوشی، کاربران می‌توانستند اطلاعاتی مانند ساعت و تاریخ را ببینند، حتی زمانی که گوشی در حالت استندبای بود.

سخت‌افزار: پردازنده و حافظه: گالکسی راند مجهز به پردازنده چهار هسته‌ای Snapdragon ۸۰۰ با سرعت ۲٫۳ گیگاهرتز، ۳ گیگابایت رم و ۳۲ گیگابایت حافظه داخلی بود که از طریق کارت میکرو SD قابل ارتقا بود.

دوربین: این گوشی دارای دوربین عقب ۱۳ مگاپیکسلی و دوربین جلوی ۲ مگاپیکسلی بود که در آن زمان عملکرد بسیار خوبی داشتند.

باتری: باتری این گوشی ۲۸۰۰ میلی‌آمپر ساعتی بود که با توجه به مصرف صفحه نمایش منحنی و دیگر قابلیت‌ها، بهینه‌سازی شده بود.

چالش‌ها و موانع: یکی از اصلی‌ترین چالش‌های گالکسی راند، قیمت بالای آن بود که باعث می‌شد تا این گوشی برای بسیاری از کاربران دست نیافتنی باشد. این قیمت بالا تا حدی به دلیل استفاده از فناوری‌های جدید در صفحه نمایش و تولید محدود بود.

با وجود نوآوری در طراحی، گالکسی راند نتوانست توجه گسترده‌ای در بازار جلب کند. بسیاری از کاربران به طراحی منحنی به عنوان یک ویژگی ضروری نگاه نمی‌کردند و ترجیح می‌دادند تا به گوشی‌های با طراحی سنتی‌تر روی بیاورند.

محدودیت تولید: سامسونگ تصمیم گرفت تا گالکسی راند را به‌صورت محدود تولید کند و بیشتر آن را در بازار کره جنوبی عرضه کرد. این محدودیت در دسترس بودن باعث شد که این گوشی به طور گسترده در دسترس مصرف‌کنندگان جهانی قرار نگیرد.

نسامسونگ گالکسی راند به عنوان یکی از نخستین گوشی‌های هوشمند با صفحه نمایش منحنی، راه را برای نوآوری‌های آینده در زمینه طراحی و فناوری صفحه نمایش هموار کرد. هرچند که این محصول نتوانست موفقیت تجاری قابل توجهی کسب کند، اما به عنوان یک تجربه و آزمایش مهم در توسعه فناوری‌های صفحه نمایش برای سامسونگ ارزشمند بود. این تجربه‌ها بعدها در محصولات موفق‌تری مانند سری گالکسی S با صفحه نمایش‌های Edge به کار گرفته شد که استقبال بیشتری از سوی کاربران به همراه داشت.

فیسبوک پرتال (Facebook Portal)

یک دستگاه ویدئو کنفرانس و نمایشگر هوشمند است که برای اولین بار در سال ۲۰۱۸ معرفی شد. این دستگاه به‌طور خاص برای برقراری تماس‌های ویدئویی و ایجاد ارتباط نزدیک‌تر با دوستان و خانواده طراحی شده بود و از امکانات و ویژگی‌های متعددی برای بهبود تجربه کاربری در تماس‌های ویدئویی برخوردار بود.

ویژگی‌ها و قابلیت‌ها: فیسبوک پرتال در چندین مدل با اندازه‌های مختلف صفحه نمایش (مانند ۱۰٫۱ اینچی و ۱۵٫۶ اینچی) عرضه شد که تجربه‌ای بصری جذاب و راحت را برای کاربران فراهم می‌کرد. دوربین پرتال قابلیت تعقیب حرکت کاربر را داشت. این دوربین با استفاده از هوش مصنوعی، می‌توانست به‌طور خودکار زوم کند و کاربران را در کادر نگه دارد، حتی اگر در حال حرکت بودند. این ویژگی، تماس‌های ویدئویی را طبیعی‌تر و پویاتر می‌کرد. فیسبوک پرتال از طریق مسنجر و واتس‌اپ به کاربران امکان برقراری تماس‌های ویدئویی با دوستان و خانواده را می‌داد. این دستگاه همچنین از قابلیت‌های مختلفی مانند فیلترهای واقعیت افزوده برای افزودن جلوه‌های خاص به تماس‌ها بهره می‌برد. این دستگاه با دستیار صوتی الکسا (Amazon Alexa) ادغام شده بود که امکان کنترل صوتی و دسترسی به امکانات خانه هوشمند، اطلاعات آب‌وهوا، اخبار و غیره را فراهم می‌کرد.

امنیت و حریم خصوصی: فیسبوک پرتال دارای پوشش فیزیکی برای دوربین و دکمه قطع صدا بود که به کاربران امکان می‌داد تا در مواقع عدم استفاده، از حریم خصوصی خود محافظت کنند.

چالش‌ها و موانع: نگرانی‌های حریم خصوصی: یکی از بزرگ‌ترین

موانع برای پذیرش گسترده فیسبوک پرتال، نگرانی‌های مربوط به حریم خصوصی بود. فیسبوک به‌دلیل رسوایی‌های متعدد مرتبط با حفظ اطلاعات شخصی کاربران، با چالش بزرگی در جلب اعتماد مصرف‌کنندگان مواجه بود. این نگرانی‌ها باعث شد تا بسیاری از کاربران از خرید و استفاده از این دستگاه خودداری کنند. اعتماد پایین به فیسبوک به‌عنوان یک شرکت که قبلاً در موضوعات مربوط به امنیت داده‌ها و حریم خصوصی مورد انتقاد قرار گرفته بود، یکی از دلایل اصلی عدم موفقیت این دستگاه در بازار بود. بسیاری از کاربران نسبت به این موضوع شک و تردید داشتند که دستگاهی از فیسبوک بتواند به‌طور ایمن از اطلاعات و حریم خصوصی آن‌ها محافظت کند.

فیسبوک پرتال در بازاری رقابتی عرضه شد که توسط شرکت‌های بزرگی مانند گوگل (Google Nest Hub) و آمازون (Amazon Echo Show) تسلط داشت. این دستگاه‌ها که پیش از پرتال عرضه شده بودند، در میان کاربران محبوبیت بیشتری داشتند و ویژگی‌های مشابهی را ارائه می‌دادند.فیسبوک پرتال به‌عنوان یک دستگاه ویدئو کنفرانس و نمایشگر هوشمند با ویژگی‌های نوآورانه معرفی شد، اما نتوانست به موفقیت بزرگی دست یابد. نگرانی‌های مربوط به حریم خصوصی و اعتماد به فیسبوک، به‌همراه رقابت شدید در بازار دستگاه‌های هوشمند، از عوامل اصلی این عدم موفقیت بودند. با این حال، فیسبوک تلاش کرد با ارائه به‌روزرسانی‌ها و مدل‌های جدیدتر، محصول خود را بهبود بخشد و همچنان در این بازار حضور داشته باشد.

تسلا مدل Y تِنت (Tesla Model Y Tent)

تسلا مدل Y تِنت (Tesla Model Y Tent) یک ایده نوآورانه و غیر معمول بود که به‌عنوان یک گزینه جانبی برای خودروی الکتریکی تسلا مدل Y معرفی شد. این چادر به‌طور خاص برای استفاده در کمپینگ و سفرهای طبیعت‌گردی طراحی شده بود و هدف آن ارائه تجربه‌ای راحت و سازگار با محیط زیست برای افرادی بود که به این نوع فعالیت‌ها علاقه دارند.

ویژگی‌ها و طراحی: طراحی خاص: چادر مدل Y به گونه‌ای طراحی شده بود که به‌طور مستقیم به قسمت عقب خودرو متصل می‌شد و از فضای داخلی خودرو به‌عنوان بخشی از فضای خواب استفاده می‌کرد. این اتصال به گونه‌ای بود که درب عقب خودرو (که به عنوان درب پشتی بالابر یا «Hatchback» شناخته می‌شود) باز می‌ماند و چادر به‌صورت محکم به بدنه خودرو متصل می‌شد.

- فضای خواب اضافی: ظرفیت: این چادر می‌توانست به کاربران فضای خواب بیشتری ارائه دهد و فضای داخلی خودرو را به همراه چادر به یک محیط بزرگتر و راحت‌تر برای استراحت و خواب تبدیل کند. این قابلیت به‌ویژه برای سفرهای طولانی و کمپینگ‌های شبانه مناسب بود.

- سازگاری با محیط زیست: منبع انرژی الکتریکی: با توجه به اینکه تسلا مدل Y یک خودروی الکتریکی است، این چادر با ایده استفاده از انرژی پاک و سازگار با محیط زیست هماهنگی داشت.

کاربـران می‌توانسـتند از باتـری خـودرو بـرای تامیـن بـرق دسـتگاه‌های مختلـف و روشـنایی داخـل چـادر اسـتفاده کننـد.
- واکنـش و اسـتقبال: اسـتقبال خـوب: ایـن محصـول بـا اسـتقبال خوبـی از سـوی علاقه‌منـدان بـه کمپینـگ و طبیعت‌گـردی مواجـه شـد. بسـیاری از کاربـران تسـلا و علاقه‌منـدان بـه ماجراجویی‌هـای فضـای بـاز، از امـکان تبدیـل خـودروی الکتریکـی خـود بـه یـک پناهـگاه قابل‌حمـل اسـتقبال کردنـد.

دلایـل شکسـت: محدودیـت بـازار هـدف: تسـلا مـدل Y تِنـت به‌طـور خـاص بـرای کاربـران خـودروی تسـلا مـدل Y طراحـی شـده بـود، کـه بـازار محـدودی را تشـکیل مـی‌داد. ایـن محصـول تنهـا بـرای افـرادی جـذاب بـود کـه هـم خـودروی تسـلا مـدل Y داشـتند و هـم علاقه‌منـد بـه کمپینـگ بودنـد. ایـن محدودیـت بـازار هـدف، باعـث کاهـش تقاضـا شـد.
رقابـت بـا گزینه‌هـای دیگـر: در بـازار کمپینـگ و تجهیـزات ماجراجویـی، گزینه‌هـای دیگـری ماننـد چادرهـای سـنتی و کمپینگ‌هـای مدرن‌تـر وجـود داشـت کـه بسـیاری از آن‌هـا قابـل حمـل و انعطاف‌پذیـر بودنـد. ایـن گزینه‌هـا ممکـن اسـت به‌صرفه‌تـر و سـازگارتر بـا انـواع خودروهـا بـوده باشـند، کـه باعـث شـد کاربـران تمایـل کمتـری بـه انتخـاب مـدل Y تِنـت داشـته باشـند.
هزینـه بـالا: ماننـد بسـیاری از محصـولات تسـلا، مـدل Y تِنـت احتمـالاً بـا قیمتـی بـالا عرضـه شـد. ایـن موضـوع می‌توانسـت بـرای بسـیاری از کاربـران، کـه بـه دنبـال گزینه‌هـای اقتصادی‌تـر بودنـد، یـک عامـل بازدارنـده باشـد.
نگرانی‌هـای مربـوط بـه نصـب و کارایـی: نصـب ایـن چـادر نیازمنـد دانـش فنـی و زمـان بـود و ممکـن اسـت کاربـران بـا مشـکلاتی در نصـب صحیـح آن مواجـه شـده باشـند. همچنیـن، عملکـرد آن در شـرایط مختلـف جـوی، ماننـد بادهـای شـدید یـا بـاران، ممکـن اسـت به‌انـدازه چادرهـای سـنتی مناسـب نبـوده باشـد.
تغییـرات در اولویت‌هـای تسـلا: تسـلا به‌عنـوان یـک شـرکت خودروهـای الکتریکـی، بیشـتر تمرکـز خـود را روی توسـعه و بهبـود خودروهایـش و فناوری‌هـای مرتبـط بـا آن‌هـا قـرار داده اسـت. بنابرایـن، محصـولات جانبـی ماننـد مـدل Y تِنـت ممکـن اسـت اولویـت کمتـری داشـته باشـند و بـه همیـن دلیـل، پشـتیبانی و توسـعه کافـی بـرای آن‌هـا فراهـم نشـود.
ایـن عوامـل ترکیبـی از دلایـل بودنـد کـه باعـث شـد تسـلا مـدل Y تِنـت نتوانـد موفقیـت بزرگـی کسـب کنـد و بـه یـک محصـول اصلـی و پرطرفـدار تبدیـل شـود.

هواوی اسند P۷ سافیر (Sapphire Huawei Ascend P۷)

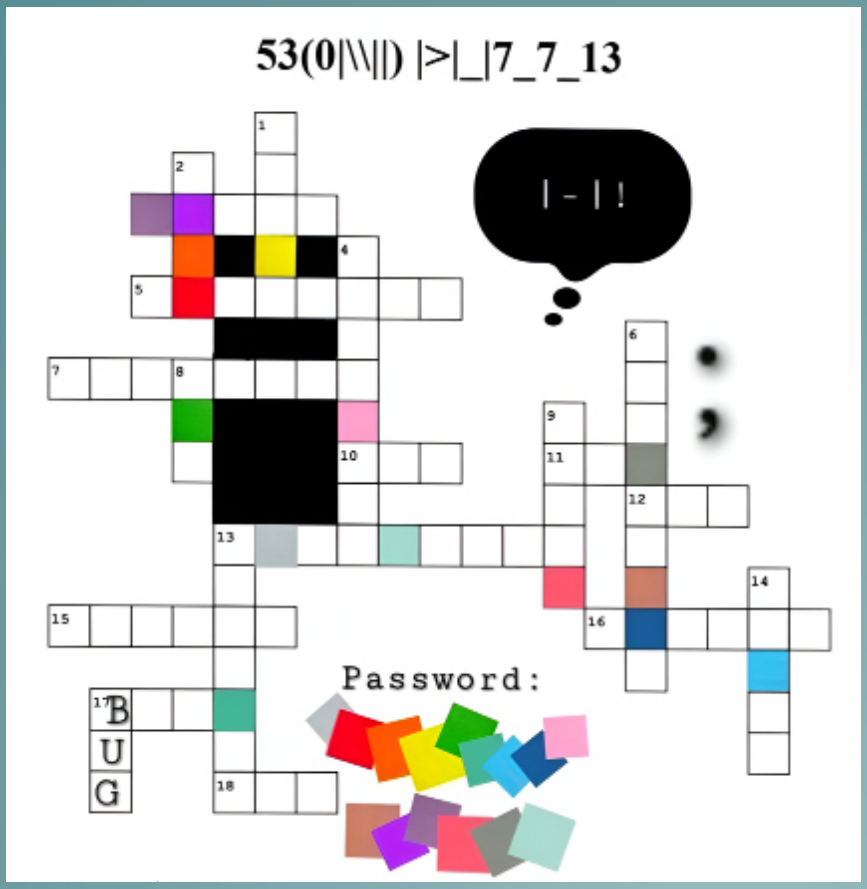
نسـخه‌ای ویـژه از گوشـی هوشـمند هـواوی اسـند P۷ بـود کـه در سـال ۲۰۱۴ معرفـی شـد. ویژگـی برجسـته ایـن نسـخه، صفحـه نمایـش ساخته‌شـده از یاقـوت کبـود بـود کـه بـه دلیـل سـختی بـالا، مقاومـت بیشـتری در برابـر خـراش و ضربـه داشـت. ایـن نـوآوری، گوشـی را بـه یکـی از مقاوم‌تریـن مدل‌هـای آن زمـان تبدیـل کـرد.

چالش‌هـا و موانـع: تولیـد صفحـه نمایـش از جنـس یاقـوت کبـود، فرآینـدی پیچیـده و هزینه‌بـر اسـت کـه ایـن موضـوع بـه افزایـش قیمـت نهایـی محصـول منجـر شـد. ایـن هزینـه بـالا باعـث شـد تـا ایـن نسـخه از هـواوی اسـند P۷ بـرای همـه کاربـران در دسـترس نباشـد و تنهـا بـه گـروه خاصـی از مصرف‌کننـدگان هـدف قـرار گیـرد. بـه دلیـل چالش‌هـای تولیـد و هزینه‌هـای بـالا، هـواوی اسـند P۷ سـافیر تنهـا بـه تعـداد محـدودی تولیـد و عرضـه شـد. ایـن محدودیـت در تولیـد باعـث شـد تـا ایـن محصـول نتوانـد بـه طـور گسـترده در بازارهـای جهانـی در دسـترس قـرار گیـرد و تنهـا در برخـی بازارهـای خـاص و بـه صـورت محـدود عرضـه شـد. هـواوی اسـند P۷ سـافیر نشـان‌دهنده تـلاش هـواوی بـرای ارائـه یـک محصـول لوکـس و مقـاوم بـا اسـتفاده از فناوری‌هـای پیشـرفته ماننـد صفحـه نمایـش یاقـوت کبـود بـود. ایـن دسـتگاه بـا ترکیـب طراحـی شـیک و ویژگی‌هـای منحصـر بـه فـرد، بـه عنـوان یکـی از محصـولات خـاص در زمـان خـود شـناخته شـد. امـا بـه دلیـل قیمـت بـالا و محدودیت‌هـای تولیـد، نتوانسـت بـه موفقیـت تجـاری بزرگـی دسـت یابـد. ایـن تجربـه امـا همچنـان بـه عنـوان یکـی از نمونه‌هـای جسـارت و نـوآوری در دنیـای گوشـی‌های هوشـمند باقـی مانـده اسـت و نشـان می‌دهـد کـه چگونـه شـرکت‌ها بـرای پیشـرفت و ارائـه محصـولات متفـاوت، ریسـک‌هایی را می‌پذیرنـد.

کلام پایانـی: کمپانی‌هـای بـزرگ فنـاوری گاهـی محصـولات غیرعـادی و غیرمتعـارف را معرفـی می‌کننـد کـه ممکـن اسـت همیشـه موفقیت‌آمیـز نباشـند، امـا ایـن محصـولات نشـان‌دهنده تـلاش ایـن شـرکت‌ها بـرای نـوآوری و گسـترش حوزه‌هـای فعالیتشـان هسـتند. ایـن نـوع محصـولات، نمونه‌هایـی از جسـارت و خلاقیـت در دنیـای فنـاوری بـه شـمار می‌آینـد و بـه رغـم عـدم موفقیت‌هـای تجـاری گسـترده، بـه عنـوان تجربیـات آموزنـده بـرای آینـده صنعـت فنـاوری باقـی می‌ماننـد. ایـن تلاش‌هـا می‌تواننـد الهام‌بخـش پیشـرفت‌های جدیـد و راه‌هـای نوآورانـه در طراحـی و توسـعه محصـولات فنـاوری باشـند.

نویسنده: آرزو پروازی

پازل جایزه دار مجله



سلام به همه!
این پازل یه پازل کراس ورد عادیه
اتفاقا کلماتش خیلی ساده‌ن😊
فقط یه چیزی در موردش خاصه
و اون اینه که زبون سوالات با هر زبونی که (شاید) تا حالا دیدید متفاوته😁

یه مثال:
کلمه ی down17 رو نگاه کنید نوشته

3|2|20|2

این تبدیل میشه به

3RR0R

و بعد

ERROR

و ارور ها در دنیای برنامه نویسی باگ نامیده میشن✨
پس ۱۷ عمودی میشه BUG

مثال دوم:
فقط برای اینکه یکم با حروف بیشتر آشنا شین،
ربات سمیکالن اینجا داره میگه Hi
که به زبون خودش میشه !|-|👋

و در آخر، مکعب های رنگیو به ترتیبی بچینین که رمز پیدا بشه🕵

اگه BUG ای در جدول پیدا کردین، به ما بگین
ممنونم😊

پ.ن: موفق باشین🍇

Across

3. 84|) |#0|2 80+|-|, |-||_||\\/|4|\\| 4|\\||) (0|\\/||>|_|+3|2
5. 4 |\\|3+\\|/0|2|(53(|_||2!+`/ 5`/5+3|\\/|
7. 4|\\| !|\\||>|_|+ |)3\\/!(3, 4150 4 |\\/||_|5!(41 !|\\|5+|2|_||\\/|3|\\|+
10. 4|>|>1!(4+!0|\\|, 5|-|0|2+
11. 4|>|>1!(4+!0|\\| |>|206|24|\\/||\\/|!|\\|6 !|\\|+3|2|#4(3
12. (3|\\|+|241 |>|20(355!|\\|6 |_||\\|!+
13. 4 |#4|\\/|0|_|5 +00+|-|
15. 5|\\|4|(3 14|\\|6|_|463
16. 8!5(|_|!+
17. 845!(!|\\||>|_|+/0|_|+|>|_|+ 5`/5+3|\\/|
18. |24|\\||)0|\\/| 4((355 |\\/|3|\\/|0|2`/

Down

1. 4 5|\\/|411 |24+
2. \\|/!|231355 |#!|)31!+`/
4. |\\|0+ 4 50|#+\\|/4|23
6. 5`/|\\/|801 +|-|4+ 3|\\||)5 4 1!|\\|3 !|\\| (++
8. 8!|\\|4|2`/ |)!6!+
9. |-|!|)|)3|\\| |\\/|3|\\/|0|2`/
13. (|-||20|\\/|3, |#!|23|#0%, 0|>3|24, 3+(
14. 0|>3|\\| 50|_||2(3 05
17. 3|2|20|2

◆ **با حل و ارسال پاســخ به تلگرام و اینستاگرام ما برنده جایزه شوید.**

کانال تلگرام
اسکن کنید...

کانال اینستاگرام
اسکن کنید...