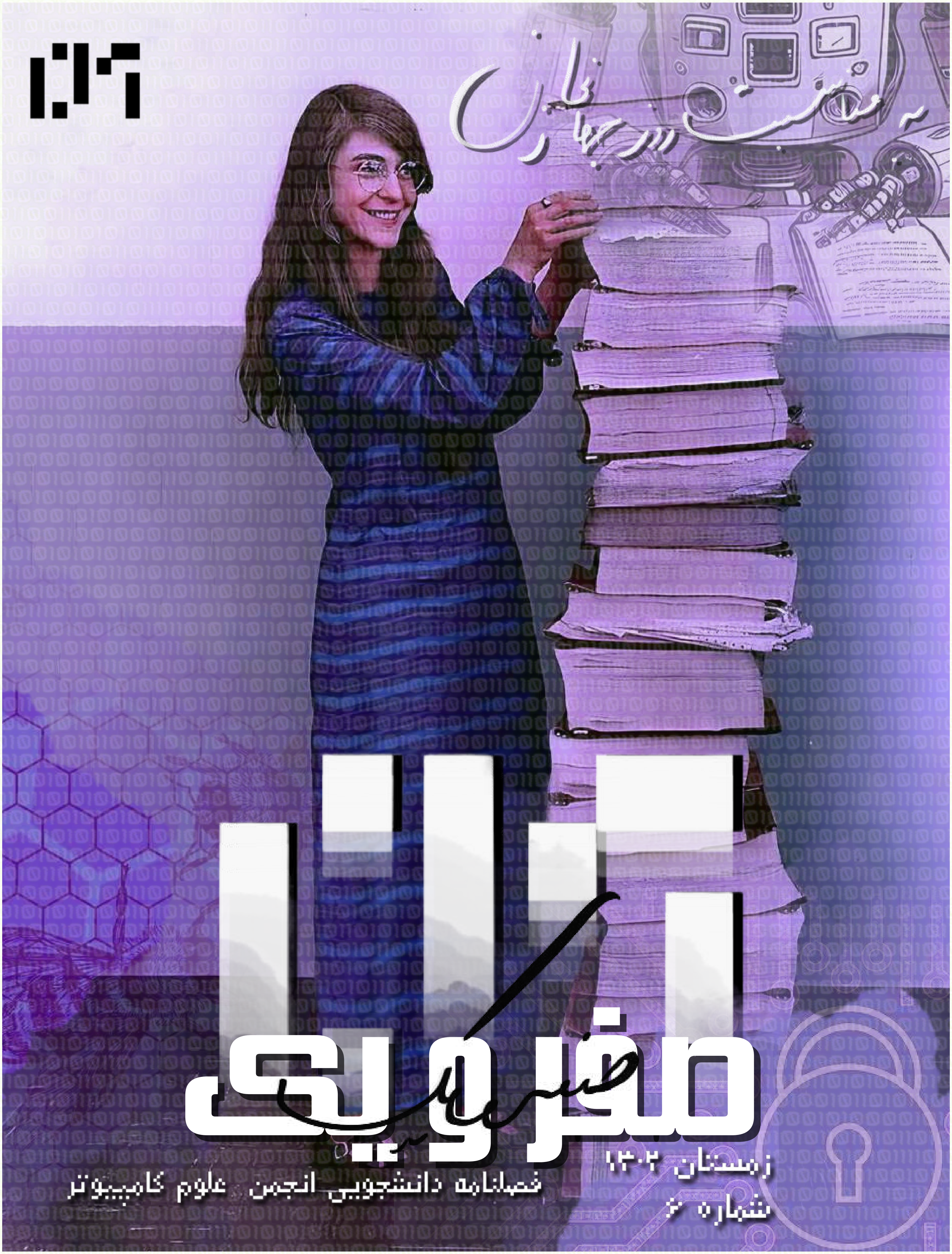
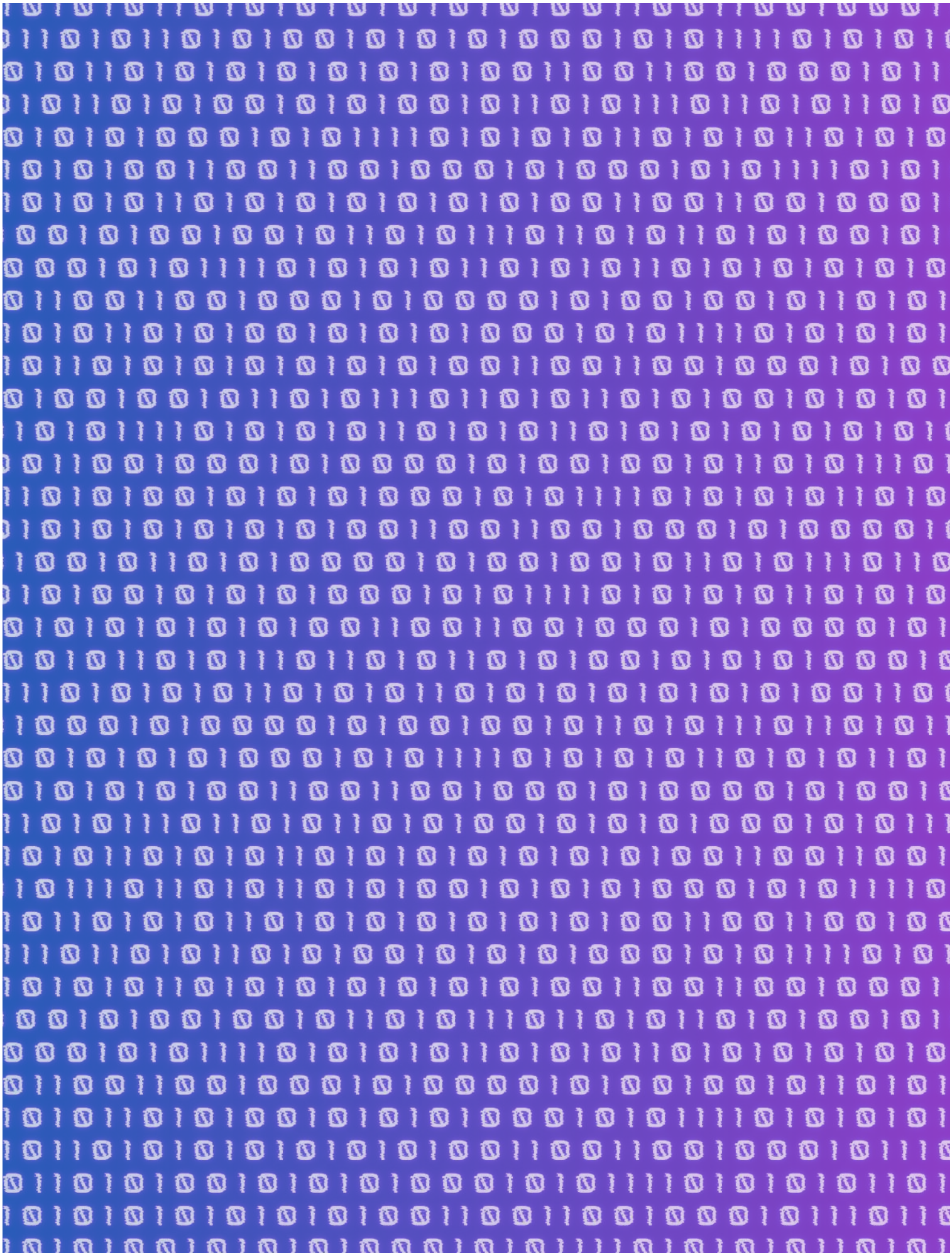
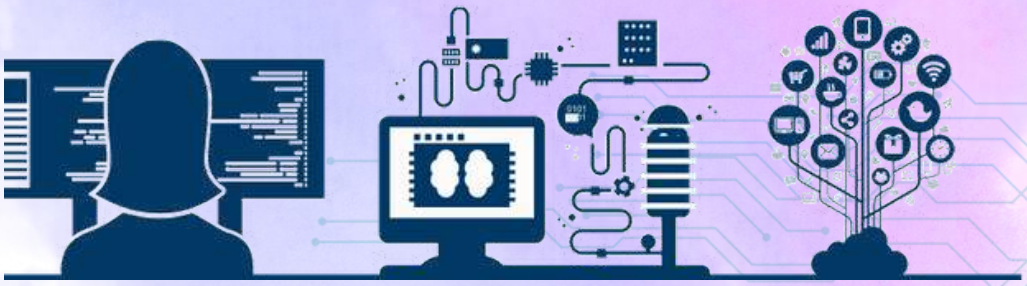

به مناسبت روز جهانی زن
صفر و یک
زمستان ۱۴۰۲
فصلنامه دانشجویی انجمن علوم کامپیوتر
شماره ۶



زمستان ۱۴۰۲
شماره ۶

فهرست مطالب



سخن سردبیر

سلام به همه دوستان صفرویکی و طرفداران نشریه صفرویک!

با افتخار، این شماره از نشریه علوم کامپیوتری را به مناسبت روز جهانی زن به شما تقدیم می‌کنم. در این شماره، به یاد مارگارت همیلتون، یکی از پیشگامان علم کامپیوتر و زنان برجسته در این حوزه، خواهیم بود و با الگوریتم رقص زنبور عسل، که تاثیر فراوانی بر رشد حوزهٔ کامپیوتر داشته است، آشنا خواهیم شد.

علاوه بر این، در این شماره، با الگوریتم‌های RSA و الگوریتم‌های حریصانه آشنا خواهید شد و در مسابقه‌ی الگوریتمی ما، فرصت خواهید داشت تا توانایی‌های خود را به چالش بکشید و با هم رقابت کنید. این مسابقه در راستای الگوریتم حریصانه‌ای است که در ابتدا آن را توضیح دادیم و سپس سوال الگوریتمی آن را طرح کردیم. پیشنهاد من به شما این است که از حل کردن سوالات الگوریتمی نترسید، به شما قول می‌دهم که تجربهٔ جذابی خواهید داشت.

پادکست‌ها، فیلم‌ها و کتاب‌های پرطرفدار در حوزهٔ علوم کامپیوتری نیز در این شماره معرفی شده است تا شما علاقه‌مندان به این حوزه، بتوانید از مطالب جذاب و آموزنده این شماره بهره‌برداری کنید.

در پایان، می‌خواهم از دکتر صادقی بی‌غم به خاطر حمایت و راهنمایی ارزنده‌اشان، که به ما کمک کردند تا نشریه صفرویک را بهتر و با کیفیت بیشتری پیش ببریم، صمیمانه تشکر کنم.

به امید روزهای بهتر،
سحر حقیقت

الگوریتم حریصانه

نویسندگان:سحر حقیقت,حلیا رضائی,هانیه گلشن

الگوریتم حریصانه، یک روش حل مسئله در علوم کامپیوتر است که بدون در نظر گرفتن تأثیرات آینده، بر اساس انتخاب بهترین گزینهٔ موجود در هر مرحله از حل مسئله عمل می‌کند. این الگوریتم‌ها بدون تحلیل عمقی از تأثیرات آینده، به دنبال بهینه‌سازی یک هدف خاص هستند.

در روش حریصانه، رسیدن به هدف در هر گام، مستقل از گام قبلی و بعدی است، یعنی در هر مرحله برای رسیدن به هدف نهایی، مستقل از این که در مراحل قبلی چه انتخاب‌هایی صورت گرفته و انتخاب فعلی ممکن است چه انتخاب‌هایی در پی داشته باشد؛ انتخابی صورت می‌پذیرد که در ظاهر بهترین انتخاب ممکن است. به همین دلیل است که به این روش، روش حریصانه گفته می‌شود.

برای فهم بهتر اگر یادتان باشد الگوریتم پویا یک روش حل مسائل محاسباتی است که زیرمسئله‌ها را یک بار حل و جواب آن‌ها را در یک جدول ذخیره می‌کند و با این کار از تکرار اجرای زیرمسئله‌ها در زمانی که مجدداً به جواب آن‌ها نیاز است جلوگیری می‌کند. مثلا اگر مسئله فیبوناچی با برنامه ریزی پویا حل شود باعث اجرای مکرر یک زیرمسئله نمی شود. الگوریتم پویا برای حل مسائل بهینه‌سازی، بهینه‌سازی تصادفی، جستجو در فضای حالات و سایر مسائل پیدا کردن راه‌حل مناسب استفاده می‌شود.ذهنیت اکثریت ما این است که همیشه استفاده از یک برنامه‌ریزی پویا، برای تمامی مسائل بهترین راه است!
این حرف کاملا درست است. البته تا زمانی که هنوز به یک مسئلهٔ بهینه‌سازی برخورد نکرده‌ایم!
احتمالا درطول تحصیل خود در رشتهٔ کامپیوتر و شاخه‌های مرتبط با آن، حداقل یک بار به مسائل بهینه‌سازی، برخورد کرده‌اید و اگرنه، درآینده حتماً برخورد خواهید کرد.

آنجاست که متوجه می‌شوید اگر بخواهید از یک برنامه‌ریزی پویا برای حل آن استفاده کنید، ممکن است مجبور شوید سال‌ها برای نتیجهٔ آن صبر کنید.
اما نگران نباشید الگوریتم حریصانه دقیقاً در این هنگام به کمک شما می‌آید!الگوریتم‌های حریصانه همیشه گزینه‌ای را انتخاب می‌کنند که درآن لحظه بهترین به نظر می‌آید، یعنی انتخاب‌های بهینهٔ محلی (نسبی) انجام می‌دهند، تا در انتها به یک جواب بهینه برای مسئله برسند.

این الگوریتم‌ها معمولاً سریع هستند و بسیار راحت پیاده‌سازی می‌شوند، گاهی چنان راحت که به درستی آن شک می‌کنید!

گرچه در بعضی از مسائل، جواب نسبی به دست آمده توسط این الگوریتم‌ها با جواب بهینه تفاوت دارد، اما گاهی حتی در چنین مسائلی (به خصوص مسائلی با پیچیدگی بالا و یا NP-hard) جواب به دست آمده، تخمین خوبی از جواب واقعی است.

به عنوان مثال، زمانی که یک دزد عجول و حریص وارد خانه‌ای می‌شود، در مسیر حرکت خود هر وسیله و کالای با ارزشی را داخل کیسه می‌اندازد. وی در این حالت چندان توجهی نمی‌کند که چه اشیائی را قبلاً برداشته و ممکن است در آینده چه اشیاء گران‌بهاتری به دست آورد. او درهر گام تنها از بین اشیای دم دست خود باارزش‌ترین آن را انتخاب کرده و به وسائل قبلی اضافه می‌کند.

این روش کاربردهای عمومی نیز دارد. به عنوان مثال، زمانی که در مقابل خرید از یک فروشگاه یک اسکناس تحویل فروشنده داده می‌شود، وی با یک حساب سرانگشتی سعی می‌کند با حداقل اسکناس‌ها و سکه‌های ممکن، باقیماندهٔ پول را تولید کرده و به خریدار تحویل دهد. این حساب سرانگشتی ممکن است روش حریصانه باشد.

در بسیاری از موارد، مسئله به طور مستقیم یا غیرمستقیم از ما می‌خواهد که یک متغیر را کمینه یا بیشینه کنیم. در اکثر مسائل این‌چنینی ما باید تعدادی انتخاب انجام دهیم، و مسئله در تعدادی مرحله حل شود. بازهٔ وسیعی از این مسائل توسط الگوریتم‌های حریصانه قابل حل هستند.

کلیت روش حریصانه در هر مرحله، انتخاب یک عنصر از عناصر موجود است. کار با یک مجموعهٔ تهی شروع شده و این عنصر قسمتی از جواب مسئله است که به ترتیبی خاص به مجموعه عناصر نهایی اضافه می‌شود.
به دلیل این که هر الگوریتم حریصانه الزاماً حل بهینه را نمی‌دهد، برای هر مسئلهٔ خاص باید اثبات کنیم که آیا الگوریتم حریصانه برای آن، جواب بهینه می‌دهد یا خیر. و این موضوع اغلب سخت‌ترین مرحلهٔ کار است.

در طی این مسیر گام‌های زیر اتفاق می‌افتد:

1. روال انتخاب حریصانه (Selection): در این گام یک عنصر برای اضافه شدن به مجموعهٔ جواب، انتخاب می‌شود. معیار یا روال انتخاب عنصر برای اضافه شدن، ارزش آن عنصر است. بسته به نوع مسئله، هرعنصر ارزشی دارد که باارزش‌ترین آن‌ها انتخاب می‌شود.

2. امکان سنجی و افزودن (Feasible): پس از انتخاب یک عنصر به صورت حریصانه، باید بررسی شود که آیا امکان اضافه کردن آن به مجموعه جواب‌های قبلی وجود دارد یا نه. گاهی اضافه شدن عنصر، یکی از شرایط اولیهٔ مسئله را نقض می‌کند که باید به آن توجه نمود. اگر اضافه کردن این عنصر هیچ شرطی را نقض نکند، عنصر اضافه خواهد شد؛ وگرنه کنار گذاشته می‌شود و بر اساس گام اول، عنصر دیگری برای اضافه شدن انتخاب می‌شود. اگر گزینهٔ دیگری برای انتخاب وجود نداشته باشد، اجرای الگوریتم به اتمام می‌رسد.

3. بررسی اتمام الگوریتم (Solution): در هر مرحله پس از اتمام گام دوم و اضافه شدن یک عنصر جدید به مجموعه جواب، باید بررسی کنیم که آیا به یک جواب مطلوب رسیده‌ایم یا خیر؟ اگر نرسیده باشیم؛ به گام اول می‌رویم و چرخه را در مراحل بعدی ادامه می‌دهیم.
به زبان ساده، در روش حریصانه طی هر مرحله یک عنصر به روش حریصانه به مجموعه جواب اضافه شده، شرط محدودیت‌ها بررسی شده و در صورت نبود مشکل، عنصر و عناصر بعدی به همین ترتیب به مجموعه جواب اضافه می‌شوند.

چه زمانی از الگوریتم حریصانه استفاده می شود؟
پیش از آن که الگوریتمی را برای حل مسئله خود انتخاب کنیم، باید به ۲ پرسش اصلی پاسخ دهیم. اولین پرسش این است که آیا برای حل مسئله به دنبال یافتن بهترین تخمین پاسخ با هزینه زمانی پایین هستیم؟
دومین پرسش در این‌باره است که آیا برای حل مسئله به دنبال یافتن راه‌حل بهینه هستیم؟
چنانچه پاسخ سوال اول، مثبت باشد، الگوریتم حریصانه می‌تواند یکی از بهترین راه‌حل‌ها برای حل مسئلهٔ ما باشد.
اما اگر پاسخ سوال دوم مثبت باشد، الگوریتم حریصانه لزوما نمی‌تواند بهترین جواب را بدهد.
برای تفهیم بهتر به این مثال ساده توجه کنید:

مسئلهٔ برنامه ریزی برای کلاس درس
فرض کنید تنها یک اتاق برای کلاس درس دارید، و می‌خواهید بیشترین تعداد ممکن کلاس را در آنجا برگزار کنید. لیستی از کلاس‌ها را در اختیار دارید.

کلاس	شروع	پایان
هنر	۹ AM	۱۰ AM
انگلیسی	۹:۳۰ AM	۱۰:۳۰ AM
ریاضی	۱۰ AM	۱۱ AM
کامپیوتر	۱۰:۳۰ AM	۱۱:۳۰ AM
موسیقی	۱۱ AM	۱۲ PM

شما نمی‌توانید همهٔ این کلاس‌ها را در آنجا برگزار کنید، چون برخی از آن‌ها با یکدیگر همپوشانی زمانی دارند.

شما می‌خواهید تا جایی که ممکن است کلاس‌های بیشتری را در این اتاق برگزار کنید. چگونه مجموعه‌ای از کلاس‌ها را انتخاب می‌کنید، به شرط آنکه بزرگترین تعداد کلاس‌های ممکن را داشته باشید؟
مسئلهٔ دشواری به نظر می‌رسد، درست است؟
در واقع، آنقدر الگوریتم آسانی دارد که ممکن است شما را شگفت‌زده کند.

نحوه‌ی عملکرد الگوریتم در زیر آمده است:

1.کلاسی را انتخاب کنید که زودتر به اتمام میرسد. این اولین کلاسی است که در این اتاق برگزار می‌کنید.

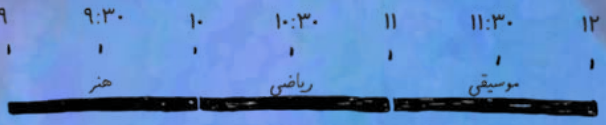
2. حالا باید کلاسی را انتخاب کنید که بعد از اولین کلاس شروع شود. دوباره کلاسی را انتخاب کنید که زودتر تمام می‌شود. این دومین کلاسی است که برگزار می‌کنید.
به این کار ادامه داده تا در نهایت به پاسخ برسید!
بیایید آن را امتحان کنیم. کلاس هنر قبل از همه، در ساعت ۱۰ صبح به پایان میرسد، بنابراین این یکی از کلاس‌هایی است که شما انتخاب می‌کنید.

کلاس	شروع	پایان
هنر	۹ AM	۱۰ AM
انگلیسی	۹:۳۰ AM	۱۰:۳۰ AM
ریاضی	۱۰ AM	۱۱ AM
کامپیوتر	۱۰:۳۰ AM	۱۱:۳۰ AM
موسیقی	۱۱ AM	۱۲ PM

اکنون به کلاسی نیاز دارید که بعد از ساعت ۱۰ صبح شروع شده، و زودتر از باقی کلاس‌ها به پایان می‌رسد.

زبان انگلیسی به دلیل تداخل با هنر، از فهرست مورد نظر ما حذف می‌شود، اما کلاس ریاضی مناسب است. در نهایت، کامپیوتر با ریاضی تداخل زمانی دارد، اما کلاس موسیقی مشکلی ندارد.

بنابراین این سه کلاس را در این کلاس درس برگزار خواهید کرد.



بسیاری شاید بگویند از آنجایی که این الگوریتم آسان است و بیش از اندازه واضح است، پس لابد اشتباه است.
اما همین ویژگی، زیبایی الگوریتم‌های حریصانه است: آن‌ها آسان هستند!

یک الگوریتم حریصانه ساده است:
در هر مرحله، حرکت بهینه را انتخاب کنید. در این حالت، هر بار که کلاسی را انتخاب می‌کنید، کلاسی را انتخاب می‌کنید که زودتر از همه به پایان می‌رسد.

از نظر فنّی: در هر مرحله شما راه حل بهینهٔ محلی را انتخاب می‌کنید و در پایان با راه حل بهینهٔ سراسری روبه‌رو می‌شوید. باور کنید یا نه، این الگوریتم ساده راه‌حل بهینه را برای این مسئلهٔ زمان بندی پیدا می‌کند!
بدیهی است که الگوریتم‌های حریصانه همیشه کارساز نیستند. اما نوشتن آن‌ها ساده است!

الگوریتم حریصانه، ماهیتی شهودی دارد و همین امر باعث می‌شود از نظر ریاضی کاملا قابل تخمین نباشد. گاهی ممکن است گزینه‌ای با ارزش را انتخاب کند که در واقع این گزینه در نهایت، ما را به بهترین نتیجه نرساند. برای جلوگیری از بروز مشکلات، بهتر است کدهایی را برای انتخاب درست بهترین گزینه، به الگوریتم اضافه کرد تا از عملکرد آن مطمئن شد.

برخی از پروتکل‌ها و الگوریتم‌های رایجی که از عملکرد حریصانه استفاده می‌کنند شامل موارد زیر هستند:

۱. الگوریتم Minimum Spanning Tree (MST)
۲.الگوریتم کوتاه ترین مسیر دایجکسترا (DIJKSTRA)
۳. بهینه سازی انتخاب فعالیت (ACTIVE SELECTION PROBLEM)
۴. کدگذاری هافمن (برای فشرده‌سازی داده‌ها بدون خرابی و آسیب استفاده می شود.)
۵. پروتکل مسیریابی از طریق میانبرها (OSPF)

تا به الآن، بیشتر از مزایای الگوریتم حریصانه گفتیم؛ برای تکمیل مطالب بهتر است کمی هم با معایب آن آشنا شویم که بتوانیم به بهترین شکل و در بهترین زمان، از آن استفاده کنیم.

همان‌طور که اشاره کردیم الگوریتم حریصانه در هر مرحله، بهترین انتخاب را براساس اطلاعات موجود در آن مرحله انجام می‌دهد، اما ممکن است در نتیجهٔ نهایی بهترین جواب را از دست بدهد. این به معنای این است که الگوریتم حریصانه ممکن است در برخی موارد به جواب‌های نادرست یا ناقص منجر شود.
همچنین الگوریتم حریصانه معمولاً به صورت مستقل از داده‌های ورودی عمل می‌کند؛ و فقط به اطلاعات فعلی توجه می‌کند. در برخی موارد، این می‌تواند منجر به جواب‌های غیربهینه شود، زیرا عدم توجه به اطلاعات قبلی و مستقل بودن از داده‌های ورودی می‌تواند باعث از دست رفتن اطلاعات مهم شود. واضح است که این یعنی الگوریتم حریصانه به ساختار مسئله بسیار وابسته است. این بدین معناست که اگر ساختار مسئله تغییر کند یا شرایط مسئله تغییر کنند، الگوریتم حریصانه ممکن است به نتایج ناقص یا نادرستی منجر شود.

به طور خلاصه، الگوریتم حریصانه دارای سرعت اجرا و کارایی قابل توجهی است، اما ممکن است به جواب‌های نادرست یا ناقص منجر شود، در برخی موارد عدم ارضای نیازهای واقعی مسئله باشد. برای استفاده از الگوریتم حریصانه، باید با دقت، ساختار مسئله و شرایط آن را بررسی کرده و بهبودهای ممکن در الگوریتم را در نظر بگیرید.
طراحی الگوریتم و روش‌های آن از موضوعات مهم حوزهٔ علوم کامپیوتر محسوب می‌شود. با کمک این روش‌ها به دنبال پیدا کردن راهی برای حل مسئله هستیم. هر یک از الگوریتم‌ها ویژگی‌های منحصربه فردی دارند که بنا به نیاز مسئله، می‌توان برخی از آن‌ها را برای یافتن پاسخ استفاده کرد.
از میان روش‌های مختلف، الگوریتم حریصانه یکی از روش‌های پرکاربرد حل مسئله محسوب می‌شود که تمرکز آن بر روی پیدا کردن راه حل مسئله با کمترین هزینه است. در مطلب فعلی به توضیح الگوریتم حریصانه و ویژگی‌ها و مزایا و معایب آن پرداختیم و مثالی کاربردی برای درک سادهٔ آن ارائه کردیم.

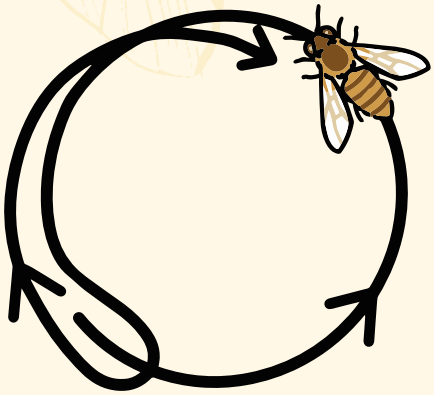
رقص زنبور عسل

نویسندگان:مریم خلیلی, اعظم سادات کامروافر

زنبورهای عسل جزو حشره‌های اجتماعی هستند. لازمهٔ تداوم زندگی اجتماعی زنبورها وجود سیستم مکالماتی مؤثر، بین آنها است. این لزوماً به آن مفهوم نیست که ارگانیسم دارای قدرت تکلم باشد و یا منحصراً نسبت به محرکی عکس‌العمل نشان دهد.

اصولاً ارتباط یعنی انتقال اطلاعات از یک فرد به فرد دیگر، از همان‌گونه که قدرت دریافت داشته باشد.
سیستم انجام این ارتباطات در حشرات مثل مکالمات انسان بر مبنای استفاده از تحریکات مختلف، مثل نور مواد شیمیایی و فیزیکی می‌باشد که این تحریکات، بوسیلهٔ اعضای مخصوص دریافت می‌شوند.

انجام مکالمات در کلنی زنبورعسل، از قرن ۱۸ برای انسان شناخته شد؛ در این قرن آقای Spitzner اعلام کرد که زنبورعسل به کمک رقص خود، می‌تواند مقدار عسل و منابع را به سایر افراد کلنی اطلاع دهد. هنگامی که یک زنبور کارگر، در جایی به یک منبع مناسب می‌رسد؛ در بازگشت به کندو آن را به طریق جالبی به سایر افراد اطلاع می‌دهد. این زنبور در نهایت شادمانی، بدن خود را می‌جنباند و در مجاورت دیگر زنبورهای کارگر، دایره‌هایی را از بالا به پایین و از پایین به بالا طی می‌کند.



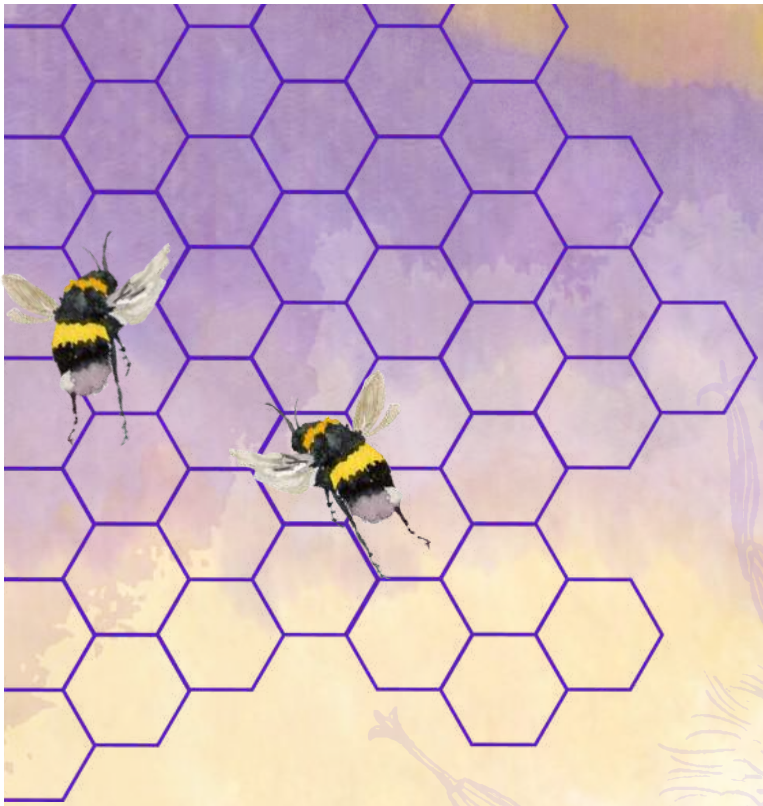
بطور کلی محققین سه نوع رقص را، در کلنی زنبور عسل مشاهده کرده‌اند:

رقص دایره ای(Round dance)
زنبور با حرکت سریع دایره‌وار، چرخیده و گاهی جهت خود را عوض کرده و در جهت مخالف حرکت می‌کند. این رقص چندین ثانیه ادامه یافته و سپس متوقف می‌شود. این رقص در مواقعی انجام می‌شود که منبع غذا بین ۰ تا ۱۵ متری کندو باشد.

رقص دانسی (Stickle dance)
در صورتیکه منبع غذا بین ۱۵ تا ۱۰۰ متری کندو باشد؛ زنبور اقدام به اجرای این رقص می‌کند.
روش انجام این رقص، مشابه رقص دایره‌ای است. با این تفاوت که جهت انجام رقص، به صورت نیم دایره یا داسی شکل است. حدود فاصلهٔ منبع در این رقص مبادله می‌شود.

رقص ارتعاش شکم (Wag Tail dance)
در صورتیکه فاصلهٔ غذا از کندو بیش از ۱۰۰ متر باشد؛ زنبور اقدام به این رقص می‌کند. در این نوع رقص ابتدا زنبور در جهت یک نیم دایره رقصیده؛ از روی خط مستقیم دایره بالا رفته و اقدام به حرکت روی نیم دایرهٔ طرف دیگر کرده و دوباره روی قطر دایره بالا آمده و در طرف دیگر روی نیم دایرهٔ طرف اول، حرکت کرده و نهایتاً یک ۸ هشت انگلیسی را می‌سازد.

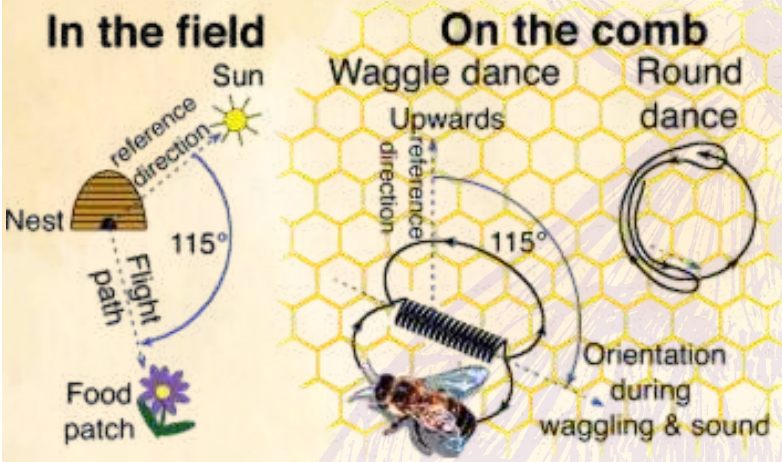
در جریان رقص ارتعاش شکم، اطلاعات متعددی بین زنبور رقص کننده و سایر زنبورها مبادله می‌گردد از جمله:
- فاصله تا منبع غذا
- جهت منبع غذا

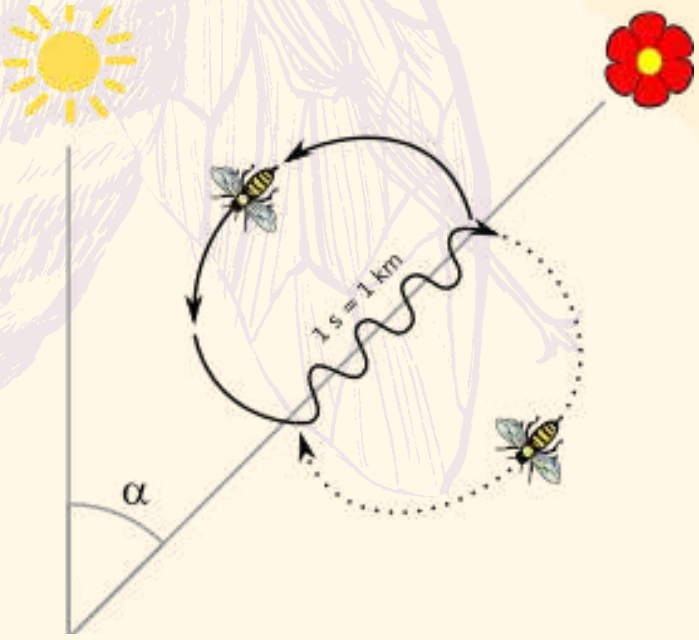


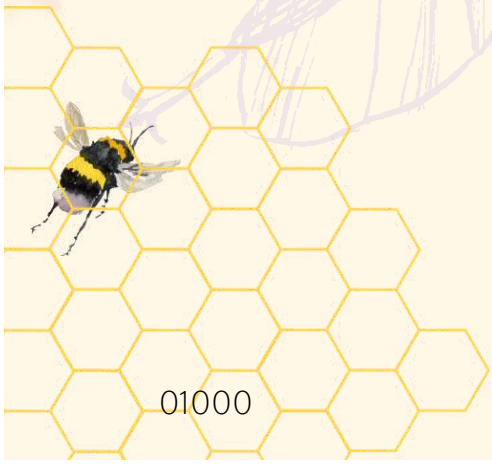
به گفتهٔ محققان، این حرکت‌ها از هشت قسمت تشکیل شده است و جهت حرکات، جهت محل غذا و میزان شدت حرکات، میزان دوری و نزدیکی مسیر را نشان می‌دهد.
به گفتهٔ دانشمندان، هنگامی که زنبور این حرکات را انجام می‌دهد زنبورهای دیگر دور او جمع می‌شوند و حرکاتش را به دقت نظاره می‌کنند.

وز وز کردن زنبورهای عسل

وز وز کردن زنبورهای عسل می‌تواند به سربازان در شناسایی مواد شیمیایی سمی در حملات تروریستی یا به هنگام جنگ کمک کند. دانشمندان آمریکایی دریافته‌اند که زنبورهای عسل، سیستم‌های زندهٔ هشداردهنده به شدت حساسی هستند؛ چرا که به هنگام تماس با انواع مختلفی از مواد شیمیایی، نحوهٔ وز وز کردن آن‌ها تغییر می‌کند.







نویسندگان: ریحانه محمدزاده, مبیناسادات شریعت پناهی

الگوریتم RSA (Rivest-Shamir-Adleman) اساس یک سیستم رمزنگاری است، که رمزگذاری کلید عمومی را امکان‌پذیر می‌کند و به طور گسترده برای ایمن‌سازی داده‌های حساس استفاده می‌شود؛ به ویژه زمانی که از طریق یک شبکهٔ ناامن مانند اینترنت ارسال می‌شود.
به طور ساده‌تر، الگوریتمی است که توسط رایانه‌های مدرن برای رمزگذاری و رمزگشایی پیام‌ها استفاده می‌شود.

RSA اولین بار در سال 1977، توسط ران ریوست آدی شامیر و لئونارد ادلمن، از مؤسسهٔ فناوری ماساچوست، به طور عمومی توصیف شد؛ اگرچه ایجاد یک الگوریتم کلید عمومی در سال 1973، توسط ریاضیدان انگلیسی کلیفورد کاکس، توسط GCHQ بریتانیا تا سال 1997، طبقه بندی شده بود.

RSA از حروف اول بنیانگذاران آن، ران ریوست، آدی شامیر و لئونارد ادلمن نامگذاری شده است.
رمزنگاری کلید عمومی، که به نام رمزنگاری نامتقارن نیز شناخته می‌شود، از دو کلید متفاوت اما از نظر ریاضی مرتبط، استفاده می‌کند.

در این سیستم دو کلید داریم، یکی را عمومی و دیگری را خصوصی می‌نامند.
کلید عمومی را می‌توان با همه به اشتراک گذاشت، در حالی که کلید خصوصی باید مخفی بماند. محاسبهٔ کلید خصوصی از کلید عمومی بسیار دشوار است و شاید بتوان گفت که از دشواری‌ها و چالش‌های اصلی، برای شکستن این الگوریتم باشد.

در رمزنگاری RSA، هم کلید عمومی و هم کلید خصوصی می‌توانند پیامی را رمزگذاری کنند. کلید مخالف با کلیدی که برای رمزگذاری یک پیام استفاده می‌شود، برای رمزگشایی آن استفاده می‌شود. این ویژگی، یکی از دلایلی است که چرا RSA، به پرکاربردترین الگوریتم نامتقارن، تبدیل شده است؛ روشی برای اطمینان از محرمانه بودن، یکپارچگی، اصالت و عدم انکار ارتباطات الکترونیکی و ذخیره‌سازی داده‌ها ارائه می‌دهد.

بسیاری از پروتکل‌ها، از جمله Secure Shell SSH، OpenPGP، S/MIME و SSL/TLS، برای رمزگذاری و توابع امضای دیجیتال، به RSA متکی هستند. همچنین در برنامه‌های نرم‌افزاری مورد استفاده قرار می‌گیرد.
- مرورگرها یک مثال واضح هستند، زیرا باید یک اتصال امن را از طریق یک شبکهٔ ناامن مانند اینترنت، برقرار کنند یا یک امضای دیجیتال را تأیید کنند. تایید امضای RSA یکی از متداول ترین عملیات انجام شده در سیستم‌های متصل به شبکه است.

چرا از الگوریتم RSA استفاده می‌شود؟

RSA امنیت خود را از دشواری فاکتورگیری اعداد صحیح بزرگ، که حاصل ضرب دو عدد اول بزرگ هستند به دست می‌آورد. ضرب این دو عدد آسان است، اما تعیین اعداد اول اصلی یا فاکتورگیری به دلیل زمان لازم برای استفاده، حتی در ابررایانه‌های امروزی غیرممکن است.

الگوریتم تولید کلید عمومی و خصوصی، پیچیده‌ترین بخش رمزنگاری RSA است. دو عدد اول بزرگ p و q، با استفاده از الگوریتم تست اولیهٔ رابین-میلر تولید شده و عدد n با ضرب p در q، به دست می‌آید. این عدد به عنوان پیمانه محاسبات برای رمزگذاری و رمزگشایی مورد استفاده قرار می‌گیرد و طول آن که معمولاً با بیت بیان می‌شود، طول کلید نام دارد.

یک عدد e (کوچکتر از n) طوری انتخاب می‌شود که نسبت به (p-1).(q-1) اول باشد. کلید عمومی عبارت است از اعداد n و e. سپس عدد d که وارون ضربی e به پیمانه (p-1).(q-1) است محاسبه می‌شود. کلید خصوصی نیز از p، q و d تشکیل شده است.

الگوریتم RSA چگونه کار می‌کند؟

آلیس کلیدهای RSA خود را، با انتخاب دو عدد اول تولید می‌کند. برای مثال p=11 و q=13 و n=p×q=143 است. بنابراین (p-1).(q-1) برابر است با 120.

او 7 را به عنوان e انتخاب و مقدار d را با استفاده از الگوریتم اقلیدسی توسعه یافته، برابر 103 محاسبه می‌کند.

باب می‌خواهد یک پیام به نام M را به شکل رمزگذاری شده به آلیس بفرستد؛ بنابراین کلید عمومی RSA او، (e , n) را دریافت می‌کند که در این مثال (143 ، 7) است. فرض کنید پیام متن سادهٔ او، عدد 9 است؛ متن‌رمز C، مطابق زیر به دست می‌آید:

M^e mod n = 9^7 mod 143 = 48 = C

وقتی آلیس پیام باب را دریافت می‌کند، آن را با استفاده از کلید خصوصی RSA خود به صورت زیر رمزگشایی می‌کند:

C^d mod n = 48^103 mod 143 = 9 = M

RSA چگونه امن است؟

امنیت RSA بر دشواری محاسباتی فاکتورگیری اعداد صحیح بزرگ، متکی است. با افزایش قدرت محاسباتی و کشف الگوریتم‌های فاکتورسازی کارآمدتر، توانایی فاکتورسازی اعداد بزرگتر نیز افزایش می‌یابد.

قدرت رمزگذاری مستقیماً با اندازهٔ کلید، مرتبط است. دوبرابر کردن طول کلید می‌تواند افزایش تصاعدی در استحکام ایجاد کند، اگرچه عملکرد را مختل می‌کند. کلیدهای RSA، معمولاً 1024، یا 8204 بیتی هستند، اما کارشناسان معتقدند که کلیدهای 1024 بیتی، دیگر در برابر همهٔ حملات، کاملاً ایمن نیستند. به همین دلیل است که دولت‌ها و برخی صنایع به سمت حداقل طول کلید 2048 بیتی حرکت می‌کنند.

با جلوگیری از یک پیشرفت پیش‌بینی‌نشده در محاسبات کوانتومی، سال‌ها طول می‌کشد تا کلیدهای طولانی‌تری مورد نیاز باشد، اما رمزنگاری منحنی بیضی (ECC)، به عنوان جایگزینی برای RSA، برای اجرای رمزنگاری کلید عمومی، مورد توجه بسیاری از کارشناسان امنیتی قرار گرفته است. و می‌تواند کلیدهای رمزنگاری سریعتر، کوچکتر و کارآمدتر ایجاد کند.

سخت‌افزار و نرم‌افزار مدرن برای ECC آماده هستند و احتمالاً محبوبیت آن افزایش می‌یابد. این می‌تواند امنیت معادل، با قدرت محاسباتی کمتر و استفاده از منبع باتری ارائه دهد، که آن را برای برنامه‌های تلفن همراه مناسب‌تر از RSA، می‌کند.

تیمی از محققان که شامل آدی شامیر، یکی از مخترعان RSA بود؛ با موفقیت یک کلید RSA 4096 بیتی را با استفاده از تحلیل رمزی صوتی، ایجاد کردند.

آسیب پذیری‌های RSA

وقتی سازمان‌ها از اعداد تصادفی ضعیف استفاده می‌کنند، اعداد اول ایجاد شده توسط آن‌ها، بسیار آسان‌تر قابل فاکتور هستند، در نتیجه زمان آسان‌تری به مهاجمان برای شکستن الگوریتم می‌دهند.

کلیدهای RSA الزامات خاصی در رابطه با تولید خود دارند، اگر اعداد اول خیلی نزدیک باشند، یا اگر یکی از اعدادی که کلید خصوصی را تشکیل می‌دهند خیلی کوچک باشد، آنگاه می‌توان کلید را خیلی راحت‌تر حل کرد.

حملات کانال جانبی

حملات کانال جانبی روشی برای حمله است، که برخلاف خود الگوریتم، از مزیت سیستمی که الگوریتم رمزگذاری را اجرا می‌کند ، استفاده می‌کند. مهاجمان می‌توانند قدرت مورد استفاده را تجزیه و تحلیل کنند، از تحلیل پیش‌بینی شاخه استفاده کنند، یا از حملات زمان‌بندی، برای یافتن راه‌هایی برای تعیین کلید مورد استفاده در الگوریتم، استفاده کنند؛ بنابراین داده‌ها را به خطر می‌اندازند.

RSA بیشتر به دلیل محصول SecurID خود شناخته شده است، که احراز هویت دو مرحله‌ای را برای صدها فناوری با استفاده از توکن‌های سخت‌افزاری که کلیدها را در فواصل زمانی معین، توکن‌های نرم‌افزاری و کدهای یک‌بار مصرف می‌چرخانند، ارائه می‌کند. در سال 2016، RSA پلتفرم SecurID را با نام RSA SecurID Access تغییر نام داد.

مارگارت همیلتون

نویسندگان:عسل بیگدلی, کیمیا بارسم, پانیذ کیانی

مارگارت همیلتون (Margaret Hamilton) مهندس نرم‌افزار و دانشمند کامپیوتر آمریکایی، در تاریخ 17 آگوست 1936 متولد شد.
او در رشتهٔ ریاضیات از دانشگاه میشیگان فارغ التحصیل شد، و در سال 1958 مدرک ریاضیات و فلسفه را از کالج ارلهام دریافت کرد. علاوه بر تحصیلات دانشگاهی، او در دبیرستان‌ها به طور مختصر ریاضیات و فرانسه تدریس می‌کرد، تا از تحصیلات سطح اول همسرش در هاروارد حمایت کند و بعداً خودش مدرک سطح دوم را دنبال کند.

بعد از آن، او به بوستون نقل مکان کرد و در دانشگاه براندیس در زمینهٔ ریاضیات محض تحقیق کرد. در آنجا، حرفهٔ او به‌طور حدی شروع شد، در سال 1960، هنگامی که یک شغل موقت در موسسهٔ فناوری ماساچوست (MIT)، برای توسعهٔ نرم افزار پیش بینی آب و هوا برای کامپیوترهای LGP-30و PDP-1 برای پروژه‌ای توسط پروفسور ادوارد نورتون لورنز از بخش هواشناسی به دست آورد.

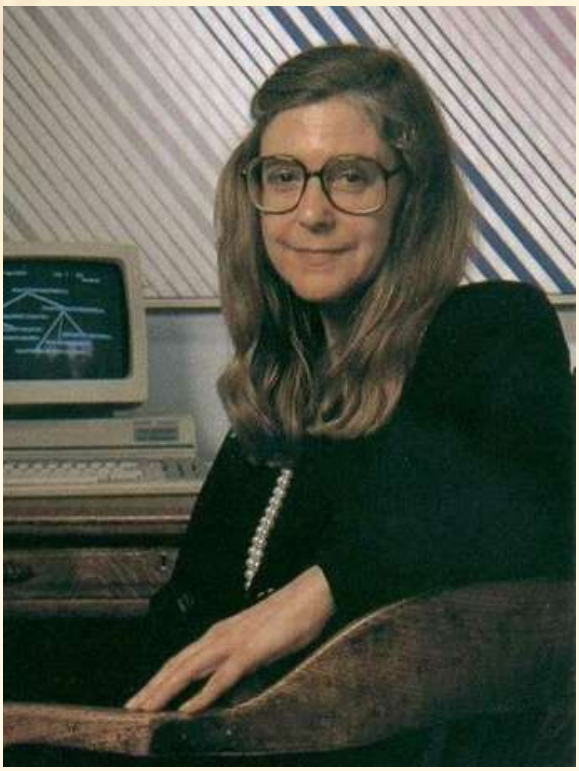
مارگارت همیلتون به عنوان یکی از پیشگامان صنعت نرم‌افزار، شناخته می‌شود و نقش بسیار مهمی در توسعه و طراحی نرم‌افزار مأمور ماژول راکت کنترل (Apollo Guidance Computer) برای مأموریت‌های فضایی آپولو داشت.

در دههٔ 1960، همیلتون به همراه تیمی از مهندسان در شرکت ماساچوست نایشنال لبوراتوری (MIT)، به طراحی و توسعهٔ نرم‌افزار مأمور ماژول راکت کنترل آپولو، مشغول شد. این نرم‌افزار مسئول کنترل سیستم هدایت و کنترل ماژول‌های فضایی آپولو در طول مأموریت‌های فضایی بود. زمانی که مأموریت آپولو 11 برای فرود روی ماه انجام شد، نرم‌افزار مارگارت همیلتون با موفقیت کار کرد و نقش حیاتی در فرود نرم و ایمنی فضانوردان داشت.

با گرامیداشت یاد این زن خارق‌العاده و تأثیر محو نشدنی او در اکتشافات فضایی، فرصتی بی‌نظیر برای همه علاقه‌مندان به فناوری و دوستداران تاریخ وجود دارد. سورس کد اصلی برنامه "هدایت آپولو 11"، که توسط مارگارت و تیمش توسعه داده شده است، حفظ شده و در GitHub در دسترس عموم قرار گرفته است. علاقه‌مندان می‌توانند با استفاده از لینک زیر در مرورگر خود، به سورس کد آپولو 11 دسترسی داشته باشند.
لینک گیت هاب:
https://github.com/chrislgarry/Apollo-11

مارگارت همیلتون برای ارائهٔ توصیف ریاضی دقیق از نرم‌افزار، و ایجاد روش‌های جدید برای آزمون و تحلیل نرم‌افزار شناخته می‌شود. او همچنین در زمینهٔ مدیریت پروژه‌های نرم‌افزاری فعالیت کرده و در سال 1986 شرکت خود را با نام Hamilton Technologies Inc تأسیس کرد.

مارگارت همیلتون به دلیل پیشروی و دستاوردهایش در حوزهٔ نرم‌افزار و کامپیوتر، مجموعه‌های زیادی از جوایز و افتخارات را دریافت کرده است. او یکی از افرادی است که تأثیر قابل توجهی بر روی توسعهٔ نرم‌افزار و صنعت فضایی داشته است؛ و به عنوان یکی از زنان برتر در زمینهٔ فناوری شناخته می‌شود.



01100

علاوه بر نقش مهم مارگارت همیلتون در توسعهٔ نرم‌افزار مأمور ماژول راکت کنترل آپولو، او در زمینه‌های دیگر نیز موفقیت‌های قابل توجهی داشته است. در ادامه، به برخی از این موفقیت‌ها اشاره می‌کنیم:

توسعهٔ مفهوم نرم‌افزار سیستمی: مارگارت همیلتون به طور فعال در توسعهٔ مفهوم نرم‌افزار سیستمی (System Software)، مشارکت داشت. او به همراه تیمش، معماری و طراحی سیستم عامل را توسعه داد و از مفاهیم نوآورانه‌ای همچون "مدیریت منابع" و "جریان کنترل" برای مدیریت و کنترل پردازش‌ها استفاده کرد.

طراحی و توسعهٔ نرم‌افزار هواپیماهای جنگی

پس از مأموریت‌های فضایی آپولو، مارگارت همیلتون به شرکت MITRE پیوست و در پروژه‌های مختلف مربوط به هواپیماهای جنگی، فعالیت کرد. او در زمینهٔ طراحی و توسعهٔ نرم‌افزارهایی برای هواپیماهایی همچون F-117 Nighthawk و F/A-18 Hornet دستاوردهای قابل توجهی داشت.

تأسیس شرکت Hamilton Technologies Inc

در سال 1986، مارگارت همیلتون شرکت خود را با نام Hamilton Technologies Inc. تأسیس کرد. این شرکت به طراحی و توسعهٔ نرم‌افزار و راه حل‌های نرم‌افزاری در حوزه‌های مختلف از جمله فضا، طبیعت، سلامتی و مدیریت پروژه می‌پرداخت.

تدریس و آموزش

مارگارت همیلتون فعالیت‌های آموزشی و تدریس نیز داشته است. او به عنوان استاد مهمان در دانشگاه‌ها و مؤسسات آموزشی مختلف، شرکت کرده و در انتشارات متعددی مقالات و کتاب‌های مرتبط با نرم‌افزار و کامپیوتر منتشر کرده است.

پروژه SAGE: مارگارت همیلتون در سال 1961 تا 1963 بر روی این پروژه مشغول به کار شد. او جزو یکی از برنامه‌نویسانی بود که برنامهٔ نرم‌افزاری اولین کامپیوتر AN/FSQ-7 را نوشتند، که وظیفهٔ آن شناسایی هواپیماهای دشمن (غیردوستانه) بود. این پروژه توسط مؤسسهٔ فناوری ماساچوست با نام پروژهٔ گردباد، شروع شد. که در ابتدا قرار بود برای پیش‌بینی آب و هوا به کار گرفته شود؛ اما مدتی بعد برای استفاده‌های نظامی در نیروی دفاع هوایی آمریکا، توسعه داده شد.

همیلتون دربارهٔ نقش خود در این برنامهٔ نظامی چنین می‌گوید:

وقتی شما در یک سازمان مشغول به کار می‌شوید، به هر کدام از شما قسمتی از یک برنامهٔ اصلی را برای برنامه‌نویسی می‌دهند این قسمت‌ها به تنهایی توسط هیچ‌کس قابل شناسایی و اجرا نخواهند بود؛ بنابراین من نمی‌دانستم درحال برنامه‌نویسی چه چیزی هستم، حتی زبان‌های به کار رفته در برنامه‌ای که من مسئول نوشتن آن بودم یونانی و زبان لاتین بود.

توسعهٔ نرم افزار آپولو 11

پس از کار بر روی SAGE، همیلتون به آزمایشگاه چارلز استارک درپیر در MIT پیوست، که در آن زمان بر روی مأموریت‌های آپولو کار می کرد. او در مدت کوتاهی مدیر و سرپرست توسعهٔ نرم‌افزار برنامه‌های Apollo و Skylab شد. در ناسا، تیم همیلتون مسئول توسعهٔ نرم‌افزاری بود که کپسول‌های برنامهٔ آپولو را در جهت‌یابی و فرود روی ماه و همچنین نسخه‌های مختلف آن را که در پروژه‌های بعدی دیگر از جمله Skylab مورد استفاده قرار می‌گرفت، هدایت می‌کرد.

پیشرفت واقعی همیلتون در انتخاب‌های طرح هوشمندانهٔ او، در مورد سیستم عامل J.Halcombe Laning بود که بسیار مهم واقع شد. در واقع، در یک لحظهٔ حساس در مأموریت آپولو 11، آینده‌نگری همیلتون و تیمش بود که مانع از شکست مأموریت شد.

همه نمی‌دانند که تنها سه دقیقه قبل از برخورد فرودگر به سطح ماه، چندین زنگ هشدار به دلیل بارگذاری بیش از حد رایانه با داده‌های دریافتی به راه افتاد. این به این دلیل اتفاق افتاد که سیستم رادار rendezvous ، که درهنگام فرود ضروری نبود، در حال به روزرسانی یک شمارنده بود و فضای کامپیوتر را دریک چرخه اشغال می‌کرد.

با این حال، به لطف برنامه‌ریزی اولویت‌های پیشگیرانه که توسط مارگارت و همکارانش ایجاد شد، فرآیندهای درگیر در فرود، با اولویت بالاتر، فرآیندهای با اولویت پایین‌تر را قطع کردند. این اجازه داد تا کنترل کامل مأموریت بازپس گرفته شود و این نقص بعداً به یک چک لیست نادرست نسبت داده شد.

مارگارت همیلتون این وضعیت را اینگونه توصیف می‌کند:

"به دلیل خطا در چک لیست دستی، سوئیچ رادار rendezvous در موقعیت اشتباه قرار گرفته بود. این باعث شد که کامپیوتر سیگنال‌های اشتباهی دریافت کند. در نتیجه، رایانه ملزم به انجام تمام عملکردهای فرود معمولی خود بود؛ در حالی که بار اضافی از داده‌های جعلی دریافت می‌کرد که 15 درصد از منابع آن را تحلیل می‌کرد. رایانه (یا بهتر است بگوییم، نرم‌افزاری که در حال اجرا بود) به اندازهٔ کافی هوشمند بود چون تشخیص داد که از آن خواسته شده است تا کارهای بیشتری را نسبت به توانش انجام دهد. سپس زنگ خطری را ارسال کرد که به فضانوردان می‌گفت: «من با وظایفی بیش از آنچه که در این لحظه می‌توانم انجام دهم پربار شده‌ام، و فقط مهم‌ترین آنها را ادامه می‌دهم». یعنی موارد مورد نیاز برای فرود... در واقع، کامپیوتر به گونه‌ای برنامه‌ریزی شده بود که کاری بیش از تشخیص شرایط خطا انجام دهد. مجموعهٔ کاملی از روش‌های بازیابی گنجانده شد. اقدامی که در این مورد انجام شد حذف فرآیندهای با اولویت پایین تر و بازیابی موارد مهم‌تر بود... اگر کامپیوتر این مشکل را تشخیص نمی‌داد و مطابق با آن واکنش نشان نمی‌داد، من شک دارم که آپولو 11 فرود موفقیت‌آمیزی روی ماه می‌داشت."
-مارگارت همیلتون

در پایان، ماجرای زندگی قابل توجه مارگارت همیلتون به عنوان یک دانشمند پیشگام کامپیوتر و مشارکت‌های اساسی او در مأموریت آپولو 11 برای همیشه به عنوان نمادی از فداکاری، نوآوری و نبوغ مورد تجلیل قرار گرفت. از روزهای اولیهٔ او در ریاضیات و فلسفه، تا کار پیشگامانه اش در MIT و ناسا، درخشش و طرز فکر آینده‌نگر همیلتون راه را برای سفر تاریخی بشریت به ماه هموار کرد.

کتاب

نویسندگان:ریحانه محمدزاده،ماناسامانی،سارا بهبودی

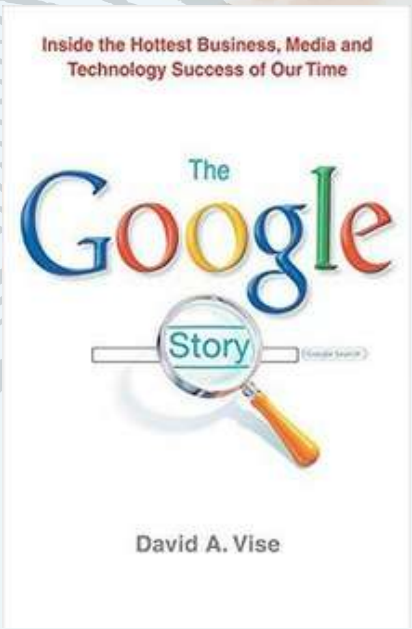
The Google Story

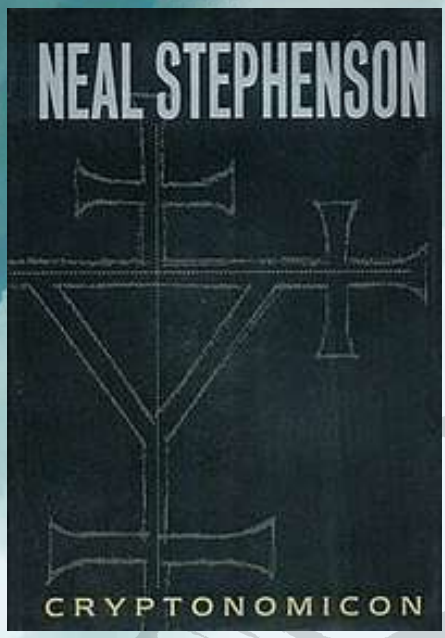
نمره *Goodreads : 3.9/5*

رای کاربران گوگل: *82* درصد کاربران گوگل این کتاب را دوست داشتند.

داستان گوگل

کتابی نوشتهٔ دیوید ویز و مارک مالسید است، که در *15* نوامبر *2005* منتشر شد. این کتاب نگاهی عمیق به تأسیس گوگل و دلیل منحصربه فرد بودن آن در میان شرکت‌های فناوری اطلاعات، دارد. همچنین دربارهٔ بنیانگذاران شرکت و فرهنگی که گوگل به آن معروف است، بحث می‌کند.





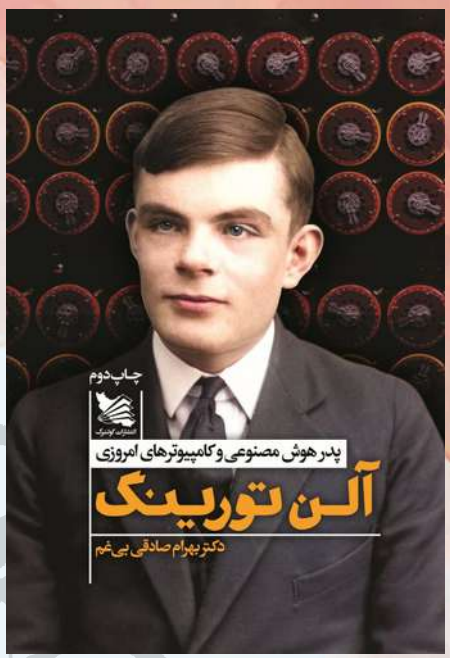
Cryptonomicon

نمره Goodreads : 4.2/5

رای کاربران گوگل: 90 درصد کاربران گوگل این کتاب را دوست داشتند.

کریپتونامیکان

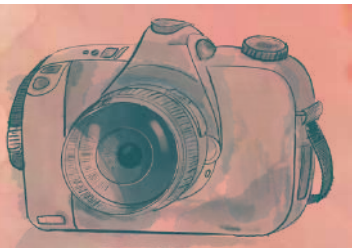
رمانی از نیل استفنسون، نویسندهٔ آمریکایی است که در سال ۱۹۹۹، در دو دورهٔ زمانی مختلف اتفاق می‌افتد. یک گروه از شخصیت‌ها، رمزگشایان متفقین دوران جنگ جهانی دوم، عوامل فریب تاکتیکی وابسته به دولت، مدرسهٔ سایفر در بلچلی پارک (بریتانیا) و شخصیت‌های نظامی و زیرک هستند. روایت دوم در اواخر دههٔ 1990 اتفاق می‌افتد، با شخصیت‌هایی که (تا حدی) از نوادگان دوره‌های زمانی قبلی هستند که از فناوری رمزنگاری، مخابراتی و رایانه‌ای، برای ساختن یک پناهگاه دادهٔ زیرزمینی در سلطنت خیالی "کیناکوتا" استفاده می‌کنند. هدف آن‌ها تسهیل بانکداری اینترنتی ناشناس، با استفاده از پول الکترونیکی و (بعداً) ارز دیجیتالی طلا، با هدف بلندمدت توزیع رسانه‌های آموزش و پیشگیری از هولوکاست (HEAP) برای آموزش به جمعیت‌های هدف نسل کشی، در مورد جنگ تدافعی است.



آلن تورینگ

کتاب آلن تورینگ پدر هوش مصنوعی و کامپیوترهای امروزی
نوشتهٔ دکتر بهرام صادقی بی غم
کتاب آلن تورنیگ نوشتهٔ دکتر صادقی بی غم، عضو هیئت علمی دانشگاه الزهرا و استاد دانشکدهٔ علوم ریاضی، در سال *1400* منتشر شد.
نام "آلن تورینگ" به عنوان "پدر هوش مصنوعی و کامپیوترهای امروزی" بارها در محافل علمی، کنفرانس‌های دانش‌بنیان و حتی سینمای هالیوود به گوش رسیده است. به جهت اهمیت کار "آلن تورینگ" و تاثیری که پژوهش‌های وی در قامت "پدر هوش مصنوعی و کامپیوترهای امروزی" داشته، افراد زیادی به زندگی او پرداخته، و تلاش کرده‌اند تا گوشه‌ای از اقدامات وی را به فراخور مخاطبی که قصد آشنا شدن با "آلن تورینگ" را دارد، برجسته نمایند. این بار دکتر "بهرام صادقی بی غم" به سراغ این دانشمند بزرگ و تأثیرگذار رفته و در کتاب صد و شصت صفحه‌ای پیش رو که به همت انتشارات گوتنبرگ به عرصهٔ چاپ رسیده، زندگی او را زیر ذره‌بین برده است.

"آلن ماتیسون تورینگ" در طول زندگانی کوتاه خود، پستی و بلندی‌های بسیاری را تجربه کرد که بخشی از آن، ناشی از انتخاب‌های شخصی وی و بیشتر آن به دلیل هوش سرشار و موقعیت‌هایی بود، که استفاده از این هوش برایش به ارمغان می‌آورد. او بخش اعظم زندگی خود را به علم ریاضی اختصاص داده بود و ریاضیات، مهم‌ترین رانه‌ای بود که به عشق آن، هر روز را به شب می‌رساند. این علاقهٔ وصف نشدنی به ریاضی و هوش بالای او در این زمینه، ترکیب شگفت‌آوری را ایجاد کرد که به وسیلهٔ آن "آلن تورینگ" توانست پایه‌های اختراعی عظیم را بنا کند. آن اختراع، کامپیوتر بود که حیات بشری را به صورت اساسی متحول کرد و هنوز هم به پیشرفت توقف‌ناپذیر خود ادامه می‌دهد. دکتر "بهرام صادقی بی غم" در کتاب پیش رو، با پرداختن به زندگی "آلن تورینگ" از بدو تولد تا مرگ وی، دستاوردهای او را به عنوان "پدر هوش مصنوعی و کامپیوترهای امروزی" بر می‌شمارد و ریشه‌های علم کامپیوتر و پیوندش با ریاضیات را در کار این دانشمند نخبه، نشان می‌دهد.



فیلم

نویسندگان:ریحانه محمدزاده،ماناسامانی،سارا بهبودی

The bit player

نوازندهٔ بیت یا بیت پلیر (The Bit Player) یک مستند بیوگرافی محصول سال 2018 کشور آمریکا به کارگردانی مارک لوینسون است، که توسط کمپانی Institute of Electrical and Electronics Engineers منتشر شد. نویسندگی این مستند را نیز مارک لوینسون برعهده داشته، و افرادی چون جان هاتن، جودیت آیوی، کالیسوا بروستر، آندرو پاستیدس، اولیویا جیلیات، وی.جی اسکارپاسی، تولیا ناپولیتانو، آدام هلر و ...در آن حضور دارند. همچنین این مستند که به مناسبت گرامیداشت تولد صد سالگی کلود شانون ریاضیدان، مهندس الکترونیک و رمزنگار معروف آمریکایی ساخته شده است، اولین بار در تاریخ ۲۹ می سال ۲۰۱۸ میلادی، در جشنوارهٔ جهانی علوم به نمایش درآمد. سپس در ۲۶ ژوئن سال ۲۰۲۰، در کشور آمریکا به صورت اینترنتی پخش گردید. جالب است بدانید شانون با مقاله‌ای که در سال ۱۹۴۸ میلادی منتشر کرد، نظریهٔ اطلاعات را بنیان نهاد و به شهرت رسید.

The Imitation Game

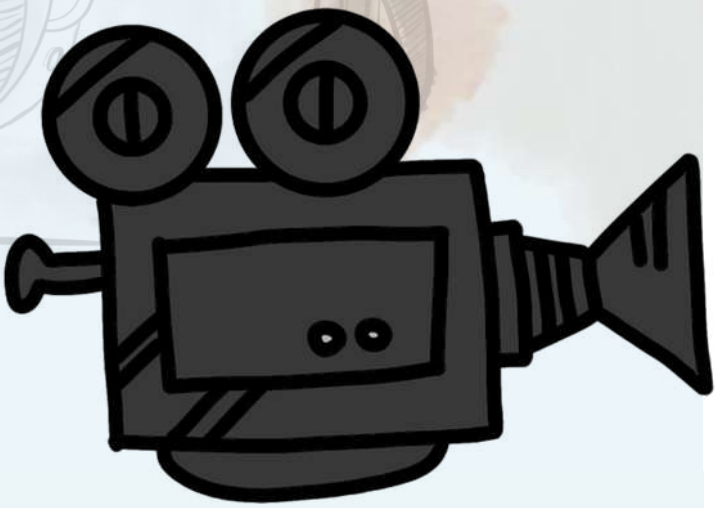
این فیلم زندگینامهٔ "آلن تورینگ" پدر هوش مصنوعی، و مراحل شکستن کد انیگما در جنگ جهانی دوم را به خوبی به تصویر می‌کشد. او با کمک ریاضیدانان همکارش، ماشینی برای شکستن کدها می‌سازد و با چالش‌های خاصی در این راه مواجه می‌شود. این فیلم نامزد دریافت هفت جایزهٔ اسکار و نامزد و برندهٔ بسیاری دیگر از جوایز نفیس فیلم‌سازی شده است.

همچنین معروف است که شانون در سال ۱۹۳۷ و در سن ۲۱ سالگی که دانشجوی کارشناسی ارشد در دانشگاه ام‌آی‌تی بوده، نظریهٔ رایانه‌های دیجیتال و مدارهای دیجیتال را پایه گذاشته است. وی در پایان‌نامهٔ خود نشان داد که با پیاده‌سازی الکتریکی منطق دو-دویی، میتوان هر مسئلهٔ منطقی و عددی را حل کرد. چنین ادعا شده که این پایان‌نامه، مهم‌ترین پایان‌نامهٔ کارشناسی ارشد تاریخ بوده است. در زمان جنگ جهانی دوم، شانون به پژوهش در رمزکاوی پرداخت و پس از جنگ نیز به رمزشکنی روی آورد.

اگر به رمزنگاری علاقه‌مند هستید حتما اسم کلود شانون واصل شانون را شنیده‌اید.

The social dilemma

معضل اجتماعی، نام فیلمی مستند و اجتماعی محصول سال ۲۰۲۰، به کارگردانی جف اورلوسکی می‌باشد. این مستند ظهور رسانه‌های اجتماعی، و آسیبی که به جامعه می‌زند را بررسی می‌کند و بر روی نحوهٔ بهره‌گیری این رسانه‌ها از کاربرانشان، از طریق سیستم‌های نظارتی نظام سرمایه‌داری و استخراج اطلاعات آن‌ها تمرکز می‌کند. معضل اجتماعی مواردی از جمله نحوهٔ طراحی این رسانه‌ها به شکلی که منجر به اعتیاد می‌شوند ، استفادهٔ از آن‌ها در سیاست ، تأثیرش بر روی سلامت روانی ، من جمله سلامت روانی جوانان و افزایش نرخ خودکشی در نوجوانان و در نهایت نقش این رسانه‌ها در پراکندن شایعه‌ها و تئوری‌های توطئه‌برانگیز در جوامع که نظیرش را بسیار دیده‌ایم را، مورد تحقیق و بررسی قرار می‌دهد.
این مستند شاید حتی شما را ترغیب کند که هر حسابی که در صفحات اجتماعی دارید را پاک کنید دیدن یکبار آن خالی از لطف نیست .



پادکست

نویسندگان:ریحانه محمدزاده،ماناسامانی،سارا بهبودی

پادکست Turing

این پادکست فارسی، به طور اختصاصی در مورد رشتهٔ علوم کامپیوتر صحبت می‌کند، در حال حاضر این پادکست دارای سه فصل هست:

فصل یک: ابتدا یک معرفی کلی در رابطه با رشته انجام شده، سپس به سراغ معرفی درس‌های پایه و علت مطالعه آن‌ها پرداخته است و نکات کنکوری در آن ذکر شده است.

فصل دو: در رابطه با موضوعات روز مثل یادگیری ماشین و هوش مصنوعی پرداخته شده، و در مورد گرایش‌های علوم کامپیوتر صحبت و معرفی شده است.

فصل سه: در این فصل در مورد درس‌های تخصصی‌تر و علت مطالعهٔ آنها صحبت شده است.

در کل این پادکست از جنبه‌ای که به زبان فارسی است، می‌تواند یک دید مناسب نسبت به رشتهٔ علوم کامپیوتر به کسانی که می‌خواهند در این رشته تحصیل کنند و یا ادامه تحصیل بدهند منبع مناسبی برای شناخت اولیه نسبت به رشتهٔ علوم کامپیوتر است.

پادکست Computer Science

این مجموعه میزبان قسمت‌هایی است که توسط دپارتمان علوم کامپیوتر دانشگاه آکسفورد، که یکی از قدیمی‌ترین بخش‌های علوم کامپیوتر در این دانشگاه است؛ منتشر شده است.

این مجموعه تحقیقات و تدریس در سطح جهانی این بخش را با ارائه گفتگوهایی که موضوعاتی مانند زیست‌شناسی محاسباتی، محاسبات کوانتومی، زبان شناسی محاسباتی، سیستم‌های اطلاعاتی، تأیید نرم‌افزار و مهندسی نرم‌افزار را در بر می‌گیرد، منعکس می‌کند.

پادکست Details cast

این پادکست که توسط نوشین بهشتی کارشناس ارشد نرم‌افزار کامپیوتر تهیه شده، شامل قسمت‌های بسیار مفید در رابطه با برنامه‌نویسی است.

برخی از موضوعات کار شده در این پادکست به شرح زیر است :

ep29: توسعه‌دهندهٔ فرانت اند کیست ؟ به همراه مریم مختاری از شرکت تپسی

ep30: توسعه‌دهندهٔ بک اند کیست ؟ به همراه سینا احمدپور از شرکت ابراوران

ep93: هوش مصنوعی رازها و واقعیت‌ها دربارهٔ آینده به همراه اسماعیل گوهری

ep106: چالش‌های مصاحبهٔ شغلی و کار ریموت با رضا امینی

پادکست طبقه شانزدهم

یک پادکست گفتگو محور، که در هر اپیزود با انسان‌های موفق در حوزه‌های مختلف گفتگو می‌کند و اکثر اپیزودهای آن، به مصاحبه با برنامه‌نویسان موفق ایرانی در سراسر دنیا اختصاص دارد.

(برای مثال اپیزود اخر آن، Ep129 که گفتگو با امیرمحمد وکیلی و صحبت دربارهٔ باگ هانتینگ و کشف باگ‌ها، از سرویس‌ها و شرکت‌هاست.

و یا Ep124 که مصاحبه با مجید هاشمیان، طراح ارشد محصول گوگل در کالیفرنیاست. این اپیزود در رابطه با سوالات مصاحبه‌های شرکت‌های بزرگی همچون گوگل و فیس بوک است.)

مسابقه الگوریتمی

نویسنده:سحر حقیقت

فرض کنید شما والدین عالی هستید و میخواهید به فرزندانتان چند کیک بدهید، اما باید به هر فرزند حداکثر یک کیک بدهید. هر فرزند i یک فاکتور تمایل g[i] دارد که حداقل اندازه‌ای از کیک است که فرزند می‌تواند با آن راضی باشد؛ همچنین هر کیک j یک اندازه s[j] دارد.
اگر s[j]≥ g[i] باشد، می‌توانیم کیک j را به فرزند i اختصاص دهیم و فرزند i راضی خواهدبود. هدف شما افزایش تعداد فرزندان راضی شده‌ی خود است و بیشترین تعداد را باید خروجی دهید.

مثال 1

ورودی:

g = [1, 2, 3] , s = [1, 1]

خروجی: 1
توضیحات: شما 3 فرزند و 2 کیک دارید. فاکتورهای تمایل 3 فرزند به ترتیب 1، 2، 3 هستند. شما 2 کیک دارید، اما اندازه‌ی هر دوی آنها 1 است، بنابراین تنها می‌توانید فرزندی که فاکتورهای تمایلش 1 است را راضی کنید. بنابراین باید 1 را خروجی دهید.

مثال 2

ورودی:

g = [1, 2] , s = [1, 2, 3]

خروجی: 2
توضیحات: شما 2 فرزند و 3 کیک دارید. فاکتورهای تمایل 2 فرزند به ترتیب 1 و 2 هستند. شما 3 کیک دارید و اندازه‌هایشان کافی است تا فرزندان را راضی کند،
بنابراین باید 2 را خروجی دهید.

محدودیت‌ها: 1
1 ⩽ طول g ⩽ 3 × 104
0 ⩽ طول s⩽ 3 × 104

جواب آخرتان را از راه های ارتباطی ما در صفحه آخر و یا با استفاده از گیت هاب نشریه برای ما ارسال کنید

تلگرام https://t.me/AlzCSSJournal01
لینکدین https://www.linkedin.com/company/zero-one-alzahra-csjournal/
ایمیل AlzCSSJournal01@gmail.com