مدرس: دکتر مجتبی رفیعی

نیمسال اول ۱۴۰۰-۱۴۰۱

ساختمان داده‌ها و الگوریتم‌ها

جلسه ۲ ساختمان داده‌ها و الگوریتم‌ها

نگارنده: نوش آفرین احمدی

۷ مهر ۱۴۰۰

فهرست مطالب

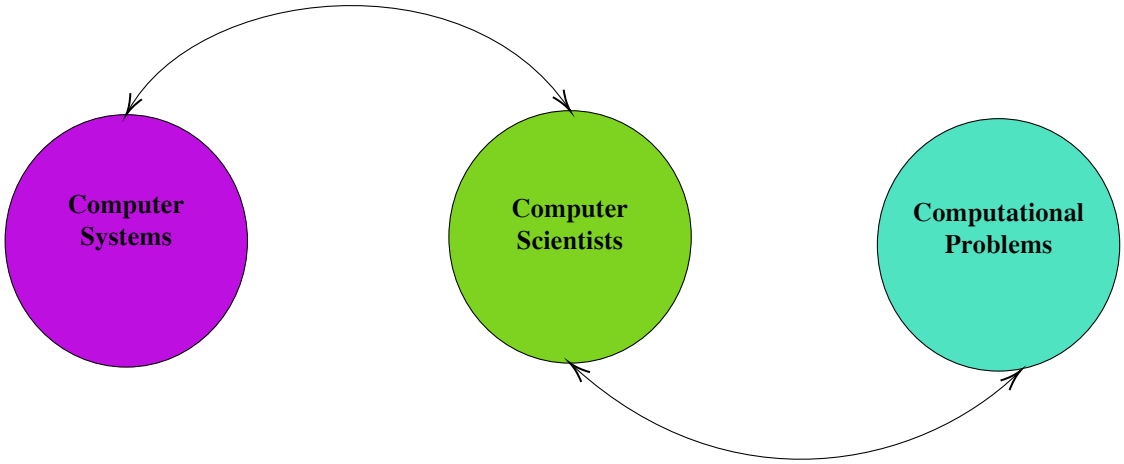
۱ جایگاه درس ساختمان داده‌ها و الگوریتم‌ها در علوم کامپیوتر

گرایش‌های مختلف مهندسی کامپیوتر، نظیر: مهندسی نرم‌افزار، هوش مصنوعی، معماری کامپیوتر یا گرایش‌های مطرح در علوم کامپیوتر، نظیر: سیستم‌های کامپیوتری، سیستم‌های هوشمند، گراف، ترکیبیات، رمزنگاری، همگی حول ارایه راه‌حل برای یک سری مسائل شکل گرفته‌اند. وظیفه اصلی یک متخصص علوم کامپیوتر (computer scientist) حل مسائل محاسباتی درحوزه‌های مختلف است. دانشمندان علوم کامپیوتر، ایده‌هایشان را برای حل یک مسئله در قالب یک الگوریتم مطرح و سپس آن را به یک برنامه تبدیل نموده و برای حل خودکار آن را به یک سیستم کامپیوتری می‌دهند. بنابراین، از یک دیدگاه کلی زیست‌بوم علوم کامپیوتر را می‌توان در سه مولفه اصلی زیر خلاصه کرد. شکل ۱، این سه مولفه اصلی را به همراه نحوه ارتباطشان به تصویر کشیده است.

۱. مسائل محاسباتی،

۲. دانشمندان وپژوهشگران علوم کامپیوتر،

۳. سیستم‌های کامپیوتری،



شکل ۱: تعامل بین مولفه‌های اصلی زیست‌بوم علوم کامپیوتر

همانطور که در شکل ۱ قابل مشاهده است، رابطه بین مسائل محاسباتی و دانشمندان حوزه علوم کامپیوتر، دوطرفه است بدین معنا که علاوه بر حل کردن مسئله، تولید کننده مسئله نیز هستند چون امکان دارد یک مسئله محاسباتی آنقدر پیچیده باشد که در طی فرایند حل کردن نیاز باشد آن را به مسئله‌های کوچکتر تبدیل کنیم و یا یک سری فرضیات در نظر بگیریم و با استفاده از فرضیات بتوانیم مسئله را حل کنیم.پس علاوه بر اینکه خودشان یک سری مسائل محاسباتی را حل می‌کنند، امکان دارد مسائل جدید محاسباتی را هم مطرح نمایند.

از طرف دیگر، همانطورکه در شکل ۱ نیز قابل مشاهده است، رابطه بین سیستم‌های کامپیوتری و دانشمندان علوم کامپیوتر نیز دوطرفه است. وقتی یک دانشمند علوم کامپیوتر برای سیستم کامپیوتری برنامه می‌نویسد سبب ارتقا خود سیستم‌های کامپیوتری هم می‌شود. یک سیستم کامپیوتری توانمندی و قابلیت‌های مشخصی دارد و بنابراین ممکن است وقتی دانشمند علوم کامپیوتر می‌خواهد مسئله‌ای را حل کند که کامپیوترهای امروزی برای حل این مسئله توانمندی های لازم را ندارند، به صورت تئوری به قضیه نگاه کند و یک مدل محاسباتی را در نظر بگیرد که مسئله در آن مدل به صورت کارایی قابلیت حل شدن داشته باشد.

ما در مبانی کامپیوتر یاد گرفتیم که چگونه شبیه یک متخصص علوم کامپیوتر فکر کنیم. یک سیستم کامپیوتری یک مجموعه قابلیت‌هایی دارد. به عنوان مثال، اگر یک ورودی خاصی بگیرد با توجه به توان پردازشی که دارد می تواند در یک زمان مشخص محاسبات را انجام دهد. با توجه به این قابلیت‌ها، ممکن است یک سری محاسبات داشته باشیم که زمان مورد نیاز آنها، سال‌ها طول بکشد، مثل مسائلی که در حوزه رمزنگاری و امنیت هستند. وقتی یک دانشمند علوم کامپیوتر می‌خواهد مسئله‌ای را حل کند، باید محدودیت‌ها و قابلیت‌های یک سیستم کامپیوتری آشنا بوده و مسئله را در قالب استانداردی با سایرین به اشتراک بگذارد به نحویکه آنها متوجه ایده‌ها و افکار او شوند. شما در مبانی کامپیوتر یاد گرفتید چگونه برای یک مسئله محاسباتی فلوچارت بکشید و شبه کد بنویسید و در نهایت آن را به یک برنامه تبدیل کنید و به سیستم کامپیوتر بدهید.

در چند جلسه آینده در حیطه مسائل محاسباتی صحبت خواهیم کرد. اینکه در این درس چه دست مسائلی مد نظر ما هستند و به بررسی تکنیک‌ها و الگوریتم‌هایی که برای حل این دست از مسائل محاسباتی وجود دارد می پردازیم، این که چند نمونه الگوریتم داریم و در این درس قرار است ما در کدام نوع از این الگوریتم ها متمرکز شویم خواهیم پرداخت.

سوال

آیا هر مسئله محاسباتی را می‌توان حل کرد؟

۲ مسائل محاسباتی

مسائل محاسباتی را می‌توان از حیث قابلیت حل‌پذیری به دو رده زیر تقسیم کرد:

۱) **غیرقابل حل**[1]: مسائلی هستند که برای آنها هیچ الگوریتمی نمی تواند وجود داشته باشد.

۲) **قابل حل**[2]: مسائلی که برای آنها الگوریتمی وجود دارد. این رده از مسائل خود به دو دسته زیر تقسیم می‌شود:

- مسائل قابل حل در تئوری (مسائل رام نشدنی[3]) : در عمل نمی‌توان این مسائل را برای نمونه‌های بزرگ به کار گرفت.
- مسائل قابل حل در عمل (مسائل رام شدنی[4]): در عمل قابل استفاده هستند.

در ادامه سعی بر آن است تا با ذکر مثال‌هایی برای هر یک از رده‌های مسائل محاسباتی، مطالب بیان شده به صورت عمیق‌تری درک شود.
مثال برای رده مسائل غیرقابل حل - مسئله توقف[5]. : فرض کنید برنامه‌ای در اختیار دارید که وظیفه‌اش این است که توصیف برنامه‌ای مثل P1 را به همراه ورودی آن مثلا X، به عنوان ورودی خود دریافت نموده و تشخیص دهد که آیابرنامه P1 روی ورودی X متوقف خواهد شد یا خیر؟ می‌توان نشان داد که هرگز نمی‌توان چنین برنامه‌ای نوشت. به عبارت دیگر الگوریتمی برای آن وجود ندارد و مسئله توقف یک مسئله غیر قابل حل است .

تجربیات لاتک‌نویسی

- برای تهیه گزارش، پیشنویس و فیلم بارگذاری شده را با دقت مطالعه و بررسی کنید. در صورت نیاز با جستجو در وب مطالب یا مثال‌های بیشتری را سعی کنید در گزارش خود لحاظ کنید.
- برای ایجاد نیم‌فاصله می‌بایست از ترکیب کلیدهای Shift+Space استفاده کرد.
  - **کلمات غلط:** می شود، می نماید، می گردد و ...
  - **کلمات صحیح :** می‌شود، می‌نماید، می‌گردد و ...
- کلمات داخل پرانتز باید دقیقا چسبیده به پرانتز باز و بسته باشد.
  - **کلمات غلط:** (علم)، (علم)، (علم)
  - **کلمه صحیح :** (علم)
- رعایت علایم نگارشی:
  - "کاما" باید چسبیده به کلمه قبل و با یک فاصله از کلمه بعد باشد.
  - "نقطه" پایان جمله می‌بایست چسبیده به آخرین کلمه جمله باشد.
  - وقتی نیاز به "بولت‌گذاری" دارید، جمله قبل از بولت‌گذای در صورت نیاز با دو نقطه (:) پایان پذیرد.
- حتما پس از پایان یافتن نگارش گزارش، متن را با دقت خوانده و سعی کنید از حیث یکپارچگی، پیوستگی و روان بودن مطالب آن را بررسی و در صورت نیاز اصلاحاتی را اعمال کنید.
- تصاویر و جداولی که در متن گزارش استفاده می‌کنید، حتما دارای عنوان باشد و در متن نیز به آنها ارجاع دهید.
- برای رسم شکل و گرفتن خروجی به صورت لاتک و تصویر، ابزار انلاین زیر توصیه می‌شود.

https://www.mathcha.io/editor

---

[1] Unsolvable
[2] Solvable
[3] Intractable
[4] Tractable
[5] Halting Problem