برنامه‌نویسی رقابتی

هدف این مقاله‌ی کوتاه اینه که تا حدودی دلایلی که باعث میشه تلاش کردن در حوزه‌ی مسابقات الگوریتمی و برنامه‌نویسی رقابتی (competitive programming یا sport programming) موضوع مهم و قابل تاملی باشه رو بررسی کنه.

برنامه‌نویسی رقابتی چی هست حالا؟

اگه خلاصه بخوام بگم، یک کانتست (مسابقه) که تعدادی برنامه‌نویس توش جمع میشن، یک سری سوال الگوریتمی که تا حالا ندیدن و معمولا یه داستانی داره رو می‌خونند و سعی میکنن یه الگوریتم (راه‌حل) براش پیدا (طراحی) کنند، کد بزنند و حلش کنن. نتیجه معمولا همون لحظه نمایش داده میشه و تیم‌ها متوجه میشن که سوال رو درست حل کردن یا نه. این نوع برنامه‌نویسی و حل مسئله، علاوه بر چالش‌های علمی و با ارزش، چالش‌های مهم دیگه‌ای مثل مدیریت زمان، کار تیمی و... را دارد.

موضوع برنامه‌نویسی رقابتی همیشه در همه‌ی دنیا در رده‌های سنی مختلف یکی از مهمترین موضوعات کامپیوتری به حساب می‌رفته. از زمان دبیرستان و المپیاد کامپیوتر گرفته تا زمان دانشگاه و ACM ICPC و مسابقات دیگه.
معتبرترین مسابقه‌ی برنامه‌نویسی رقابتی در دبیرستان مسابقه‌ی IOI(International Olympiad in Informatics) هست. این مسابقه در همه‌جای دنیا از جمله ایران بسیار جدی گرفته می‌شه. شرکت‌کنندگان مطرح این مسابقه تقریبا بدون استثنا در بهترین دانشگاه‌ها ادامه تحصیل میدن.

مسابقه‌ی ACM ICPC هم معتبرترین مسابقه‌ی برنامه‌نویسی دانشجویی به حساب میره که در اینجا هم شرکت‌کننده‌های مطرح در کمپانی‌های خوب و بزرگ دنیا مشغول کار میشن و یا در دانشگاه‌های مطرح ادامه تحصیل میدن.

در ایران هم شرکت‌های مطرح معمولا به دنبال جذب برنامه‌نویس‌های به اصطلاح ای‌سی‌امی بودن و هستن، مثلا شرکت کافه‌بازار درصد نسبتا زیادی از نیروی کارش رو بچه‌های المپیادی و ای‌سی‌امی تشکیل میدن.

سوال مهمی که اینجا مطرح میشه و سعی می‌کنیم تا حدی به اون پاسخ بدیم اینه که چرا هم جامعه‌ی آکادمیک و هم بازار کار و شرکت‌های بزرگ اینقدر به چنین موضوعی اهمیت میدن؟

برای جواب به این سوال کمی باید بررسی کنیم که ما از برنامه‌نویسی رقابتی چه چیزهایی به‌دست میاریم. به صورت کلی اگر بخوایم به این قضیه نگاه کنیم به نکات زیر می‌رسیم:

- یاد می‌گیریم که چیزی که توی ذهن داریم رو در زمان کوتاهی تبدیل به کد کنیم.
- الگوریتم‌ها و ایده‌های حل مسئله‌ی زیادی یاد می‌گیریم که جاهای مختلف کاربرد دارند.
- یاد می‌گیریم که در حالت کلی برای حل کردن مسئله چطوری باید به اون فکر کرد، به طور دقیق‌تر یاد می‌گیریم که چجوری برای حل مسئله‌های مختلف بهشون حمله کنیم.
- معمولا این نوع مسابقات تیمی هستن و کار تیمی یکی از مهمترین بخش‌های اون محسوب میشه، این باعث میشه کار تیمی رو بهتر یاد بگیریم. به عبارت دقیق‌تر، یاد می‌گیریم توانایی‌هامون رو در ارتباط با هم‌تیمی‌هامون در جهت پیشرفت تیم به صورت بهینه به اشتراک بگذاریم.

- به طور **معمول** (البته نه لزوما) یک برنامه‌نویس رقابتی نسبت به بقیه سریع‌تر به جواب مسائل (نه تنها مسائل الگوریتمی بلکه مسائل در حالت کلی) میرسه. دلیلش هم اینه که یادگرفته که چجوری داده‌های مهم مسئله رو پیدا و تحلیل کنه و چجوری به مسئله حمله کنه.
- یک برنامه‌نویس رقابتی به واسطه‌ی سوالات زیادی که حل کرده، با کد زدن راحت شده و معمولا بهتر کد میزنه. به طور دقیق‌تر، فقط به فکر به نتیجه رسیدن و جواب گرفتن از کد نیست و به صورت ناخودآگاه (شاید هم آگاه) موقع کد زدن به efficiency کدی که میزنه توجه داره.

حالا بیاید ببینیم محیط‌های آکادمیک و شرکت‌های معتبر و بزرگ دنیا به دنبال چه ویژگی‌هایی هستن و چرا این ویژگی‌ها براشون مهمه.

شرکت بزرگی مثل گوگل رو در نظر بگیرید، این شرکت تقریبا در همه‌ی زمینه‌هایی که کار می‌کنه پیشرو هست. پیشرو بودن به چه معناست؟ یعنی کارهایی انجام میده که مشابه‌ش انجام نشده و یا نیاز داره که اون کار رو بهتر انجام بده، بنابراین قاعدتا با مسائل و مشکلاتی روبه‌رو میشه که قبلا حل نشده ویا نیاز به بهینه سازی بیشتر داره. حالا بیاید خودمون رو بذاریم به جای گوگل، چه کسی رو استخدام می‌کنیم بر اساس نیازمون؟

بلد بودن زبان‌های برنامه‌نویسی، ابزارهای مختلف و فریموورک‌ها البته که مهم‌اند ولی در نهایت ویژگی که غالب میشه و مهمتر محسوب میشه توانایی حل و تحلیل مسئله است.

همین موضوع رو میشه به جامعه‌ی آکادمیک رو ریسرچ هم بسط داد. در نهایت هر کاری که می‌خوایم انجام بدیم به یک مسئله‌ای تبدیل میشه که باید حل و یا بهینه بشه.

یک برنامه‌نویس رقابتی که مسیر درستی رو پیش رفته باشه با صرف زمان کمتر نسبت به بقیه می‌تونه یک زبان برنامه‌نویسی، یک ابزار یا یک «هر چیزی» رو یاد بگیره (معمولا). چرا؟ دقیقا نمیدونم :)) ولی حل مسائل زیاد که توانایی حل مسئله و تحلیل مسئله رو در پی داره باعث میشه یادگیری سریع‌تر بشه.

نوع نگاه‌کردن به مسئله، حمله به اون و حل کردنش، ویژگی‌های مهمی هستن که مستقل از کاری که داریم می‌کنیم به اون‌ها نیاز پیدا می‌کنیم و بسیار مهم‌اند. این ویژگی‌ها تا حد بسیار زیادی اکتسابی و قابل یادگیری‌اند.

نکته‌ی مهم اینه که بدونیم یک‌راه خاص و منحصر به‌فرد برای رسیدن به این ویژگی‌ها وجود نداره و کارهای زیادی ممکنه باعث بشه این ویژگی‌ها کسب بشن، ولی یکی از راه‌های آزموده‌شده همین برنامه‌نویسی رقابتیه.

حالا کمی واضح‌تر شد که چرا شرکت‌های بزرگ و محیط‌های آکادمیک به برنامه‌نویس‌های رقابتی اهمیت ویژه‌ای میدن و حتی هر ساله هزینه‌ی زیادی برای برگزاری مسابقات خودشون مثل Google CodeJam، Facebook Hckercup و ... می‌کنند و چرا در مصاحبه‌های کاری شرکت‌های بزرگ دنیا مثل گوگل، فیسبوک، مایکروسافت و ...، و حتی شرکت‌های پیشرو کشور مثل کافه‌بازار معمولا مسائل الگوریتمی مطرح می‌شوند.

موضوع آخر که می‌خوام کمی راجع بهش صحبت کنم یادگیری و تلاش‌کردن در حالت کلیه. وقتی با دانشجوهای جدیدالورود یا همون صفری خودمون درمورد کارهایی که می‌تونن انجام بدن صحبت می‌کنم، چیزی که غالبا می‌شنوم اینه که «**وقت ندارم**!» ولی دوست دارند یاد بگیرن که مثلا خوب مسئله حل کنن و خوب کد بزنند، یعنی عملا دنبال فرمول و معجزه‌اند.

حالا از اینکه یه دانشجوی ترم یک و دو واقعا کار زیادی نداره که بکنه بگذریم، این جمله‌ی «وقت ندارم» در ۹۹ (عدد کثرت :)) درصد مواقع فقط و فقط بهونه‌ست، حالا یا برای فرار از کارهای جدید و یا دلایل مشابه. قاعدتا کار جدید نکردن، طبق روند ثابت درس خوندن و دانشگاه پیش رفتن و در حالت کلی تلاش نکردن برای چیزهای بزرگ‌تر(احتمالا سخت‌تر) و جدید کار ساده‌تریه. نمیشه با همه چیز خوردن رژیم گرفت، نمیشه در ۱۰۰ ساعت انگلیسی یاد گرفت و . . . در حالت کلی نمیشه تلاش نکرد و پیشرفت کرد، مستقل از میزان هوشی که در کارهای مختلف داریم. چیزی که قصد گفتنش رو دارم اینه که باید بفهمیم و باور داشته باشیم که برای ذره ذره پیشرفت (واقعی) باید تلاش کرد، زمان فهمیدن و درک کردن این موضوع همین الانه که تقریبا جای خوبی از راه هستیم.

هر چه گفته شد نظر و تجربه‌ی شخصی من بوده و قاعدتا وحی منزل نیست :)

در ادامه میتونید این چند نمونه رو بررسی کنید:

- https://www.google.com/amp/s/www.geeksforgeeks.org/toptalent-interview-divanshu-got-google-mountain-view/amp/
- https://www.youtube.com/watch?v=IVplO2XPxvU
- https://www.redgreencode.com/12-reasons-to-study-competitive-programming/