

هوالعلم



مبانی کامپیووتر و برنامه‌سازی

نیمسال اول سال تحصیلی ۱۴۰۲ – ۱۴۰۳

مسائل برنامه‌نویسی

فؤاد رضایی

مسئله 1: تجزیه اعداد صحیح به عامل‌های اول آنها

الگوریتم: در قسمت ابتدایی برنامه از الگوریتم آزمون تقسیم یا به اصطلاح (trial division) استفاده شده است که برای بررسی اینکه آیا یک عدد اول است یا خیر از عدد 2 تا حداقل سقف ریشه دوم آن عدد را پیمایش می‌کند. در قسمت دوم برنامه هم از همان الگوریتم استفاده شده است با این تفاوت که در این قسمت ما آزمون تقسیم را بر مقسم علیه‌های اول انجام میدهیم

برنامه: این برنامه به اندازه کافی خوانا نوشته شده است فقط لازم به ذکر است که خروجی کد ما در تابع `factorize` یک لیست متشکل از عامل‌های اول تشکیل دهنده آن عدد است. در تابع `UserInput` برای زیبایی بیشتر در خروجی از یک حلقه `for` استفاده شده است تا تعداد تکرار‌های هر عامل را به صورت توانی به ما نشان دهد.

مراجع: در این سوال از مرجعی استفاده نشد.

لازم به ذکر است که اولین بروز من برای تجزیه اعداد صحیح به عامل‌های اول ان برمی‌گردد به سوال شماره 3 Euler Project ک لینک ان را برا شما در همینجا قرار میدهم.

<https://projecteuler.net/problem=3>

مسئله 2: بسط فاکتوریلی اعداد طبیعی

الگوریتم: در ابتدا یک الگوریتم بازگشتی برای محاسبه فاکتوریل یک عدد با این منطق ریاضیات ک فاکتوریل اعداد صفر و یک برابر عدد 1 است. و برای محاسبه فاکتوریل های بیشتر از منطق فراخوانی تابع به صورت بازگشتی استفاده میشود. برای محاسبه بزرگترین فاکتوریل قبل از عدد معین هم از الگوریتم جست و جوی تکراری استفاده شده است.

برنامه: برنامه از 3 قسمت تشکیل شده است

1. محاسبه فاکتوریل یک عدد مشخص

2. پیدا کردن بزرگ ترین فاکتوریل به طوری ک کوچکتر از عدد مشخص شده باشد

3. بسط کانتور عدد ک متشكل از دو عمل بالا به صورت تکراری است تازمانی ک عدد ما بزرگتر از 1 است (در هر مرحله عدد ما با عملگر - کوچکتر میشود)

در این سوال نیز برای زیبایی در خروجی از تابع `UserInput` استفاده شده است ک یکی از کارهای آن این است ک با استفاده از حلقه `while` خروجی تمیز و مرتبی را به ما نشان دهد.

مراجع: در این سوال از مرجعی استفاده نشده است.

مسئله 3: محاسبه کوچک‌ترین مضرب مشترک چند عدد

الگوریتم: در این سوال برای محاسبه ک.م.م دو عدد از رابطه ریاضی استفاده شده است به این صورت که ک.م.م دو عدد برابر است با حاصل ضرب آن دو عدد تقسیم بر ب.م.م آنها برای محاسبه ب.م.م نیز از الگوریتم اقلیدس به صورت بازگشته استفاده شده است برای محاسبه ب.م.م چند عدد از یک تابع بازگشته استفاده شده است که هر بار ب.م.م دو عدد را حساب می‌کند و در قدم بعد ب.م.م خروجی را با ب.م.م عدد بعدی محاسبه می‌کند. از همین منطق برای محاسبه ک.م.م نیز استفاده شده است

برنامه: برنامه از 4 تابع اصلی تشکیل شده از که عبارتند از:

1.محاسبه ب.م.م دو عدد

2.محاسبه ب.م.م چند عدد

3.محاسبه ک.م.م دو عدد

4.محاسبه ک.م.م چند عدد

در انتهای این تابع **UserInput** اعداد وارد شده را در یک لیست نمایش میدهد و خروجی آنها نیز در جلو ان لیست چاپ می‌کند.

مراجع:

Anany Levitin - **Introduction to the Design and Analysis of Algorithms**
فصل اول کتاب فوق برای محاسبه ب.م.م و ک.م.م دو یا چند عدد.

اولین برشور من با سوالی شبیه چنین سوالی بر میگردد به سوال شماره ۵ Euler ک شاید برای شما نیز جالب باشد. لینک آن را در همینجا قرار میدهم.
<https://projecteuler.net/problem=5>

مسئله 4: مرتب‌سازی تیم‌ها

الگوریتم: مهم ترین الگوریتم استفاده شده در حل این سوال پیاده سازی یک الگوریتم برای جایگشت های مختلف است. و الگوریتم به این صورت عمل میکند که فهرستی از عناصر را به عنوان ورودی می‌گیرد و با انتخاب یک عنصر در یک زمان و ایجاد مجدد جایگشت‌ها برای عناصر باقی‌مانده، به صورت بازگشته جایگشت ایجاد می‌کند.

برنامه: تابع `correct_orders` بررسی میکند که ایا ترتیب ورودی تیم‌ها بر اساس نتایج ترتیب درستی هست یا خیر در باره نحوه عملکرد تابع `permutations` در قسمت الگوریتم توضیح داده شد. تابع `find_correct_orders_sequence` ترتیب مدد نظر را بین جایگشت‌های صحیح به ما بر میگردد و در آخر تابع `UserInput` بسته به تعداد تیم‌ها تعداد خروجی ممکن را دریافت میکند و در خروجی نمایش میدهد.

قسمت‌هایی که توضیح مفصل داده نشد در متن کد گویا هستند

مراجع:

از انجایی که مجاز به استفاده از کتابخانه‌های پایتون نبودیم تولید همه جایگشت‌های ممکن چند شی یکی از چالش‌های اصلی من بود چرا که در مسایل دیگر مانند سوال 7 نیز کاربرد داشت. من مقاله مختصر در لینک زیر را برای درک مفهوم جایگشت و پیاده سازی آن مطالعه کردم.

<https://www.baeldung.com/cs/array-generate-all-permutations>

مسئله ۵: انتگرال گیری عددی

الگوریتم:

از آنجایی که مجاز به استفاده از عملگر $*$ نیستیم یک تابع مینویسیم که بتواند عملگر $**$ پیاده سازی کند. تابع `custom_pow` به طور خطی عدد a را به توان b میرساند. به تابع فاکتوریل هم نیاز داریم که به طور بازگشتی به محاسبه فاکتوریل میپردازد (در قبل به تفصیل توضیح دادم)

در اینجا سه تابع را باید تعریف کنیم با استفاده از محدودیت های موجود:

۱. برای پیاده سازی تابع $\sin(x)$ از بسط تیلور استفاده میکنیم راجب این الگوریتم میدانیم که:

سری تیلور یه نمایش ریاضی برای توابع ریاضی از جمله سینوس است که ان را به صورت جمع تعداد نامحدودی چند جمله ای نمایش میدهد. سری تیلور تابع سینوس به صورت زیر است.

$$\sin(x) = x - (x^3 / 3!) + (x^5 / 5!) - (x^7 / 7!) + \dots$$

از یه تولرانس استفاده میکنیم به این معنا که سری تیلور تابع سینوس تا کجا باید ادامه پیدا کند. از دوتابعی که در بالا تعریف کردیم در این الگوریتم چیاده سازی استفاده میشود.

۲. برای پیاده سازی تابع چند جمله ای و محاسبه مقدار تابع در یک نقطه ای معلوم تابع **polynomial** را تعریف میکنیم که در ورودی ضرایب چند جمله ای و مقدار x را برای محاسبه دریافت میکند. پیاده سازی بسیار ساده است و در متن کد با کامنت توضیح داده شده است.

۳. پیاده سازی تابع `sqrt` که برای پیاده سازی ان با دقیقی مناسب از الگوریتم هرون استفاده کردیم. این الگوریتم بر اساس تکرار محاسبات و بهره گیری از مقدار تخمینی قبلی جذر، به تقریبی بهتر از تخمین قبلی می رسد. ابتدا یک تخمین اولیه برای جذر مورد نظر قرار داده می شود و سپس با استفاده از فرمول تقریبی جدید، تخمین جدیدی بدست می آید. این مراحل تا زمانی ادامه می یابد که تخمین جدید به تقریب دقیق جذر نزدیک شود. الگوریتم هرون از جواب های قبلی خود استفاده می کند و به طور مستقیم و ساده جواب را به دست می آورد.
$$\text{تخمین جدید} = (\text{تخمین قبلی} + (\text{عدد ورودی} / \text{تخمین قبلی})) / 2$$
 و این کار تا زمانی ادامه

پیدا میکند ک به دقت مد نظر برسیم

برنامه: برنامه در چند بخش مجزا کار میکند. روش محاسبه انتگرال به روش ذوزنقه ای ک در متن سوال توضیح داده شده و در پیاده سازی آن هم به کمک زبان پایتون در کد به اندازه کافی توضیح داده شده است.

در بخش های بعدی به پیاده سازی توابع مورد نیاز در هر بخش میپردازیم
برای تابع `sin` تابع مقداری ک میخواهیم `sin` در ان نقطه محاسبه شود را از کاربر میگیرد.
برای تابع `sqrt` تابع به عنوان ورودی مقداری ک میخواهیم جذرش را حساب کنیم میگیرد
برای تابع `polynomial` تابع 2 ورودی میگیرد
1. مقدار تابع چندجمله ای در `x` مد نظر
2. ضرایب تابع چند جمله ای ک تعداد انها نمایانگر درجه تابع است.
برای مثال:

`coefficients = [1, -3, 2] → x^2 - 3x + 2`

`when x is 4 → (4)^2 - 3*(4) + 2 = 6`

قسمت اخر نیز تابع `UserInput` است ک از کاربر سوال هایی میپرسد و برنامه طبق نیازه کاربر پیش میرود و توابع مورد نیاز فراخوانی میشوند.

مراجع:

الگوریتم هرون :

<https://medium.com/@gauravswarankar/heron-algorithm-or-babylonian-method-square-root-14fb599db5d7>

بسط تیلور برای محاسبه سینوس:

https://en.wikipedia.org/wiki/Taylor_series

مسئله 6: تجزیه شهر

الگوریتم: برای حل این برنامه از چند الگوریتم متفاوت استفاده شد که به شرح انها میپذارم
یک الگوریتم برای محاسبه معادله عمود منصف بین دو نقطه
یک الگوریتم برای پیدا کردن نقاط برخورد عمود منصف با مستطیل و محاسبه دردسترس
ترین نقاط برخورد

برنامه: تابع `find_perpendicular` شبیه و عرض را از مبدا یک خط عمود بر پاره خطی که با
دو نقطه تعریف شده است محاسبه می کند.

در ادامه تابع `calculate_rectangle_impact_points` برای محاسبه خط عمود بر یک پاره
خط که با دو نقطه تعریف شده است و نقاط برخورد احتمالی بین آن خط عمود بر یک
مستطیل نوشته شده است.

لازم به ذکر است که درون این تابع یک تابع دیگری به نام `is_within_bounds` قرار دارد که
وظیفه آن این است که بررسی کند نقطه مشخص شده درون مستطیل قرار دارد یا خیر.
در انتهای این تابع نقاط احتمالی برخورد با مستطیل به ما برگردانده میشود.

در ادامه باید نقطه همرسی عمود منصف ها را پیدا کنیم و طبق قضیه ای در ریاضیات
دیبرستان میدانیم که عمود منصف های یک مثلث در یک نقطه همرس هستند و برای پیدا
کردن آن نقطه کافی است که نقطه برخورد دو عمود منصف را پیدا کنیم زیرا میدانیم عمود
منصف سوم نیز از همان نقطه میگذرد وظیفه این کار بر تابع `calculate_impact_point` است.

در انتهای تابع `UserInput` داده هارو از کاربر میگیرد و توابع مورد نیاز رو فراخوانی میکند.
مراجع: این سوال بسیار سوال چالش بر انگلیزی بود. و حتی مطمئن نیستم که به طور کامل
برای همه حالات به درستی کار کند. برای فهم ریاضیات این سوال از دوستان ترم بالایی کمک
گرفتم .

مسئله 7: ساخت رشته‌ها

الگوریتم: اصلی ترین قسمت الگوریتمی حل این سوال پیدا کردن جایگشت های مد نظر است که در سوال 4 به توضیح آن پرداختم. در این قسمت بیشتر به اینکه برنامه به چه شکل کار میکند میپردازم

برنامه:

تابع `allStringCombinations` تمام حالت های ممکن ساخت رشته را ایجاد میکند که همان قسمت جایگشت است که در بالا توضیح دادم. این کار به صورت بازگشتی انجام میشود.

در تابع `stringMaking` نیز ما تعداد کاراکتر ها و خود کاراکتر هارو از کار بر میگیریم و با استفاده از تابع بالا تمام جایگشت های ممکن را تولید میکنیم
مراجع: در حل این سوال از مرجعی جدیدی استفاده نشد. کد این سوال به اندازه کافی گویاست و از کامنت استفاده شده.

نمیدونیم چیو در چ حد باید توضیح بدم برای این سوال ☺

مسئله 8: ارزیابی عبارت‌های پسوندی

الگوریتم: در حل این سوال از پیاده سازی پشته استفاده شده است. به این صورت که ابتدا یک تازمانی که به عملوند برخورد می‌کنیم آنرا به پشته اضافه می‌کنیم زمانی که به عملگر رسیدیم به ترتیب عملوند‌ها را از پشته خارج می‌کنیم و با عملگر مد نظر عملیات ریاضی رو انجام میدهیم و دوباره حاصل را به پشته اضافه می‌کنیم و این کار را تا زمانی که به طور کامل رشته ورودی را پیمایش کنیم

برنامه: کلیت برنامه متشکل از یک حلقه فور برای پیمایش رشته مد نظر است و 6 شرط برای 6 عملگر مدنظر در انتهای اولین عضو پشته که تنها عوض آن نیز هست به عنوان خروجی نشان داده می‌شود.

مراجع:

از مرجعی استفاده نشد. فقط منطق کارکرد پشته‌ها از کتاب
Anany Levitin - Introduction to the Design and Analysis of Algorithms
قبلًا مطالعه شده بود

مسئله ۹: دوره زمانی طلایی

الگوریتم: الگوریتم استفاده شده در حل این سوال الگوریتم مرتب سازی شمارشی است به ای صورت که تعداد دانشمندان در هر بازه زمانی را می‌شمارد و بازه زمانی با حداکثر تعداد دانشمندان پیدا می‌کند.

برنامه:

در این برنامه یک حلقه فور فهرست دانشمندان را پیمایش می‌کند و برای هر دانشمند با یک حلقه فور دیگر در بازه زمانی از سال تولد تا سال مرگ پیمایش صورت می‌گیرد. سپس از یک دیکشنری برای ذخیره تعداد دانشمندان در هر سال استفاده می‌کند. اگر سال در دیکشنری موجود نباشد با ۱ مقدار دهی اولیه می‌شود و اگر وجود داشته باشد ۱ واحد افزایش می‌ابد.

چون استفاده از \max مجاز نبود در برنامه پیاده سازی شد. ک قابل فهم است

مراجع: در پیاده سازی این برنامه از مرجعی استفاده نشد.

مسئله 10: حل دستگاه‌های معادلات خطی

الگوریتم: روش کرامر در متن سوال توضیح داده شده و پیاده سازی آن نیز به زبان پایتون در کد قابل فهم است.

برای محاسبه ترمینان از الگوریتم بسط لاپلاس به صورت بازگشتی استفاده شده.
اگر ماتریس 2×2 بود با استفاده از فرمول $(ad - bc)$ دترمینان محاسبه می‌شود. در غیر این صورت از بسط لاپلاس به کمک حذف سطر و ستون‌ها استفاده می‌شود.
به طور کلی، الگوریتم به صورت بازگشتی با کاهش اندازه ماتریس تا زمانی که به حالت پایه یک ماتریس 2×2 برسد، محاسبات را انجام میدهد.

برنامه:

نیازی به توضیحات این قسمت نیست $\textcircled{7}$ کد قابل فهم است و به اندازه کافی از کامنت استفاده شده است

مراجع: بسط لاپلاس برای محاسبه دترمینان یک ماتریس مربعی
https://en.wikipedia.org/wiki/Laplace_expansion

پایان