هو العلیم

مبانی کامپیوتر و برنامه‌سازی

نیمسال اول سال تحصیلی 1403 – 1402

مسائل برنامه‌نویسی

پدرام زمانی نژاد

مسأله 1: تجزیه اعداد صحیح به عامل‌های اول آنها

الگوریتم:

ایده کلی حل سوال اینگونه بوده که باید اعداد رو به ترتیب از 2 شروع کنیم و نسبت به عدد ورودی مورد نظر تقسیم کنیم ببینیم که اون عدد بخش پذیر هست یا نه . نکته ی سوال به این صورت بوده که فقط کافی بوده که اگر عددی بخش پذیر بود اینقدر اون عدد رو تقسیم کنیم تا اینکه دیگه اون عدد که داریم تقسیم رو باهاش انجام میدهیم به خارج قسمت عدد ورودی بخش پذیر نباشه . و اینقدر این کار رو انجام میدهیم که عدد ورودی مورد نظر به یک برسه که خب واضحاً خود یک هم عامل بوده و من ننوشتمش که عدد یک رو هم نشان بده کما اینکه یک اصلا عدد اول نیست و با تعریف عدد اولی که من میدونم همخونی ندارد.

برنامه:

برنامه سعی شده به صورت خوانا نوشته بشه و توضیحاتی داخل فایل موجود هست .

در برنامه من برای مقداری بالا بردن بازده اومدم و اول از همه هر عددی که میگیریم رو به 2 تقسیم میکنم که درجه عدد 2 بدست بیاد . دلیل اینکار این بوده که بعدا وقتی میخوام روی تمامی اعداد حلقه بزنم میتونم اعداد رو یکی در میان چک نکنم و خب **سرعت** برنامه مقداری میره بالا .

کار دیگه ای که انجام شده برا بازدهی بالاتر این بوده که به جای اینکه از 2 تا خوده عدد مورد نظر حلقه زده بشه تا نصف اون عدد ادامه میدیم چون که در بدترین حالت ممکن انگار که اون عدد تشکیل شده از یک عدد اول ضربش با عدد 2 که خب همین امر برا موجب شد که لازم نشه کامل برم و فقط تا نصف عدد کافیه چون مکمل دو هستن و کمتر از دو عامل اولی نیست .

مراجع:

مرجع خاصی برا حل این سوال استفاده نشده .

مسأله 2: بسط فاکتوریلی اعداد طبیعی

الگوریتم:

ایده ی سوال اینجوریه که بیایم و بزرگترین فایکتوریل، کوچکتر از عدد مورد نظر رو پیدا کنم و بعدش عدد مورد نظر رو به فاکتوریل تقسیم کنیم . در این تقسیم خارج قسمت ضریب فاکتوریل میباشد . دوباره اینقدر این کار انجام میشه تا عدد به صفر برسه که حتماً میرسه چون با یک فاکتوریل عدد کامل میشه و جمع کاملی خواهد داشت .

برنامه:

برای حساب کردن فاکتوریل یک تابع در فایل utils که داخل پوشه utils هست نوشته و استفاده شده . خروجی برنامه به صورت یک رشته میباشد که شامل مجموع های فاکتوریل ها میباشد .

مراجع:

در این سوال هم از منبع خاصی استفاده نشده است .

مسأله 3: محاسبه کوچک‌ترین مضرب مشترک چند عدد

الگوریتم:

برای پیدا کردن کوچکترین مضرب مشترک در درس نظریه اعداد سال دوازدهم خوانده ایم که :

$$lcm(a,b) = \frac{a \times b}{\gcd(a,b)}$$

که در اینجا lcm همان کوچکترین مضرب مشترک و gcd همان بزرگترین مقسوم علیه مشترک میباشد . تنها چالش حساب کردن ان برای چندین ورودی بود .

برنامه:

برای اینکه بتوانم چندین ورودی را به یک تابع بدهم از *args استفاده شده که همه ی ورودی های تابع را گرفته و در یک تاپل قرار میدهد .

حالا برای حساب کردم lcm این اعداد کافی بود برای 2 عدد lcm را پیدا کـرده و جـواب آن را بـه اعداد قبلی خود اضافه کرده تا به یک عدد برسیم و آن عدد lcm تمام ورودی های خواهد بود .

مراجع:

تابع gcd در کلاس قبلا زده شده و از همان استفاده شده است که الگوریتم اقلیدوس میباشد .

مسأله 4: مرتب‌سازی تیم‌ها

الگوریتم:

با وجود وقتی که برای حل این سوال بوده و حتی کمک گرفتن برای ایده ی حل بنده ناتوان بودم که این سوال را حل کنم و برنامه ای برایش بنویسم .

برنامه:

مراجع:

مسأله 5: انتگرال‌گیری عددی

الگوریتم:

چالش های اصلی این سوال محاسبه سینوس هر زاویه رادیانی و پیدا کردن هر عدد رادیکالی بوده است که سینوس هر زاویه رادیکالی با سری تیلور نوشته شده و برا محاسبه رادیکال روش نیوتون به کار برده شده است . و برای محاسبه انتگرال هم از این رابطه استفاده شده است :

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(x_0) + 2\sum_{k=1}^{n-1} f(x_k) + f(x_n)\right]$$

که در آن : $h = \frac{b-a}{n}$ میباشد .

برنامه:

در برنامه یک تابع با چند ورودی اغاز بازه پایان بازه تعداد بازه هایی که میخواهیم (n) ، و با مشخص کردن نوع تابع که یکی از سه نوع (sin, sqrt, poly) است هر کدام را جداگانه محاسبه میکند.
Poly برای توابع چند جمله ای لحاظ شده است .
در بخش محاسبه سینوس اعداد **شروع و پایان بازه ها در عدد پی ضرب میشوند** زیرا سری تیلور باید ایکس ها به رادیان باشد .

مراجع:

برای سری تیلور : https://en.wikibooks.org/wiki/Trigonometry/Power_Series_for_Cosine_and_Sine

برای رادیکال از روش نیوتون : https://math.mit.edu/~stevenj/18.335/newton-sqrt.pdf

مسأله 6: تجزیه شهر

الگوریتم:

برای حل این سوال از دیاگرام ورونی (voronoi diagram) استفاده شده .
اساس کار این الگوریتم عمود منصف های بین هر دو نقطه میباشد چرا که نقاط روی هر عمود منصف، فاصله ی آن با دو نقطه برابر میباشد پس نواحی بین این دو نقطه به دو دسته تقسیم میشود که یکی، به یکی از نقاط و دیگری ، به نقطه ی دیگر نزدیک میشود .

برنامه:

برنامه در چند تابع نوشته شده است . تابع (find_prependicular) برای پیدا کردن شیب و عرض از مبدا ، عمود منصف هر نقطه میباشد . تابع (impact_prependicular_rectangle) برای نقاط برخورد هر عمود منصف با اضلاع مستطیل است . تابع(impact_prependicular) برای پیدا کردن نقاط برخورد عمود منصف ها با یکدیگر است و تابع (main_idea) برا کنار هم گذاشتن اجزای پازل میباشد .
این تابع برا سه نقطه تست شده است و برای بیشتر تست نشده است .
ضعف : این تابع نقاط یک چند ضلعی را بر نمیگرداند و فقط **نقاط مهمی که مورد نیاز کشیدن این دیاگرام هست را بر میگرداند .**

مراجع:

اسم این الگوریتم را یکی از افراد کلاس به اسم **سید محمد دانیال قربی** به من معرفی کرد .
جستوجوی بیشتر در : https://en.wikipedia.org/wiki/Voronoi_diagram
انجام شده است .
و برای درک بیشتر : https://www.youtube.com/watch?v=j2c3kumwoAk&t=79s
این ویدیو را تماشا کرده ام .

مسأله 7: ساخت رشته‌ها

الگوریتم:

این سوال اساسا جاگشت های n حروف را میخواست .

بنابراین این گونه برنامه نوشته شده است که ابتدا جاگشت دو حرفی را تولید میکنیم با اضافه کردن هر حرف و حرف بعدی ، سپس برای ادامه حرف های دوحرفی که تولید شده است را به حروف یک حرفی که از آن استفاده نشده اضافه میکنیم و حروف سه حرفی را میسازیم و دوباره این فرایند تکرار میشود

برنامه:

درقسمت اول برنامه جاگشت های با دو حرف را اول تولید میکنیم و برعکس آن ترکیب را به تولید میکنیم و به لیستی تحت عنوان result میریزیم بعد از آن نیاز داریم حلقه ای بزنیم که بتوانیم از شروع حروف دو حرفی شروع به چرخش کنیم و مواردی که در ان ترکیب ها نیست را به آنان اضافه کنیم . از آن جایی که قبلا برعکس حروف دو حرفی را اضافه کرده ایم پس با این کار برعکس تمام جایگشت های دو تایی به بالا را خود برنامه تولید میکند .

با استفاده از متغیر perm_count در حر بار چرخش حلقه اگر بخواهیم جایگشت های چهار تایی را تولید کنیم پس به تعداد جایگشت های دو تایی و یک تایی نیاز داریم که با استفاده از رابطه تعداد جایگشت ها با استفاده از متغیر counter به perm_count اضافه میکنیم و حلقه را ادامه میدهیم.

دلیل این کار این است که جایگشت های چهار تایی ؛ جایگشت های سه تایی بودند که عنصر استفاده نشده از آنها به انتهایشان اضافه میشود .

مراجع:

برای ایده گرفتن **امین منصوری** تیکه از یک کتاب به اسم : Introduction to the Design and Analysis of Algorithms

مسأله 8: ارزیابی عبارت‌های پسوندی

الگوریتم:

در این سوال اینجوری حل شده که اگر از چپ به راست بریم هر بعد از هر دو عدد یک عملگر هست. پس اگر ورودی را تبدیل به یه لیست همه ی اینها بکنیم و برعکس بکنیم که از چپ شروع به حرکت بکنیم باید به ازای سومین عنصر هر بار جلو رفتن یک عملگر داریم و میتونیم اون عملیات را تکرار کنیم ، سپس عملگر و اعداد را محاسبه میکنیم و از آن لیست خارج میکنیم و نتیجه را به اول آن لیست جایگذاری میکنیم.

چالش این نوع حل این بوده که وقتی اضافه کنیم جواب را عضو چهارم این لیست عملگر هست . و باید عضو دوم و سوم را با عملگر چهارم محاسبه کنیم و سپس جایگذاری کنیم.

برنامه:

برنامه ی سوال مورد خاصی برا توضیح نداشت ، زیرا سعی شده در قسمت الگوریتم آن را کامل توضیح بدم .

فقط دو متغیر index و first_index برای چک کردن بوده . متغیر index نشانه گذار عملگر بوده است و first index نشانه گر عضو هایی بوده که قرار بوده محاسبه شوند .

برای تشخیص عدد یا عملگر بودن آن عضو از قابلیت int() استفاده شده چرا که اگر عدد نباشد پس به value ارور میخوریم که خب به معنی پیدا شدن عملگر میباشد و محاسبات انجام میشود .

مراجع:

منبع خاصی برای حل این سوال استفاده نشده است .

مسأله 9: دوره زمانی طلایی

الگوریتم:

برای حل این سوال ، اول تمامی افراد را داخل یک لیست که در آن یک لیست دیگه است که عضو اول اسم آن شخص ، عضو دوم سال تولد و عضو سوم سال مرگ میباشد. بعد از ان به عزای هر تاریخ تولد چک میشود که چند دانشمند در آن سال قید حیات داشته و نمرده است ؛ در این برنامه شخصی که تازه به دنیا آمده است در آن سال شمارش نمیشود .

بعد از آن مرگ هر دانشمند را مقایسه میشود به این صورت که اگر قبلا آن سال مقایسه شده بوده است دوباره حساب نشده و فقط یک دانشمند به آن سال اضافه میشود در غیر اینصورت دوباره این دانشمند هم مقایسه میشود .

بعد از محاسبه شدن تعداد دانشمندان در هر سال ، سال هایی که بیشترین دانشمندان در آن حضور دارند را حساب کرده و بعد از آن چک میشود که اگر این سوال ها توانایی تشکیل دوره دارند یا خیر برای مثال اگر در سال 1995 و سال 2015 هر دو 5 دانشمند زنده بوده است اگر در بین این دو سال، سالی وجود نداشته باشد که کمتر یا بیشتر از 5 دانشمند در آن زنندگی کنند؛ بازه زمانی 1995 تا 2015 دوران طلایی حساب میشود.

برنامه:

در کد تابع find_golden_time وظیفه ی پیدا کردن تعداد دانشمندان در هر سال را دارد که توضیحات لازمه در فایل داده شده است.

در تابع golden_time وظیفه پیدا کردن بیشترین تعداد دانشمند در یک سال را دارد و پیدا کردن دوره ی زمانی اگر وجود داشته باشد.

مراجع:

برای حل سوال و ایده ی سوال منبعی وجود نداشته ولی برای کمک گرفتن در یک قسمت از کد از chat gpt کمک گرفته شده است که در کد ، آن قسمت مشخص شده است .

مسأله 10: حل دستگاه‌های معادلات خطی

الگوریتم:

این سوال الگوریتم خاصی نیاز نداشته است، چرا که الگوریتم لازمه در متن سوال داده شده بوده است. و فقط بار کد نویسی داشته است.

برنامه:

داخل برنامه دوتابع به نام های det برای گرفتن دترمینان ماتریس مورد نظر و تابع calc_x برای محاسبه معادلات خطی ساخته شده است.

تابع det یک تابع بازگشتی با شرط خروج این است که اگر دترمینان دو در دو داشتیم با رابطه ای که خوانده ایم محاسبه میشود که باعث مقداری خیلی کمی سریع شدن برنامه میشود. رابطه :

$$\begin{vmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{vmatrix} = (\boldsymbol{a} \times \boldsymbol{d}) - (\boldsymbol{b} \times \boldsymbol{c})$$

لازم به ذکر است که برای محاسبه این دترمینان به دلیل اینکه قاعده کرامر در اصل تمامی ماتریس های n در n را تبدیل به ماتریس یک در یک میکند ما نیاز به یک ماتریس که کپی ای از ماتریس اصلی باشد که در کد به اسم temp_data نشان داده شده است در این تابع بود. برای اینکار باید اصطلاحاً یک دیپ کپی از ماتریس اصلی انجام می شد چرا که در زبان برنامه نویسی پایتون اگر از لیستی کپی بگیریم در اصل یک pointer به آن شی گذاشته ایم . که این کار با کپی گرفتن از تک تک اعضای ماتریس اصلی با استفاده از list comprehension انجام شده است.

در تابع calc_x اول از ماتریس ضرایب دترمینان گرفته و آن را ذخیره میکنیم، سپس با چرخش به تعداد اعضای جواب معادلات هر بار یک کپی از ماتریس اصلی گرفته و سپس با استفاده از قاعده کرامر ستون jام ماتریس ضرایب را با ماتریس ستونی جواب ها عوض کرده و دترمینان میگیریم سپس جواب دترمینان ماتریس دوم را تقسیم به دترمینان ماتریس ضرایب کرده و برابر با x عه jام میگذاریم.

مراجع:

برا این سوال هم مرجع خاصی استفاده نشد چرا که متن سوال کامل بوده است.

فقط برای درک بیشتر قاعده کرامر از این ویدیو استفاده شد : https://www.youtube.com/watch?v=Wb34tiSGM8A