هوالعلیم

مبانی کامپیوتر و برنامه‌سازی

نیمسال اول سال تحصیلی ۱۴۰۳ – ۱۴۰۲

مسائل برنامه‌نویسی

مسأله ۱: تجزیه اعداد صحیح به عامل‌های اول آنها

هر عدد صحیح بزرگ‌تر از ۱ را می‌توان دقیقاً به یک شیوه به شکل حاصل‌ضرب عامل‌های اول آن تجزیه کرد. مسأله، تعیین عامل‌های اول اعداد صحیح و تعداد دفعات تکرار هر یک از آن عامل‌ها در شکل تجزیه‌ای عدد است.

مثال:

$$20 = 2^2 \times 5$$

$$51191188125 = 3 \times 5^4 \times 7^2 \times 11 \times 37^3$$

ورودی برنامه: یک عدد صحیح است.

خروجی برنامه: عامل‌های اول عدد صحیح و تعداد دفعات تکرار هر یک از آنها است.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

مسأله ۲: بسط فاکتوریلی اعداد طبیعی

هر عدد صحیح نامنفی m را می‌توان به این شکل (که به آن **بسط کانتوری**[1] عدد هم گفته می‌شود) نوشت:

$$m = a_n n! + a_{n-1}(n-1)! + \cdots + a_2 2! + a_1 1!$$

در این رابطه، n عددی بزرگ‌تر از صفر است و هر a_i ، یک عدد صحیح نامنفی است با قید $0 \leq a_i \leq i$.

مثال:

$$2 = 2!$$

$$11 = 3! + 2(2!) + 1!$$

$$115 = 4(4!) + 3(3!) + 1!$$

$$1000 = 6! + 2(5!) + 4! + 2(3!) + 2(2!)$$

ورودی برنامه: یک عدد صحیح نامنفی است.

خروجی برنامه: بسط فاکتوریلی آن عدد است.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

[1] Cantor expansion

مسأله ۳: محاسبه کوچک‌ترین مضرب مشترک چند عدد

کوچک‌ترین مضرب مشترک چند عدد صحیح، کوچک‌ترین عددی است که بر هر یک از آن اعداد بخش‌پذیر باشد.

مثال:

$$\mathrm{lcm}(6,9) = 18$$

$$\mathrm{lcm}(2,5,7) = 70$$

$$\mathrm{lcm}(12, 39, 100) = 3900$$

$$\mathrm{lcm}(234, 734, 902, 513, 652) = 719699033196$$

ورودی برنامه: تعدادی عدد طبیعی است.

خروجی برنامه: کوچک‌ترین مضرب مشترك اعداد است.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

مسأله ۴: مرتب‌سازی تیم‌ها

فرض کنید نتایج نهایی یک دوره مسابقات ورزشیِ گردشی که در آن، هر یک از n تیم، دقیقاً یک بار با هر یک از تیم‌های دیگر، بازی کرده، در اختیار شما گذاشته شده است. هر بازی، یا با پیروزی یکی از دو تیم به پایان رسیده است یا با تساوی دو تیم. هدف، تشکیل دنباله‌ای از نام (یا شماره) تیم‌ها است به گونه‌ای که هر تیم، در بازی با تیمی که بلافاصله بعد از آن در دنباله قرار گرفته است، شکست نخورده باشد.

مثال:

فرض کنید سه تیم با شماره‌های ۱ و ۲ و ۳ با یکدیگر بازی کرده باشند و این نتایج به دست آمده باشد:

- تیم ۱ ، با تیم ۲ مساوی کرده باشد؛
- تیم ۲ ، تیم ۳ را شکست داده باشد؛
- تیم ۳ ، تیم ۱ را شکست داده باشد؛

آنگاه می‌توان سه تیم را در این ترتیب گذاشت:

$$1\,,2\,,3$$

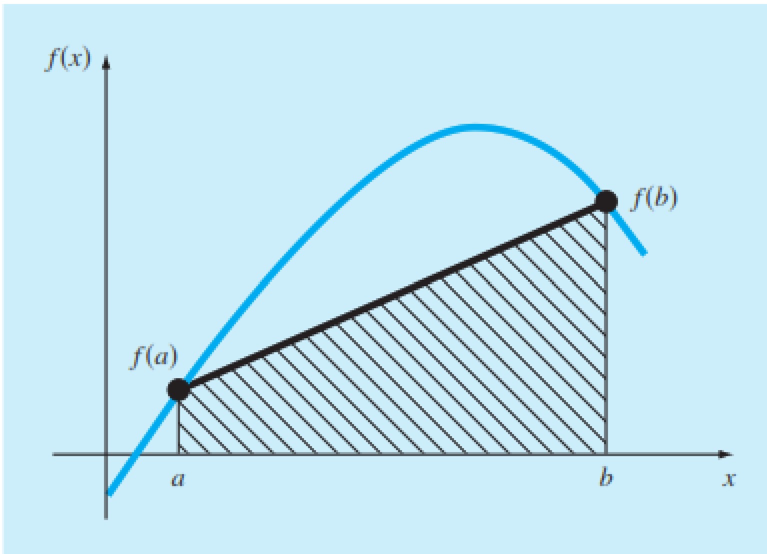
ورودی برنامه: تعداد تیم‌ها و نتیجه هر یک از مسابقات است. (هر تیم را می‌توان با نام آن یا با شماره آن مشخص کرد.)

خروجی برنامه: دنباله‌ای از نام (یا شماره) تیم‌ها است به گونه‌ای که هر تیم، در بازی با تیمی که بلافاصله بعد از آن در دنباله قرار گرفته است، شکست نخورده باشد.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

مسأله ۵: انتگرال‌گیری عددی

روش‌های متعددی برای محاسبه تقریبی انتگرال معین توابع شناخته شده است. یکی از این روش‌ها، روشی است که به نام **قاعده ذوزنقـه‌ای**[2] شناخته می‌شود. فرض کنید $f(x)$ تابعی تک متغیره و a و b دو عدد حقیقی باشد. علـت نامگـذاری این روش به قاعده ذوزنقه‌ای، این است که مقدار انتگرال تابع $f(x)$ در هر بازه را با مساحت ذوزنقـه‌ای واقـع در زیـر نمودار تابع $f(x)$ در بازه $[a,b]$ تقریب می‌زنیم.



در حالت کلی، برای استفاده از روش ذورنقه‌ای، بازه $[a,b]$ را به n بازه هر یک با طول $h=\frac{b-a}{n}$ تقسـیم مـی‌کنیم. دو نقطه انتهایی هر بازه برحسب a و b مشخص می‌شود:

$$[x_0,x_1],[x_1,x_2],\cdots,[x_{n-1},x_n]$$

در اینجا $x_0=a$ و $x_n=b$ و $x_i=a+ih$ است.

مقدار انتگرال تابع در هر بازه را با مساحت ذوزنقه‌ای واقع در زیر نمودار تابع در آن بازه تقریب می‌زنیم:

$$I=\int_a^b f(x)dx=\int_{x_0}^{x_1} f(x)dx+\int_{x_1}^{x_2} f(x)dx+\cdots+\int_{x_{n-1}}^{x_n} f(x)dx$$

$$I=\int_a^b f(x)dx=h\frac{f(x_0)+f(x_1)}{2}+h\frac{f(x_1)+f(x_2)}{2}+\cdots+h\frac{f(x_{n-1})+f(x_n)}{2}$$

[2] trapezoidal rule

نهایتاً می‌توان فرمولی که مبنای قاعده ذوزنقه‌ای است را به این صورت نوشت:

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

ورودی برنامه: تابع $f(x)$ و اعداد حقیقی a و b و عدد طبیعی n است. توابع مورد نظر عبارتند از:

- $f(x) = \sin x$
- $f(x) = \sqrt{x}$
- $f(x) = \sum_{i=0}^{n} a_i\, x^i = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0$

خروجی برنامه: مقدار تقریبی انتگرال معین $\int_a^b f(x)dx$ است.

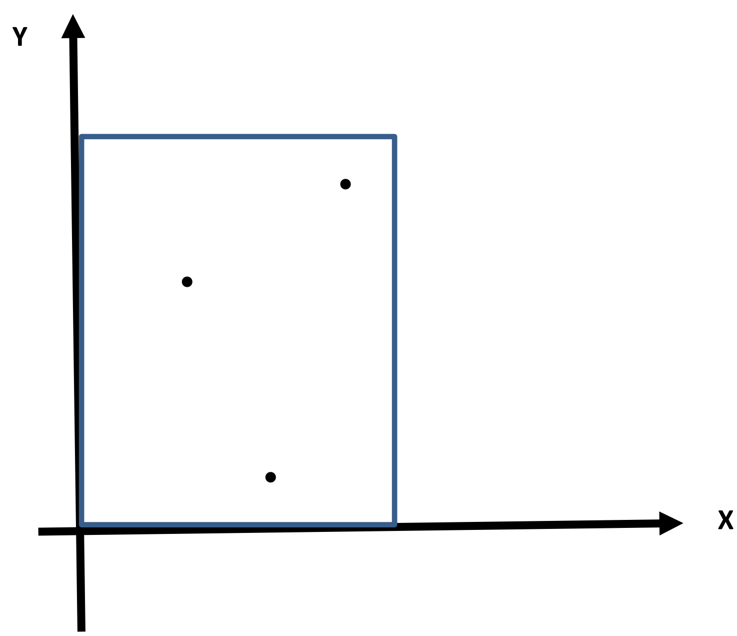
توابع مجاز: input()، int()، float()، list()، len()، range()، print()، توابع کلاس list، توابع کلاس tuple، توابع کلاس str، توابع کلاس set و توابع کلاس dict. (استفاده از عملگر ** مجاز نیست.)

مسأله ۶: تجزیه شهر

یکی از مسائلی که مدیران فروشگاه‌های زنجیره‌ای معروف با آن مواجه‌اند، تعیین مکان مناسب برای افتتاح شعبه‌های جدید است. یک معیار برای انتخاب مکان فروشگاه جدید این است که آن شعبه به شعبه‌ای (از شعبات موجود) که تعداد خریداران از آن بیشتر است، نزدیک‌تر باشد تا به شعبه‌های دیگر؛ بلکه به این شکل بتوان بار خریداران را متوازن‌تر در بین شعبه‌ها تقسیم کرد.

وضعیتی را تصور کنید که سه شعبه از یک فروشگاه در شهری بزرگ افتتاح شده‌اند. و مدیر فروشگاه می‌خواهد که شعبه چهارم را هم افتتاح کند. مسأله مدیر فروشگاه این است که نقشه شهر را به سه ناحیه تقسیم کند به قسمی که هر ناحیه شامل تمام نقاطی باشد که به یکی از سه فروشگاه نزدیک‌ترند تا به دو فروشگاه دیگر. بعد از آنکه ناحیه‌های مورد نظر مشخص شدند، او خود مکان مناسب برای ایجاد شعبه جدید را در یکی از آن سه ناحیه انتخاب خواهد کرد.

برای حل این مسأله، دانستن محیط شهر و مکان سه فروشگاه لازم است. فرض کنید محیط شهر مستطیلی به طول x کیلومتر و عرض y کیلومتر باشد. ما می‌توانیم این مستطیل را در دستگاه مختصات دکارتی به گونه‌ای نمایش دهیم که رأس پایینی و چپی مستطیل، منطبق بر مبدأ مختصات باشد. و مکان سه فروشگاه را نیز با سه نقطه در فضای دو بُعدی نمایش می‌دهیم. با این ترفند، اگر مختصات یک نقطه مثلاً $(7,\ 18)$ باشد، یعنی باید از پایین‌ترین و چپ‌ترین نقطه در مرز شهر ۷ کیلومتر به راست و سپس ۱۸ کیلومتر به بالا حرکت کنیم تا به آن نقطه برسیم. با این مفروضات، شما برای حل مسأله مدیر، باید راهی برای تجزیه یک مستطیل (نقشه شهر) به سه چندضلعی (ناحیه‌های مطلوب) بیابید.



ورودی برنامه: مقدار طول و عرض مستطیل (برحسب کیلومتر) و مختصات سه نقطه (مکان سه فروشگاه موجود) است.
(نقاط ورودی، باید در داخل یا روی مستطیل واقع باشند.)

خروجی برنامه: رئوس هر یک از سه چندضلعی (سه ناحیه) مورد نظر است.

توابع مجاز: input() ، int() ، float() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

مسأله ۷: ساخت رشته‌ها

فرض کنید $A = \{a_1, a_2, \cdots, a_n\}$ مجموعه‌ای از n نویسه (کاراکتر) باشد. این نویسه‌ها ممکن است حروف کوچک یا بزرگ الفبای زبان انگلیسی یا نمادهایی مثل + و % و] یا هر نماد دیگری باشند. مسأله، ساخت همه رشته‌هایی به طول ۱ تا n است که می‌توان با نمادهای الفبا ساخت. محدودیت این است که در هیچ رشته‌ای نباید نماد تکراری وجود داشته باشد.

مثلاً اگر $A = \{a, b, c\}$ باشد، آنگاه رشته‌های متفاوتی به طول ۱ و ۲ و ۳ که می‌توان با سه نویسه a و b و c ساخت عبارتند از:

$$a, b, c$$
$$ab, ac, ba, bc, ca, cb$$
$$abc, acb, bac, bca, cab, cba$$

ورودی برنامه: تعداد کاراکترها و خود کاراکترها است.

خروجی برنامه: همه رشته‌هایی به طول ۱ تا n است که می‌توان با n کاراکتر ساخت.

توابع مجاز: input()، int()، list()، len()، range()، print()، توابع کلاس list، توابع کلاس tuple، توابع کلاس str، توابع کلاس set و توابع کلاس dict.

مسأله ۸: ارزیابی عبارت‌های پسوندی

عبارت حسابی **میانوندی**[3]، به عبارتی حسابی گفته می‌شود که در آن، هر عملگر دودویی در میان دو عملوند (عدد) قرار گرفته باشد. برای مثال، این عبارت، یک عبارت حسابی میانوندی است:

$$(2+3)*(9-4)$$

عبارت حسابی **پیشوندی**[4]، به عبارتی حسابی گفته می‌شود که در آن، هر عملگر دودویی پیش از دو عملوند (عدد) قرار گرفته باشد. برای مثال، این عبارت، یک عبارت حسابی پیشوندی است:

$$* + 2\ 3\ - 9\ 4$$

برای ارزیابی یک عبارت حسابی پیشوندی، از راست به چپ آن را پردازش می‌کنیم و به هر عملگری که رسیدیم، آن را روی دو عملوندی که بلافاصله در سمت راست آن قرار گرفته‌اند، اعمال می‌کنیم و حاصل عملیات را به عنوان یک عملوند، جایگزین دو عملوند و عملگر می‌کنیم. عبارت حسابی پیشوندی بالا را به صورت زیر ارزیابی می‌کنیم:

$$* + 2\ 3\ - 9\ 4$$

$$* + 2\ 3\ 5$$

$$* 5\ 5$$

$$25$$

ورودی برنامه: عملوندها و عملگرها در یک عبارت حسابی پیشوندی است که یک به یک از چپ به راست وارد می‌شوند. (می‌توان از هر یک از عملگرهای حسابی + و - و * و / و // و ** در عبارت حسابی پیشوندی استفاده کرد.)

خروجی برنامه: مقدار عبارت حساب پیشوندی است.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، ord() ، chr() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

---
[3] infix
[4] prefix

مسأله ۹: دوره زمانی طلایی

ویراستار کتاب «تاریخ علم جهان» می‌خواهد این را بداند که در چه دوره زمانی، بیشترین تعداد از دانشـمندان برجسـته زنده بوده‌اند. منظور از «دانشمندان برجسته» افرادی است که سال تولد و سال وفات آنها در کتـاب ذکـر شـده باشـد. (ذکری از دانشمندان زنده در کتاب به میان نیامده است.) در بخش نمایه[۵] کتاب، نام همه دانشمندان به ترتیـب الفبـایی ذکر شده است و بعد از نام هر دانشمند، سال تولد و سال وفات او نیز مشخص شده است.

مسأله این است که از روی اطلاعات مذکور در نمایه کتاب، دوره زمانی مورد نظر و نام دانشـمندان زنـده در آن دوره مشخص شود. در موردی که شخص A در همان سالی که شخص B متولد شده باشد، از دنیا رفته باشد، زمان وفات شخص A قبل از زمان تولد شخص B به حساب آورده می‌شود.

ورودی برنامه: نام و سال تولد و سال وفات هر یک از دانشمندان است.

خروجی برنامه: دوره زمانی طلایی (که در آن دوره بیشترین تعداد از دانشمندان برجسته زنده بودند) و نام دانشمندانی که در آن دوره زمانی می‌زیستند.

توابع مجاز: input() ، int() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .

---

[5] index

مسأله ۱۰: حل دستگاه‌های معادلات خطی

ساختار کلی دستگاه خطی n معادله در n مجهول را در نظر بگیرید:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

حل این دستگاه به معنای یافتن مقادیر مجهول $x_1, x_2, \dots, x_n$ است به گونه‌ای که با آن مقادیر، تمام n معادلـه درسـت شوند.

دستگاه خطی n معادله در n مجهول را می‌توان با نمادهای ماتریسیِ

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

به صورت فشرده‌تر هم نمایش داد:

$$Ax = b$$

به A ماتریس ضرائب و به b بردار طرف راست دستگاه گفته می‌شود.

روش‌های متعددی برای حل دستگاه‌های معادلات خطی وجود دارد. یکی از آن روش‌ها، **قاعـده کرامـر**[۶] اسـت. بـرای استفاده از این روش، محاسبه دترمینان ماتریس‌های $n \times n$ لازم است.

دترمینان ماتریس $n \times n$

$$A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0n-1} \\ a_{10} & \dots & \dots & a_{1n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-10} & \dots & \dots & a_{n-1n-1} \end{bmatrix}$$

که با $\det A$ نشان داده می‌شود، اگر $n = 1$ باشد، برابر است با a_{00} ، و اگر $n > 1$ باشد، با استفاده از فرمول بازگشتی

$$\det A = \sum_{j=0}^{n-1} s_j\, a_{0j} \det A_j$$

تعریف می‌شود. در اینجا، s_j برابر است با ۱، اگر j زوج باشد و برابر است با ۱-، اگر j فرد باشد؛ a_{0j} عنصر سـطر 0 و

[6] Cramer’s rule

ستون j ماتریس A است؛ و A_j هم ماتریس $(n-1)\times(n-1)$ ای است که از حذف سطر 0 و ستون j ماتریس A بـه دست آمده باشد.

اکنون با مشخص شدن تعریف و نحوه محاسبه دترمینان یک ماتریس، می‌توان قاعده کرامر را بیان کـرد. طبـق **قاعـده کرامر**، جواب دستگاه معادلات خطی با این فرمول‌ها به دست می‌آید:

$$x_1 = \frac{\det\ A_1}{\det\ A}\ , \dots\ , x_j = \frac{\det\ A_j}{\det\ A}\ , \dots , x_n = \frac{\det\ A_n}{\det\ A}$$

در اینجا، $\det\ A_j$ دترمینان ماتریسی است که از جایگزینی ستون j اُم ماتریس A با ستون b به دست می‌آید.

ورودی: ماتریس A (ماتریس ضرائب) و بردار b (بردار طرف راست) است.

خروجی: مقادیر متغیرهای مجهول $x_1, x_2, \dots , x_n$ است.

توابع مجاز: input() ، int() ، float() ، list() ، len() ، range() ، print() ، توابع کلاس list ، توابع کلاس tuple ، توابع کلاس str ، توابع کلاس set و توابع کلاس dict .