SigmaCamp's *Problem of the Month* Contest

SEPTEMBER 2025

IMPORTANT: For the 2025-2026 season, POM is experimenting with two format changes:

- **Video solutions** must be submitted for each problem.
- Each month will include a **project** (worth 30 points) in place of one or two subjects.
- We now have monthly **POM office hours** where you can ask questions about this month's POM problems! See sigmacamp.org/pom/office-hours for details.

Video Submissions

Starting this year, **all solutions must be accompanied by short videos explaining your work.**

- Videos must be at most **2 minutes long** for 5pt/10pt problems, and at most **6 minutes long** for projects.
- **Solutions must be narrated**, but you do not need to show your face. Acceptable formats include:
  - Screensharing slides or a drawing app (e.g., MS Paint) with narration.
  - Recording a whiteboard, paper, or easel with narration (contents may be pre-written).
  - Speaking directly to the camera.
- **Submit videos as links** (Google Drive, Youtube, Dropbox, etc.). Extra requested files may be submitted as a single PDF file per problem.
- For coding problems, submit your code along with a video explaining your submission. Only **Python-3** (`.py`) and **Java** (`.java`) code submissions are accepted.

Please see sigmacamp.org/pom for full details on the 2025 POM format change.

Physics

5 points:

This problem is about pulleys. If you don't know much about pulleys, **DON'T PANIC!** Links to videos and information can be found at the bottom of the problem. This problem is split into four parts that build on each other.

If you only get through some of the parts, you can get partial points for the ones you do submit. You can also get partial points if your reasoning is on the right path. Since September is the first month of POM, we will be very generous with partial credit when grading.

Additional information that may be helpful is provided at the end of the problem.

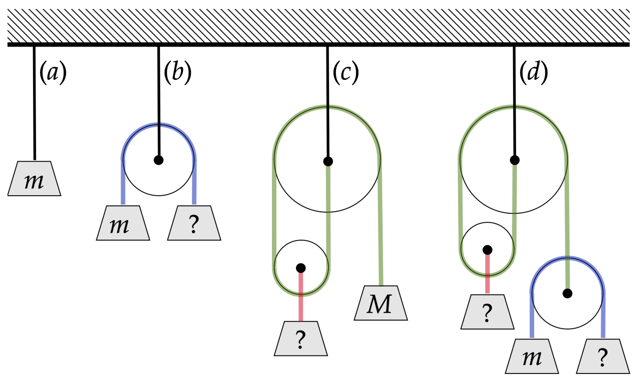


Figure 1: Diagrams for the 5 point Physics problem. All lines, black and coloured, represent strings. Circles represent pulleys, and grey trapezoids represent weights. If a weight is labelled with a letter, that letter represents the weight's mass.

All of the systems in this problem are assumed to be at rest (that is, all masses and pulleys are stationary). The ropes and pulleys are massless.

(a) (0 points) In Figure 1(a), what is the tension in the black string? (Since this is worth 0 points, the answer to this part is at the end of the problem, and you are not require to submit the solution. But think about it before you go look at the answer.)

(b) (1 point) In Figure 1(b), what is the tension in the blue rope? What is the mass of the weight labeled with a question mark? What is the tension in the black rope?

(c) (1 point) In Figure 1(c), what is the tension in the green rope? What is the tension in the red rope? What is the mass of the weight labeled with a question mark?

(d) (3 points) In Figure 1(d), what are the masses of the weights labeled with question marks? Which tensions do you need to compute to find them?

Additional information

- Tension is a force transmitted through a string due to it stretching/being pulled. While physical strings in the real world can be quite complicated (for example, they can stretch, fray, or behave like a spring), in an idealized problem we make a very simple approximation – they have no mass, and they cannot stretch. This simple approximation is often good enough for many real-world applications. This video is a really good introduction to tension and other forces.
- The force due to gravity on an object of mass m on the earth's surface is $F_g = mg$, where $g = 9.81\text{m/s}^2$. However, g is often approximated as 10m/s^2. You may use this approximation in your solutions.

- Newton's second law states that the acceleration $\vec{a}$ of an object of mass m is related to the net forces $\vec{F}_{net}$ acting on the object as follows: $\vec{F}_{net} = m\vec{a}$.
- When working on problems with multiple masses, it is often convenient to draw what is called a "free body diagram" for each mass (here's a good video). A free body diagram for an object is a diagram indicating all the forces acting on the object. Such a diagram can be used to find an equation for the net force on the object.
- You might find this block-and-tackle video good for improving your intuition about how pulleys work. This article goes into more depth. It is especially useful for part (c).

The answer to (a) is that mass pulls down with force $F_g = mg$, so to counterbalance it, the tension in the rope must be mg.

Hint:

First, consider any of the ropes. Any point on the rope has the same tension as any other point on that same rope. If this rope is vertical, for example, this means that everything above that rope is being dragged down with force equal to the tension, and everything below is being dragged up with that same force.

Now, consider any of the pulleys. If you draw a box around a pulley, there will be ropes coming out of that box in various directions. If the pulley is not moving anywhere, this means all the tensions in those ropes must cancel each other out.

These two principles are enough to solve the problem.

10 points:

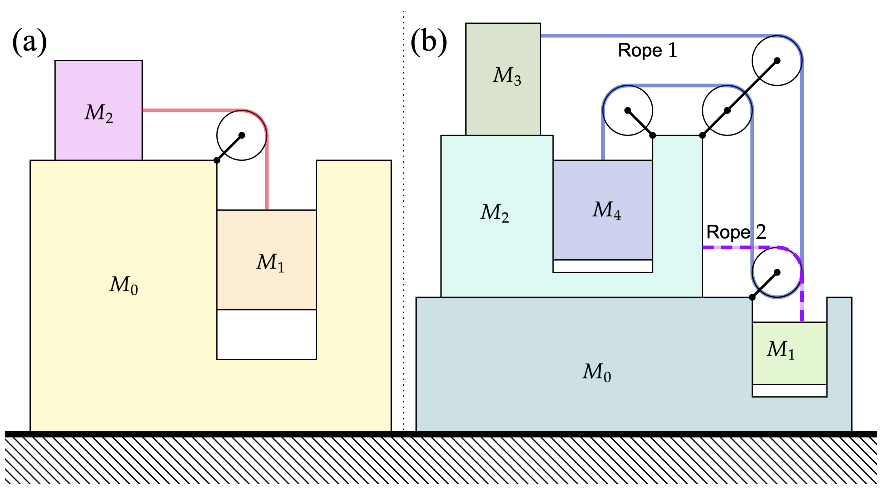


Figure 2: Diagram for the 10 point Physics problem.

(a) (2 points) Consider the system of blocks and pulleys drawn in in Fig. 2(a). All blocks are frictionless. Pulleys and ropes have no mass. In the absence of all external forces besides gravity, find the acceleration of M_0 in terms of M_1 and M_2. This is a standard problem – using external resources in your solution is allowed, but you must cite them.

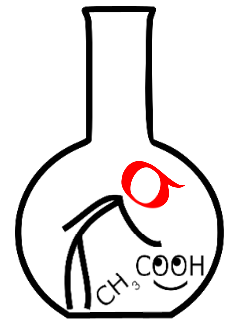
(b) (8 points) Consider the block and pulley system shown in Fig. 2 (b). Find the initial acceleration of M_0 in terms of M_1, M_2, M_3, and M_4. Make the following simplifying assumptions:

- There is no friction between blocks, between M_0 and the ground, or between the rope and the blocks.
- There is no slipping between either rope and the pulley. In particular, note that both ropes are wrapped around the *same* pulley.

- The sizes of the drilled holes are such that M_1 and M_4 can only move vertically, not swing horizontally.

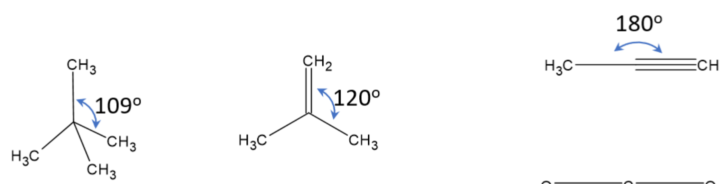
Hint:

Part (a) is a very standard textbook problem – some googling may give a pedagogical solution, for example here. You are welcome to use these sources, but you must cite them. Problem (b) may be solved by the same methods as (a).

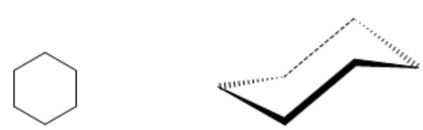


Chemistry

In the second half of the 19th century, chemists realized that molecules are not random heaps of atoms: their atoms are arranged according to rules, and each molecule has a definite three-dimensional shape. They later showed that a molecule's geometry can be inferred from the types and numbers of bonds between its atoms. For example, a carbon atom with four single bonds has tetrahedral geometry with bond angles of about 109.5°. A carbon involved in a double bond is trigonal planar with angles near 120°. A carbon in a triple bond is linear, giving 180° bond angles; the same linear geometry occurs when a carbon forms two double bonds to two different atoms (as in CO2).



When an organic molecule contains only single bonds, it tends to adopt conformations with bond angles close to 109.5°. Single bonds usually allow rotation, which lets the molecule relieve strain. Cyclohexane, for instance, would have 120° angles if it were a flat hexagon, but it never adopts that planar form because it would be too strained. Instead, the ring puckers into the chair conformation, in which the C–C bonds are staggered (wedged bonds point toward the viewer and dashed bonds away from the plane of the page, see figure below).

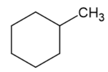


What if the optimal angles cannot be achieved, even after bond rotation? That occurs, for example, in small rings such as cyclopropane and cyclobutane (three- and four-membered rings). In such cases the molecule experiences ring strain—especially angle and torsional strain—because the bonds cannot adopt their preferred geometries. These molecules can exist, but they are higher in energy than expected. Strain can arise not only in very small (and sometimes very large) rings, but whenever a molecule cannot adopt a conformation with near-optimal bond angles and torsional relationships. Significant strain raises a compound's energy, making it more reactive and increasing its heat of combustion. If the strain is extreme, the molecule becomes too reactive to isolate and is considered unstable.

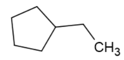
To help with the following problems, you may build models of these molecules using toothpicks and clay beads and examine how their valence-bond angles are distorted. To simplify modelling, you may represent methyl group as balls (just make sure their radius is realistic: not too big and not too small). When building the models, remember that H–C and C–C bond lengths differ. **To facilitate grading, please attach the pictures of the models you made for both problems, along with brief explanations in one PDF file.**

5 points:

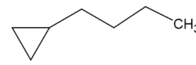
The hydrocarbons 1-5 shown below have the same empirical formula, C_7H_{14}.



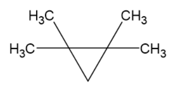
1



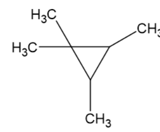
2



3



4



5

Consequently, their combustion produces the same amounts of CO_2 and H_2O, so the balanced combustion equation is identical for all five. However, their heat released during combustion is different. Rank compounds 1-5 from highest to lowest heat release during combustion. As mentioned above, you may build models of the molecules to help you, and attach images.

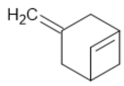
Hint:

As you can see, all these molecules contain only two types of bonds—carbon–carbon and carbon–hydrogen—and each molecule has the same number of each. Their combustion products are also identical, and the numbers of H_2O and CO_2 molecules formed are the same. In any reaction, however, the energy released equals the difference between the energies of the reactants and the products.

A compound's energy is often approximated as the sum of its bond energies. By that logic, these compounds should produce the the same heat during combustion. Yet they do not, because their geometries differ: in some molecules, the bond angles are suboptimal, introducing strain. Clearly, when strained molecules burn, the energy of that strain is released, thereby increasing heat of combustion. If you identify which bond angles are suboptimal and the extent of the geometric distortion, you can arrive at the correct explanation.

10 points:

The image shows several organic molecules drawn in standard notation (carbon atoms are implied at the vertices, and hydrogens on carbon are omitted, except in methyl groups).

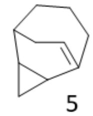


1

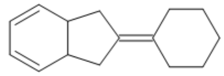
2

3

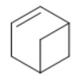
4



5

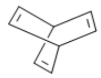


6



7

8

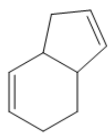


9

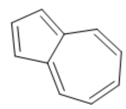
10

11

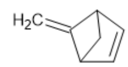
12



13

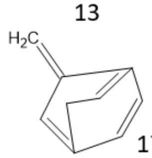


14

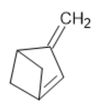


15

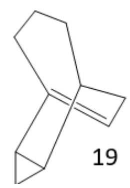
16



17



18

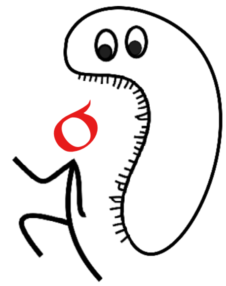


19

Given that compounds 2–4 exist at room temperature, whereas 1 and 5 do not, identify which other compounds in the set are stable and can be prepared in pure form. As mentioned above, you may build models of the molecules to help you visualize these molecules and identify possible steric strain, and attach images.

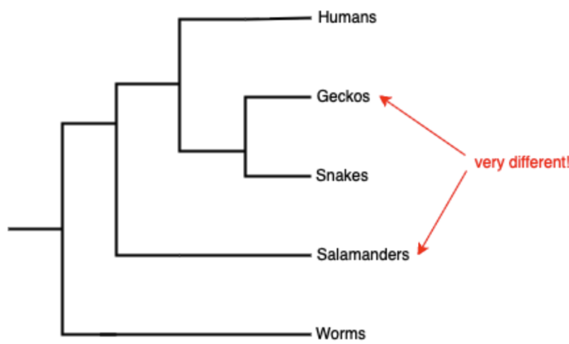
Hint:

Consider the geometry of carbon atoms involved in double bonds. Ideally, they are perfectly planar: the carbon atom, its double-bonded partner, and the two other atoms attached to it all lie in the same plane. Any deviation from planarity—even by a single atom—comes with a significant energetic penalty.

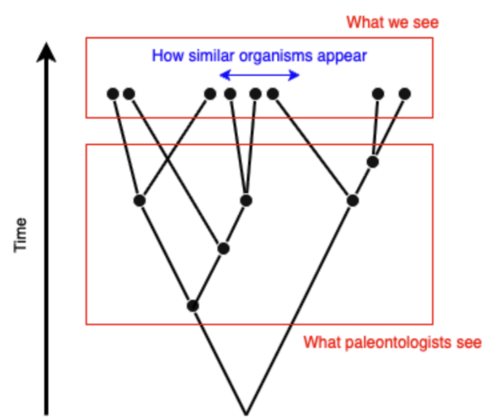


Biology

Some species can look similar morphologically, but be very different in terms of their biological function and evolutionary history, and vice versa. For example, salamanders and geckos are actually very distant types of organisms, even more so than geckos and humans. Snakes are even more distant from worms. Clearly, morphological adaptations don't tell the full story about what makes species different.



An even more drastic example of this effect can be found in the microscopic world. Microbes that are vastly different in terms of how they function as organisms can be very hard to tell apart by sight. Even genetic analysis can prove complicated because of effects like *horizontal gene transfer* and *endosymbiosis*.



Whether it's animals or bacteria, just looking at an organism isn't enough to tell you about its history and biology.

Below are some optional interactive resources that may help you think through this problem or explore it further. You are not required to use them, they are provided as additional references if you'd like more context or guidance:

Tree of Life Web Project

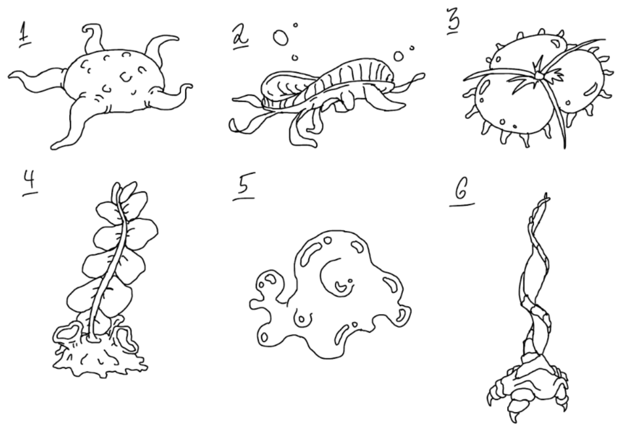
One Zoom Tree of Life Explorer

5 points:

The Sigmanauts discovered a distant planet teeming with life, and two teams of biologists began studying its inhabitants. The first team concentrated on the anatomy and morphology of the local creatures. Based on their observations, they proposed that all species could be divided into six major groups according to outward appearance and lifestyle:

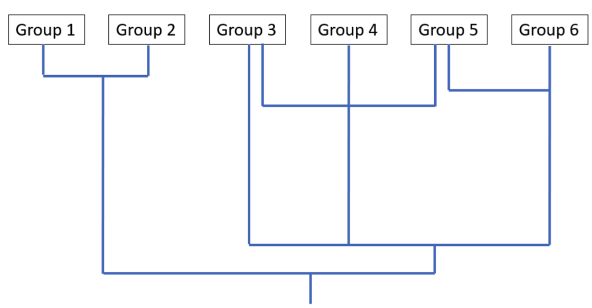
1. Animals with five legs.
2. Freely swimming, disk-shaped creatures with seven fins.
3. Animals displaying threefold symmetry.
4. Sedentary, feather-bodied forms.
5. Amorphous, blob-like organisms.
6. Multi-legged animals exhibiting screw-axis symmetry.

They also made some drawings of representative organisms from each of these six groups:



Meanwhile, the second team of biologists focused on genomic research. They quickly learned how to read the genetic code of local organisms, sampling 10–15 species from each of the six groups. By sequencing their genomes, they were able to reconstruct an evolutionary tree that traced all modern lineages back to a common—but now extinct—ancestor.

The tree is shown below.



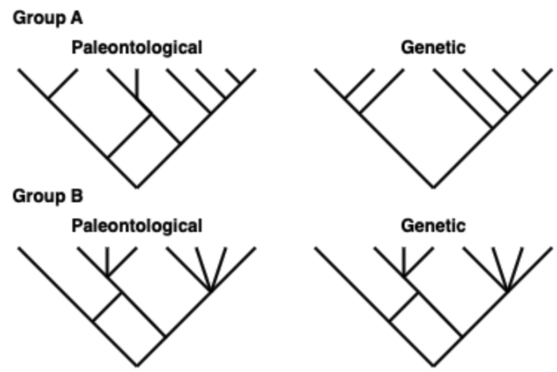
How do you explain the architecture of this tree? Which evolutionary events could have led to this topology? Provide similar examples from the Earth's evolutionary tree. In your answer, assume that the evolution on that planet occurs mostly by the accumulation of mutations. Be specific.

Hint:

No hint this month.

10 points:

The Sigmanauts then traveled to another planet with life. They looked at two different groups of animals and similarly prepared two evolutionary trees for each group; one constructed only from paleontological data, and one from genetic analysis.



One of these groups of animals was extensively subjected to interspecific gene transfer early in its evolutionary history, and the other was not. Which is which?

Hint:

No hint this month.

Linguistics & Applied Sciences

5 points:

Here are phrases in an artificially constructed language and their translations, **not all in matching order**:

A. Ko gerazami rakoti juzati	1. *A short rabbit*
B. Tiratizaju migeju	2. *The dog runs quickly*
C. Ramirage ramizage	3. *A tall giraffe*
D. Tira mijuzako koge	4. *The rabbit sees a cat*
E. Tiratiraza jugemi	5. *The sleepy cat runs*
F. Ko gerazara juzati gezamira	6. *A slow car*
G. Timi mijutimi tira	7. *I dance poorly*
H. Ko tiratiraza timirami gerazami	8. *You calm me*

Match each phrase to its translation.

Translate the following phrase from the language into English:

Ko tiratizaju timirami geko

Translate the following phrase from English into the language:

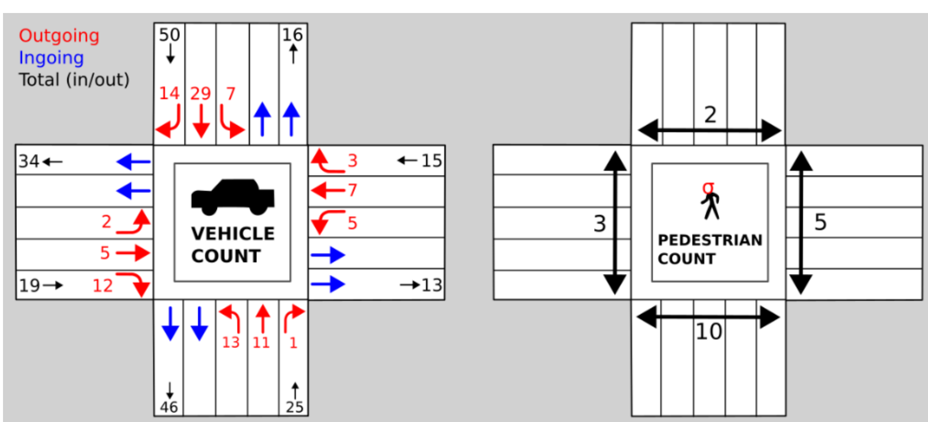
The sleepy dog worries you

Hint:

Here are 3 matches: A5, F2, H4

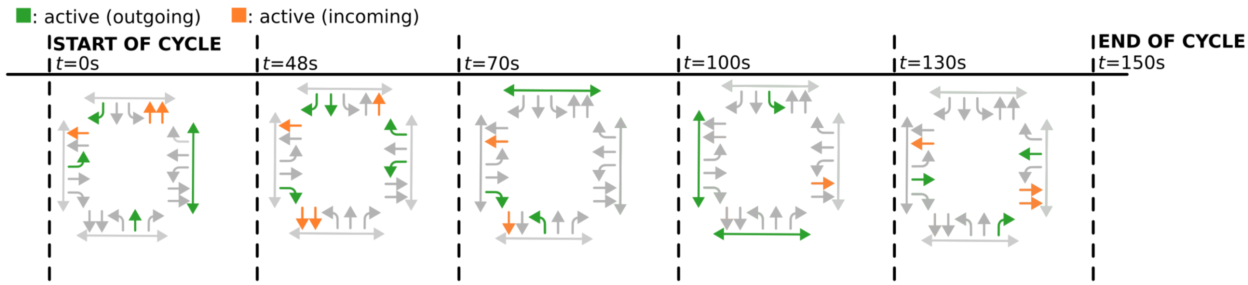
10 points:

SigmaCity's planning committee is updating the Sigma St & 2025 Ave intersection, citing historically bad congestion. Each side of the fourway intersection has 5 lanes and a pedestrian crossing.



It takes $T = 28 + 2N$ seconds for N pedestrians to cross the street. In 30 seconds, 5 cars can make a turn and 7 cars can go straight across. Remember that 2 lanes can not zipper merge into 1 lane.

A proposed schedule is shown below:



(a) Identify as many problems with the proposed schedule as you can.

(b) Adjust the cycle of the lights to safely allow the number of vehicles/pedestrians indicated in the first figure above to pass the intersection in one cycle (i.e. in one cycle 50 vehicles from the top should get through, 19 from the left, etc...).

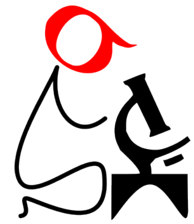
(c) Design your own schedule that minimizes the duration of the cycle while allowing for the same number of cars and pedestrians.

Explain any additional assumptions you made about the vehicles and the pedestrians.

Note: you do not have to specify your design the same way as the proposed schedule, but it must be clear what lanes/crossings are active at what times, and what the length of your cycle is.

Hint:

No hint this month.



Project (30 points)

Mathematics and Computer Science

This project will ask you to explain some of the theory behind the Fibonacci sequence and related sequences, and to implement algorithms for calculating them. Some of the questions are introductory and others are more involved. We recommend that you consider all the questions in the order they are written, but you are not required to solve any earlier questions to present later ones.

There are many parts and subparts to the project. You may choose to implement the algorithms and/or provide a theoretical answer to any of them. You are not expected to answer and/or implement all of them to get a full score. The theoretical and coding portions will be graded separately. In particular, code with a minimal mathematical explanation, or a theoretical explanation without any accompanying code will both score points.

Given the open nature of some of the questions, the exact scoring will be determined after all submissions are collected and reviewed. As a general guidance, the mathematical part and the implementations will each be worth 15 points total.

The *standard Fibonacci* sequence is a sequence which starts with $1, 1, \ldots$ and where each successive element is the sum of the two elements that precede it. Namely, if we let F_n be the n-th Fibonacci number, we have $F_1 = 1$, $F_2 = 1$, and for each subsequent number $n > 2$, we define $F_n = F_{n-1} + F_{n-2}$. As you most likely have seen before, the first few elements of this sequence are $1, 1, 2, 3, 5, 8, 13, \ldots$

The main goal for this project is to describe and implement algorithms to find F_n and related numbers for large and very large n as quickly and precisely as possible.

There are terms related to the Fibonacci sequence, other similar sequences, and methods to compute these sequences throughout this problem. You are encouraged to use any available sources of information to learn about these terms, ideas, and methods.

The project consists of two parts: Part A is concerned with the Fibonacci sequence, and Part B is concerned with its generalizations. **You do not need to complete all questions to get full points**; for example, you can solve any three questions from Part A and any two questions from part B, or provide partial solutions for all 7 questions.

Part A: Fibonacci sequence

You may choose 3 out of 4 questions to complete in Part A to receive full credit.

1. Besides the standard recursive definition $F_n = F_{n-1}+F_{n-2}$, there are other clever ways to compute Fibonacci numbers. Pick one of these options, research it, and explain how and why it works:

- an explicit formula for F_n using the golden ratio (aka *Binet's formula*);
- matrix multiplication;
- anything else you can find or think of, that is not a straightforward implementation of the definition by a recursive or iterative algorithm.

2. Implement (using Python or Java) **at least two** of the following options for the standard Fibonacci sequence. Your program should output F_n for any n (including large n, up to $n = 10{,}000$), where n is given as input. The possible methods include:

- recursion;

- a `for` loop without recursion;
- matrix multiplication;
- an explicit formula for F_n using the golden ratio (aka *Binet's formula*);
- anything else you can find or think of.

If any of your solutions do not use the recursive definition directly and were not explained before, explain why your program is computing F_n with any needed mathematical details.

3. Using one of the algorithms you wrote in Question 2, or an online list of Fibonacci numbers, determine the number of decimal digits in each of the 20 numbers $F_{100}, F_{200}, \ldots, F_{2000}$.

Create a plot of the number of digits in F_n vs. n; you should see a straight line. What is the slope of this line? Can you connect the slope to the numbers in the explicit formula for F_n?

4. A central concern of algorithms is *runtime*: how long an algorithm takes to run. Any algorithm which finds F_n given n has some runtime $T(n)$. Describing the runtime $T(n)$ exactly is often hard, so we instead focus on the general "shape" of the runtime.

For example, if an algorithm takes some constant time c_{start} to initialize, then loops n times with each loop taking c_1 time at each step, then the total runtime of the implementation is $T(n) = c_{start} + c_1 n$. As n grows, the runtime $T(n)$ becomes almost proportional to n, and we say the runtime is linear. (This notion of *asymptotic* behavior of $T(n)$ as n gets very large is formalized using *big-O notation*, which you may use, but it is not required.)

(a) Before making empirical measurements, make a theoretical prediction for the runtime of two or more of your algorithms in Question 2. Is the runtime proportional to n? To n^2? Some other function of n? Be sure to consider very large n; the runtime depends on the behavior of your algorithm as n grows.

(b) Now, let's validate your prediction with measurements. In Python, this can be done with the `time` package. Here's an example of how you can use it:

```
import time

def fib(n):
    ...

def time_fib(n):
    t0 = time.time()
    fib(n)
    t1 = time.time()
    print("My code runs in" + t1-t0 + "seconds")
```

In Java, you can time your code using the `System.currentTimeMillis()` function.

Create scatter plots of the amount of time it takes your algorithms to evaluate F_n for at least 20 values of n covering the range $1 < n < 20{,}000$. Do the graphs validate your predictions? (If you are using Python, you are likely to be surprised at this step.)

Part B: Generalizations

You may choose 2 out of 3 questions to complete in Part B to receive full credit.

5. We can define different sequences (sometimes called Lucas sequences) by using the same equation $L_n = L_{n-1} + L_{n-2}$ for $n > 2$, but different initial conditions $L_1 = p_1$, $L_2 = p_2$. For the standard Fibonacci sequence, $p_1 = p_2 = 1$.

(a) Using *generating functions* (see any of the links here, in particular this one), *characteristic polynomials*, or any other method of your choosing, derive the explicit formula for the case $p_1 = 3$, $p_2 = 10$. Use it to find L_{10}; confirm your result by implementing a recursive or iterative algorithm to compute it.

(b) Let us go a bit farther and change the equation. For example, if $L_n = 2L_{n-1} + 3L_{n-2}$ for $n > 2$, $L_1 = L_2 = 1$, what is the explicit formula? Use it to find L_{10}; confirm your result by implementing a recursive or iterative algorithm to compute it.

6. To generalize even further, we can have a recurrence relation that goes more than 2 terms back, the most famous being $T_n = T_{n-1}+T_{n-2}+T_{n-3}$ (sometimes called tribonacci numbers). We will define the P-numbers p_n, which start with 1, 2, 3, ... and follow the recursive relation $p_n = 2p_{n-1} + p_{n-2} - 2p_{n-3}$.

The characteristic polynomial for $p_n = 2p_{n-1} + p_{n-2} - 2p_{n-3}$ has easy-to-guess roots. Using it, or some other method (like generating functions), derive an explicit formula for p_n, given that $p_1 = 1$, $p_2 = 2$, $p_3 = 3$, use it to find p_{10}, and confirm your result by implementing a recursive or iterative algorithm to compute it.

7. In this question, use any of the techniques you have learned to compute elements of a general constant coefficients recurrence relation. Namely, consider a sequence defined by $p_n = a_1 p_{n-1} + \cdots + a_k p_{n-k}$ for some fixed k and for all $n > k$, with given values of $p_1, \ldots, p_k$.

Create a function in your program named `general_recurrence()` that reads a file `input.txt` that contains 4 lines:

- The first line will contain the number k.
- The second line will contain k numbers $a_1, \ldots, a_k$.
- The third line will contain k numbers $p_1, \ldots, p_k$.
- The fourth line will contain a single positive integer n.

Your program will then compute and output a single number p_n in an output file `output.txt`.

Project Submission Instructions

- Submit a **video at most 6 minutes long** describing your solutions to all the questions you answered (both theoretical and coding). Explain your answer, how you arrived at it, and what resources you used. For coding questions, explain how your code works, as well as any intricacies about your code that we should know when reading it.
- For the theoretical questions, submit **a single PDF file** containing your solutions to the theoretical questions.
- For the coding questions, submit **a single Python-3 or Java code file** containing all of the coding parts with each question in a different function.

Hint:

No hints this month.