PROBLEM OF THE MONTH

POM

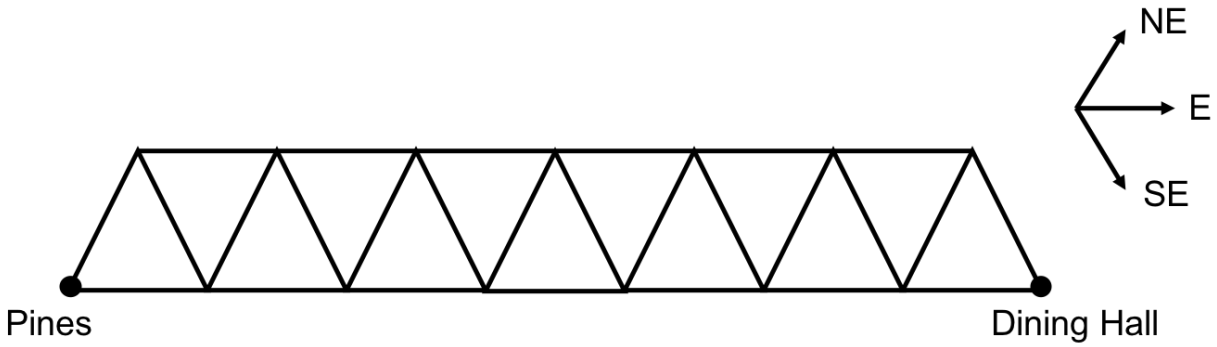
2018 - 2019

October, 2018

MATHEMATICS

5 points:

A Sigma-camper leaves Pines and walks to the dining hall along the paths shown in the image. In how many ways can he reach the dining hall if he is only allowed to walk east, northeast, and southeast?



Hint:

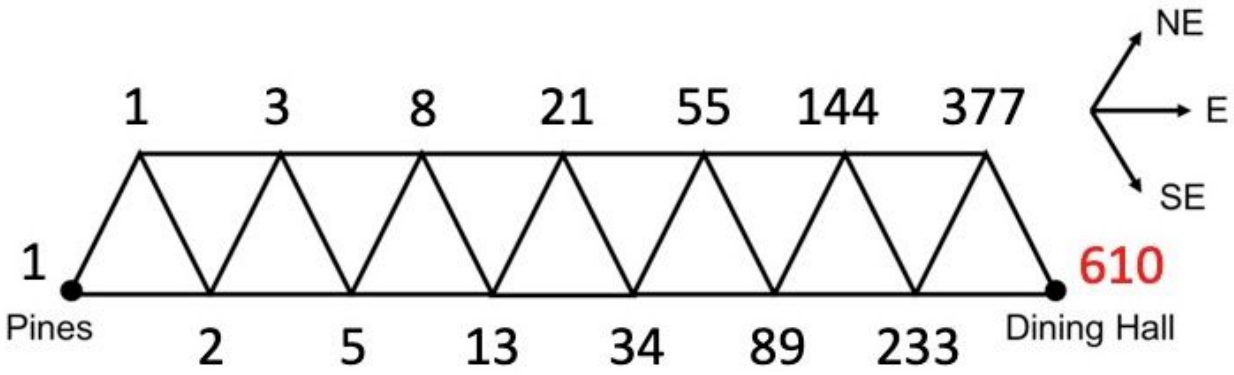
Try to write down at each vertex of the graph the number of ways leading to it.

Answer: 610

Solution:

Let us write at each vertex of the graph the number of different ways leading to it from Pines. Starting from left to right we have 1,1,2,3,5,... . It is easy to see that one can come to a given vertex from either of two vertices from the left. Correspondingly the number of ways leading to a given vertex is equal to the sum of numbers of the previous two previous vertices on the left. One can easily continue the sequence using this rule (see the picture) 1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,.... The correct answer at the “Dining Hall vertex” is 610. The remarkable numbers of that sequence are called Fibonacci’s numbers. They appear in

the work of Italian mathematician Leonardo Fibonacci in year 1202. These numbers have a lot of very interesting properties and applications. You can find some of those on: https://en.wikipedia.org/wiki/Fibonacci_number



10 points:

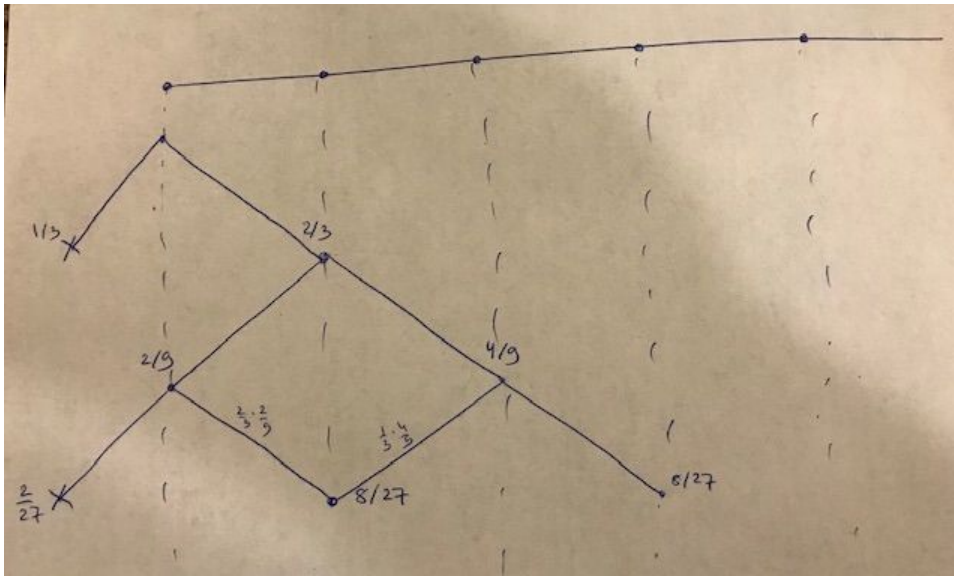
The Sigma robotics team built a stepping robot, positioned it at the brink of an abyss at the bank of the Silver Lake and sent it to reach a destination by following a narrow trail, where the robot can only move in one direction, either making a step forward, or a step backward. However, the robot somewhat lost its sense of direction. Instead of moving directly away from the abyss, it makes steps randomly, but a step away from the abyss is two times as likely as a step towards it.

a) What is the probability that the robot falls into the abyss before making six steps?

b) What is the probability that the robot eventually falls into the abyss?

Hint:

Try to make a tree encoding various paths of the robot in the following way:



Answer:

a) 107/243

b) ½

Solution:

a) Let us denote $p(t,n)$ the probability that the robot is at the distance of n steps from the abyss at time t. Initially, at $t=0$ we have $p(0,-1)=0$, $p(0,0)=1$, and $p(0,n)=0$ for $n=1,2,3,\ldots$. We assumed that $n=-1$ corresponds to robot falling into abyss. We have the following rules for finding probabilities at later times

$p(t+1,-1)=p(t,-1)+\frac{1}{3}p(t,0);$

$p(t+1,0)=\frac{1}{3}p(t,1)$

and

$p(t+1,n) = \frac{2}{3}p(t,n-1) + \frac{1}{3}p(t,n+1),$ for $n = 1,2,3,...$

Using these rules we easily find probabilities after first 6 steps.

t=0: 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

t=1: 1/3, 0, 2/3, 0, 0, 0, 0, 0, 0, 0

t=2: 1/3, 2/9, 0, 4/9, 0, 0, 0, 0, 0, 0

t=3: 11/27, 0, 8/27, 0, 8/27, 0, 0, 0, 0, 0

t=4: 11/27, 8/81, 0, 8/27, 0, 16/81, 0, 0, 0, 0

t=5: 107/243, 0, 40/243, 0, 64/243, 0, 32/243, 0, 0, 0

t=6: 107/243, 40/729, 0, 16/81, 0, 160/729, 0, 64/729, 0, 0

Here the left number is the probability of being in abyss. We conclude that the probability for the robot to fall into abyss in 6 steps is equal to $107/243 \approx 0.44$.

b) Let us denote the probability for robot to fall into the abyss starting from the edge of the abyss p_0. It is clear that this probability is the sum of the probability of the robot stepping towards the abyss (⅓) and the probability of stepping out of the abyss (⅔) multiplied by the probability p_1 for the robot to fall into abyss if it starts from the point one step from the abyss. We obtain that $p_0 = \frac{1}{3} + \frac{2}{3}p_1$. Now comes the most non-trivial part of the solution. If the robot starts from the point $n = 1$ it can only fall to the abyss by first getting to the point $n = 0$ and then by falling to the abyss starting from the point $n = 0$. Both of those probabilities are equal to p_0 and we obtain the relation $p_1 = p_0^2$. Combining these two equations we have

$p_0 = \frac{1}{3} + \frac{2}{3}p_0^2$

This quadratic equation has two roots $p_0 = 1,\ 1/2$. The correct solution is given by $p_0 = 1/2$. Compare with the probability to fall after first 6 steps 0.44 obtained in the part a).

One can generalize this problem to the robot stepping from the abyss with the probability p and out of the abyss with the probability $1-p$. Then the probability of falling eventually into the abyss is $\frac{1-p}{p}$ if $p \geq \frac{1}{2}$ and 1 if $p \leq \frac{1}{2}$. Remarkably, the probability is 1 for $p = \frac{1}{2}$. This problem is very important in the probability theory. It is directly related, for example, the the problem of “Gambler’s ruin”

https://en.wikipedia.org/wiki/Gambler%27s_ruin

PHYSICS

5 points:

By measuring the work that a horse can do within a certain time, James Watt defined the horsepower as 33,000 ft·lbf/min, which in modern units this is approximately 745 Watt. What is the maximum velocity that a horse of a single horsepower can pull the carriage weighting 745 kg up the road of a constant slope, which rises by 10 m per every 100 m travelled? Neglect the losses due to friction and other dissipations in the carriage.

Hint:

Consider the rate of change of the potential energy of the carriage.

Answer:

$v = 1\ m/s = 3.6\ km/h$

Solution:

When carriage travels distance ΔL, its potential energy changes by $\Delta E = 0.1\Delta L mg$. If carriage travels with velocity v, the time required to cover distance ΔL is, $\Delta t = \Delta L/v$. The rate of energy change, which determines the required power is, $W = \Delta E/\Delta t = 0.1mg\Delta L/\Delta t = 0.1mgv$. Hence, the velocity that corresponds to one horsepower, 745 W for a 745 kg carriage is, $v = W/(0.1mg) = 1m/s = 3.6km/h$.

10 points:

In times before the round wheel was invented, carriages in Sigmaland were equipped with square wheels. How many horses of 1 horsepower (745 Watt) are needed to pull such a carriage weighting 745 kg and having square wheels of 1m side with the velocity 5 km/hour on an even, horizontal road? Assume that collision of a square wheel with the road is fully inelastic.

Hint:

Consider the work needed for the square wheel to make a quarter turn. Note that all this energy is dissipated in an inelastic collision after each quarter turn.

Answer:

11 horses.

Solution:

In order to displace the carriage by 1 m, which equals to the side of the square wheel, the center of mass of the carriage needs to be lifted by half the length of its diagonal minus half side, $(\sqrt{2}-1)/2$ (we assume all wheels rotate synchronously). The corresponding potential energy change is, $\Delta E = mg(\sqrt{2}-1)/2$. When wheel falls on the ground, this energy is lost in an inelastic collision and then the cycle repeats. Moving carriage with velocity v requires power, $W = \Delta E/\Delta t = mgv(\sqrt{2}-1)/2 = 745 \cdot 10.36$. Hence, 11 horses are needed.

CHEMISTRY

5 points:

Alice, a college faculty, came to her lab and found that 8 bottles of chemicals on the shelf had unreadable labels. She told Bob, her technician, that, according to safety rules, these chemicals should be discarded. According to their inventory, the solid chemicals in these bottles were: sodium chloride, potassium fluoride (dihydrate), glucose, anhydrous potassium carbonate, sodium thiosulfate, sucrose, sodium iodide, and salicylic acid.

Alice decided not to drop all chemicals into one waste container, and, to facilitate their utilization, she asked Bob to put them into two separate waste containers: one for inorganic and one for organic waste. “How can I decide which chemical should go into which container?” Bob asked. “Their labels are unreadable.”

“Bob, try to do some simple test,” Alice replied. “It is not too difficult to distinguish inorganic compounds from organic ones, especially when they are solids.”

The next day, Bob told Alice that he had separated the inorganic and organic compounds, and, according to his test, 3 bottles appeared to be inorganic, and 5 bottles were organic. “Hmm,” Alice said, “Something is wrong. According to the list, 5 compounds were supposed to be inorganic ones, and only 3 were organic. Which test did you use, Bob?”

“Alice,” Bob replied, “From my high school chemistry course, I know that ionic inorganic compounds are solids with a high melting temperature, whereas all organic solids have a low melting point. I took small samples from each bottle and started to heat them gradually. The samples from five bottles melt at temperatures below 250-300 degrees, and some of them even melted below 100 degrees. They are definitely organic compounds, I am pretty confident there is no mistake here.”

“My dear Bob,” Alice replied, “Your rationale looks correct, but you missed one point. Some of those compounds were ….”

Please, continue Alice’s statement and explain what was wrong with Bob’s method. How could his method be modified to make it work?

Hint:

Bob is right that inorganic salts (a.k.a.ionic compounds) have very high melting temperature. However, the salts in the Bob list are hydrates. Read about “water of crystallisation” and “hydrates”. What happens when a hydrate is heated (especially when a hydrate salt is very soluble in water)? Look at the “hot ice” experiment on Youtube and explain what would Bob have observed if, for example potassium fluoride and potassium fluoride *hydrate* are heated to 100 degrees.

Solution:

In solid state, many inorganic ionic compounds (a.k.a salts) contain water molecules. These compounds are called “hydrates”, and water molecules occupy some concrete positions in a crystal lattice, and they are an integral part of the crystal’s structure. An example of such a hydrate is cupric sulfate pentahydrate ($CuSO_4 \cdot 5H_2O$), which forms very beautiful blue crystals. The bonds between water molecules and the atoms of a salt are weak (it is not considered as a real covalent bond), so when hydrate crystals are heated, water molecules are released. For example, when cupric sulfate pentahydrate is heated, water evaporates, and cupric sulfate (without water) forms, which is a white powder. This process is reversible: when water is added to anhydrous cupric sulfate, a blue color restores.

So far, it is not too interesting. What is more interesting is behavior of hydrates of salts that are very soluble in water. An example is sodium thiosulfate, which usually exists in a form of a pentahydrare $Na_2S_2O_3 \cdot 5H_2O$. This pentahydrate forms very nice colorless crystals, and when these crystals are heated, water is released, and the amount of water is sufficient to dissolve sodium thiosulfate. In other words, heating of sodium thiosulfate crystals releases water that immediately dissolves this compound, so to a non-educated observer, the process looks like melting of these crystals.

When “molten” thiosulfate are cooled down, the crystal structure is restored, and the liquid solidifies. The same effect explains the famous “Hot ice” experiment (several videos of this experiment are available of Youtube)

(- A note. *That is not a real melting, because if you continue to heat this liquid, water will evaporate, and you will get a solid, i.e. anhydrous sodium thiosulfate* $Na_2S_2O_3$).

That is the same effect that Bob observed when he tried to melt inorganic compounds. Had he tried to heat “molten” sodium thiosulfate and potassium fluoride at higher temperature, he would observe evaporation of the liquid and formation of a solid residue. That is a typical behavior of inorganic salt hydrates.

10 points:

"*We had two bags of ferric chloride, seventy-five ounces of sodium hydroxide pellets, five kilograms of high purity citric acid, a saltshaker half-full of aspirin, and a whole galaxy of multi-colored pH papers, rubber balloons, strings etc... Also, a quart of isopropanol, a quart of acetone, a case of Poland Spring water, a pint of raw ether, and two dozen grams of isoamyl alcohol. Not that we needed all that for our graphomaniac exercises, but once you get locked into a serious chemicals collection*, *the tendency is to push it as far as you can. The only thing that really worried me was the ether. There is nothing in the world more helpless and irresponsible and depraved than a man in the depths of an ether binge, and I knew we'd get into that rotten stuff pretty soon.*"

Using the stuff described in this quote, can you prepare a dark ink? Which items listed here are needed for that, and how will you do it?

Hint:

An old procedure for preparation of ink consisted in boiling iron salts with oak bark, oak apples, or similar natural materials that are rich in phenols (phenols are the compounds that have an OH group attached to a benzene ring). Can we prepare a compound having a benzene ring with an OH group using the materials from our list? (One more hint: a reaction that we have to do is called “alkaline hydrolysis”)

Solution:

Phenolic compounds form complexes with trivalent iron ions Fe^{3+}. These complexes have a deep color (usually deep blue or deep purple, depending on concrete phenol used in the reaction). You can see this reaction on Youtube use the key words *ferric chloride phenol*. We have ferric chloride in our set, but we have no phenol. That is a problem. How can we get it?

You probably know that a scientific name of aspirin is “acetylsalicylic acid”. If you look at the formula of salicylic acid (it is googlabe, find it by yourself), you will see that it has a hydroxy group attached to a benzene ring, which means it is a phenol (it is a carboxylic acid too, however, that is normal when an organic compound belongs to more than one class of compounds). Therefore, we actually *do* have a phenol, more precisely, phenol’s *precursor*. That means, we have to convert acetylsalicylic acid into salicylic acid (i.e. to remove the acetyl group from the phenolic hydroxide (google the formula of acetylsalicylic acid, a.k.a. aspirin, to understand what I mean).

How can we do that? Usually, the acetyl group can be removed by treatment with alkali. We do have alkali in out set (sodium hydroxide), so that means we have to take some amount of sodium hydroxide pellets, dissolve it is minimal amount of water, add aspirin, and heat the mixture. After some time, cool the mixture, make the solution neutral with citric acid (other acid can be used too, but we have citric acid in our set; we also have pH paper for that), and add the material we obtained to the ferric chloride solution. A deeply colored solution that you will obtain can be used as an ink.

BIOLOGY

If you look through the history of important discoveries in biology and medicine, you may notice a very common pattern: a natural or an experimentally-induced pathology often yields insight into normal biological function. As one former Cold Spring Harbor professor has described it, half-joking, if a biologist were to study a radio, they would shoot it with a shotgun until the radio stopped working, and then identify the broken parts as obviously being critical to the radio's function. Funny as that sounds, careful analysis of anomaly and disease can turn into a very useful strategy when trying to understand normal physiology. Moreover, this approach holds true at many levels of experimental biology, from the biochemistry of single cells to the behavior of complex primates. The questions you are about to work on are all examples of just this principle.

First we will consider an example that takes place on a physiological macro-scale, focusing on the evolution of human brain anatomy.

5 points:

A patient who was involved in an accident suffered extensive trauma to the back of his head. Having imaged his brain, the doctors determined that his occipital lobe was extensively damaged, and that the patient would be blind. The hospital where the patient received treatment happened to be a research hospital, so as part of the follow up, the patient was signed up to be part of a study of vision disorders conducted on the premises. When given a normal vision test, the patient reported to be totally blind – he could not identify any shapes, colors, or movement.
During the second part of the study, images of real life objects were presented to the patient while the researchers measured the patient's skin conductance. This parameter reflects the readiness of the skin to sweat in response to arousal, part of the fight-or-flight system, which is the organism's instinctive reaction to danger. To their big surprise, the researchers found that any time the patient was shown an image containing a snake-like object, their device detected a spike in skin conductance, meaning that the patient was having an emotional response to these images. Other images produced no such response. Same experiments with healthy individuals who participated in this study as a control group demonstrated the same spikes in skin conductance when snake-like objects *as well as other generally threatening images* were presented to them, and no change in conductance was observed when neutral images were presented.

Please provide an explanation of the results observed in this study. What does this study tell us about the evolutionary history of the primate brain?

Hint:

When you are asleep or unconscious your pupils still contract in response to a bright light. When some object moves quickly toward your eyes, you blink even if you haven't realized what was happened. This means your brain is still capable of processing some visual information even if your mind is not involved in it. Does your cortex play a role in this process?

Answer:

Humans have one of the most advanced vision systems to ever evolve, second perhaps only to birds of prey. This is thought to be in large part due to our use of visual stimuli as a primary means of interaction with the world. We rely on visual stimuli for everything from finding and consuming food (attraction to color, grasping), to avoiding danger (predator avoidance, ability to see through camouflage), to building social interactions (rapid detection of eye movement and emotion in other humans). This is a rather unique situation from an evolutionary perspective. For example, unlike us and other closely-related primates, rats rely on smell in order to grasp for food, and can properly feed themselves even after their vision has been compromised. One likely reason why we rely on vision as much as we do is that from early on in primate evolution, vision was selected as the primary defense mechanism against our most common predators – constrictors, and later venomous snakes. This argument is summed up in what is known as snake detection theory. First postulated by Dr. Lynne Isbell in 2006, it sums up evidence from multiple studies of the mammalian fear response and primate visual pathways going back to the 1980s and 90s.

The basis for the theory lies in the fact that visual information is processed from the retina along multiple structures of the brain. Processing of visual information into conscious understanding is the function of the visual cortex, located in the back of the head in the area known as the occipital lobe. However, the visual cortex itself is not directly connected to the retina, and the information it receives is actually first passed through areas of the midbrain, amygdala and the hypothalamus, which are in themselves much more ancient structures that pre-date consciousness as we understand it. These structures are the first to receive visual input from the retina and can trigger an immediate and complex response, although we do not realize it. The action of the hypothalamus (more specifically, the pulvinar nuclei region) and the amygdala are enough to produce a motor and emotional response to a predatory threat, long before the cortex tells us what it is that we are experiencing.

In humans, damage to the visual cortex results in loss of conscious awareness, also known as blindsight. Like the patient described in our question, humans with blindsight are still able to locate visual targets despite having no awareness of having seen the targets. Blindsighted macaques can move their heads out of the way of a stimuli that appears to suddenly move towards them. Conditions of blindsight can be imposed experimentally in healthy people as well: all that is necessary is to flash an image so fast that the conscious structures of the brain have no time to perceive it. Curiously, blindsight simulation experiments have been done on people with snake phobia. As you may be guessing by now, these people developed physiological signs of anxiety when shown pictures of snakes, though not when they were shown other images.

Isbell LA., Snakes as agents of evolutionary change in primate brains, Journal of Human Evolution, 2006 July; 51(1): 1-35
https://www.ncbi.nlm.nih.gov/pubmed/16545427

Ohman A, Soares JJ., “Unconscious anxiety”: phobic responses to masked stimuli, Journal of Abnormal Psychology, 1994 May; 103(2):231-40
https://www.ncbi.nlm.nih.gov/pubmed/8040492

10 points:

We can learn much about a particular gene’s function by studying what happens when we remove that gene from the genome. It is thanks to this approach that we have been able to identify essential and non-essential genes in simple organisms like yeast and fruit flies, and then find their parallels in our human genome.

As far as essential genes go, their loss through somatic mutations results in the death of the mutant cell, or even of the entire organism, if the loss occurs at early embryonic stages. However, the genomes of most eukaryotes contain a high number of non-essential genes, any one of which may be lost without producing catastrophic effects on the cells carrying the mutation. Below, we provide a simplified description of an experiment that has been done with yeast.

During a study of yeast, two genes (let us denote them yA and yB) were found to be non-essential, because deletion of either one of them produced viable mutant yeast. However, deletion of both yA and yB was lethal to the yeast.

Question 1: Provide a possible explanation for this observation.

Genes yA and yB were found to have analogs in the human genome (let us call them hA and hB). A study of highly-proliferating tumors in a certain population of cancer patients showed that their tumors have a mutation in the hA gene that renders hA inactive.

Question 2. Can you think of how this information, together with the previous observations from experiments with yA and yB, may be used to design cancer therapy? Can you think of a reason why this approach may not work in humans? What experiments can you think of in order to check that concern?

Hint:

When two different proteins appear to be non-essential, it could mean that either they are both not involved in any vital processes, or that they are both doing something useful, and their function is …. (continue)

Answer:

When simultaneous perturbation of two genes results in the death of an organism, it is known as synthetic lethality. The basis of this concept was first described by an American geneticist Calvin Bridges in the early 20th century, who discovered that combining parents that carried specific non-lethal mutations produced non-viable progeny in fruit flies. As a rule, non-lethal mutations are possible because of genetic robustness – a capacity of genetic systems to handle abnormalities. Cells become genetically robust by keeping functionally redundant genes and proteins, which create alternative pathways to achieve the same goal with varying degrees of efficiency. If a genetically robust system loses the function of one of its genes, it has others that can either fully or partially compensate for the loss. Synthetic lethality can therefore be produced when both the main and the compensatory gene functions are perturbed. In the question above, genes yA and yB therefore act in parallel, functionally redundant pathways.

The concept of synthetic lethality has been used in chemotherapeutic strategies. Since cancer cells have acquired genetic changes that distinguish them from normal cells, finding synthetic-lethal partners of genes already perturbed in cancer can identify effective drug targets. Applying this to our question, if we know that:

1. genes yA and yB are synthetic lethal partners in yeast, and
2. genes hA and hB are human analogs of yA and yB,

we can infer that hB and hA are likely to be synthetic lethal partners in humans. Further knowing that:

3. hA is rendered inactive in a group of cancer patients,

we can hypothesize that eliminating the function of hB with drugs can induce synthetic lethality specifically in cancer cells that already have hA inactivated.

Such an approach has already been applied to specific variants of human breast and ovarian cancers that are marked by mutation of BRCA1 or 2 DNA repair genes. In the absence of BRCA1/2, cancer cells are forced to rely on another gene, PARP1, for repairing all DNA double-strand breaks. Inhibition of PARP1 with chemotherapeutic drugs therefore produces catastrophic accumulation of unrepaired DNA double-strand breaks, resulting in cancer cell death.

One of the reasons why information about synthetic lethality in model organisms such as yeast and drosophila does not always apply to human genetics and medicine is because higher-order organisms like mammals tend to have higher genetic complexity in the form of gene isoforms. Thus, where yeast may have only one gene, their human analogs can have multiple tissue-specific isoforms, which has to be taken into account when trying to make any gene into a drug target. The hB in our question is likely going to exist as hB-1, hB-2, hB-3 and hB-4. You

could still test the likelihood of successful therapeutic approach if you can demonstrate synthetic lethality in cultured human cell lines given that all of the following is true:

1. the cell line you chose expresses the specific hB isoform that is targeted by the drug being tested;
2. the hA and hB isoforms expressed in actual tumors are also the same as the ones expressed in your cell line of choice;
3. other isoforms of hA or hB are not expressed in the tumors to the degree where you can observe compensatory activity.

Nijman MBS., *Synthetic lethality: General principles, utility and detection using genetic screens in human cells*, FEBS Letters, 2011 Jan 3; 585(1): 1-6
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3018572/

Shaheen M, Allen C, Nickoloff JA, Hromas R., Synthetic lethality: exploiting the addiction of cancer to DNA repair, Blood, 2001 Jun 9; 117(23):6074-82
https://www.ncbi.nlm.nih.gov/pubmed/21441464

COMPUTER SCIENCE

- **Your program should be written in Java or Python**
- You can write and compile your code here:
  http://www.tutorialspoint.com/codingground.htm
  Please note that *codingground* site modified its structure and now all the input for the program run is entered on a separate tab. This is convenient as the same input can be used across multiple runs without re-entry
- No GUI should be used in your program: eg., easygui in Python. All problems in POM require only text input and output. GUI usage complicates solution validation, for which we are also using *codingground* site. Solutions with GUI will have points deducted or won't receive any points at all.
- Please make sure that the code compiles and runs on http://www.tutorialspoint.com/codingground.htm before submitting it.
- Any input data specified in the problem should be supplied as user input, not hard-coded into the text of the program.
- Submit the problem in a plain text file, such as .txt, .dat, etc.
  No .pdf, .doc, .docx, etc!

Introduction:

Sigma1D is a one-dimensional universe. Everything in that universe has natural length measured in sigmeters. Unizon is a trading giant that sells all sorts of goods in Sigma1D online. For efficiency Unizon standardized on 10 sigmeter long boxes. Unizon optimizes packing of sold items in boxes in order to ship the least number of boxes and save on shipping costs. In the problems below items will be represented by their length in sigmeters.

5 points:

Write a program that takes an arbitrarily long list of items (i.e. their length in sigmeters). Your program should figure out how many boxes could be completely filled, if you can put no more than 2 items in one box.

For example, given items 10, 9, 8, 7, 5, 2, 2, 1 three boxes could be completely filled: 10; 9+1; 8+2. Remaining items 7, 5 and 2 sigmeters long cannot fill any boxes completely.

Your program should print the number of full boxes and the items that go into each of them.

Hint:

It is convenient to create an array where each element *a[i]* would contain the count of items *i* in the input.

Solution:

Python-3:

```
import collections

# items = [10, 9, 8, 7, 5, 2, 2, 1]
line = input("enter items: ").rstrip("\r\n")
items = [int(x.strip()) for x in line.split(',')]

m = 10 # box length
c = collections.Counter(items) # python's multiset: { item -> count }
i = 1
for item in items:
  if item == m: # single item fit
    print("box # %d has %d" % (i, item))
    i += 1
    c[item] -= 1
  elif item < m: # may fit another item
    remainder = m - item
    c[item] -= 1
    if c[remainder] > 0:
      print("box # %d has %d and %d" % (i, item, remainder))
      i += 1
      c[remainder] -= 1
    else:
      c[item] += 1 # undo 1st item
print("end.")
```

Java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Packing5 {
  private int[] items; // = { 10, 9, 8, 7, 5, 2, 2, 1 };
  private int m = 10; // box length

  private void input() throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("enter items: ");
    String line = br.readLine().trim();
    items =
Arrays.stream(line.split(",")).map(String::trim).mapToInt(Integer::parseInt).toArray();
  }

  public void test() throws IOException {
```

```
    input();

    Map<Integer, Integer> c = new HashMap<>(); // item -> count
    Arrays.stream(items).forEach(item -> c.merge(item, 1, Integer::sum));

    int i = 1;
    for(int item : items) {
      if(item == m) { // single item fit
        System.out.printf("box # %d has %d\n", i++, item);
        c.put(item, c.get(item)-1);
      }
      else if(item < m) { // may fit another item
        int remainder = m - item;
        c.put(item, c.get(item)-1);
        if(c.containsKey(remainder) && c.get(remainder) > 0) {
          System.out.printf("box # %d has %d and %d\n", i++, item, remainder);
          c.put(remainder, c.get(remainder)-1);
        }
        else {
          c.put(item, c.get(item)+1); // undo 1 st item
        }
      }
    }
  }

  public static void main(String[] args) throws IOException {
    Packing5 test = new Packing5();
    test.test();
    System.out.println("end.");
  }
}
```

10 points:

Write a program that takes an arbitrarily long list of of items (i.e. their length in sigmeters). Your program should propose efficient packing of these items into boxes. It should output a number of rows where each row represents one box and contains a list of items.

If any of the items the program receives is too large to fit into a box, then the program should list all of them and terminate.

The program should pack the items into fewer boxes than there are items, and if such a packing is impossible, then it should indicate so by printing "Only trivial packing is possible".

For example, given items 2, 4, 7, 4, 3, 5, 2, 3, 2, 1, the following can be a solution:
2, 4, 4
7, 3
5, 2, 3
2, 1

Try to be as efficient in packing as possible. Describe the intuitive reasoning of your algorithm in the program comments.

Hint:

Sort the items from largest to smallest and start packing each box with the largest remaining item.

Solution:

Python-3:

```
# This is a so called Bin Packing Problem (https://en.wikipedia.org/wiki/Bin_packing_problem)
# There are exact/better solutions referenced above,
# e.g. Schreiber, Ethan L.; Korf, Richard E. (2013), Improved Bin Completion for Optimal Bin Packing and Number Partitioning
# but they are more complicated. Here we'll consider 2 approximate but simple solutions (and they are faster).

# items = [2, 4, 7, 4, 3, 5, 3, 1]
line = input("enter items: ").rstrip("\r\n")
items = [int(x.strip()) for x in line.split(',')]

m = 10 # box length

# check all items < m
# I separated this check for clarity
oversized = list(filter(lambda x: x > m, items))
if len(oversized) > 0:
  raise Exception("oversize items: %s" % oversized)

# (1) Scan from largest to smallest items and check if the current item can still fit into the current bin.
#     If not, then put it into the next bin.
#     This is an "on-line" algorithm.
items_sorted = sorted(items, reverse=True)
bins = []
cur_bin = 0
cur_item = 0
remainder = m
bins.append([])
while cur_item < len(items_sorted):
  if items_sorted[cur_item] > remainder:
    cur_bin += 1
    bins.append([])
    remainder = m
  bins[cur_bin].append(items_sorted[cur_item])
  remainder -= items_sorted[cur_item]
  cur_item += 1
if len(bins) == len(items):
  print("only trivial packing is possible")
print("bins: %s" % bins)

# (2) Scan from largest to smallest items and for each item, check if we can find item(s) from the right that can fit
```

```
#      into the current bin as well. The difference from (1) is that we not only checking the
item immediate to the right but
#      smaller ones, too. And by scanning towards smaller items we ensure that we picked up the
largest item(s) that can still
#     fit into the current bin alongside with the current item.
bins = []
cur_bin = -1
cur_item = 0
n = len(items_sorted)
while cur_item < n:
  cur_bin += 1
  bins.append([])
  bins[cur_bin].append(items_sorted[cur_item])
  remainder = m - items_sorted[cur_item]
  cur_item += 1
  i = cur_item
  while i < n:
    if remainder == 0:
      break
    if items_sorted[i] <= remainder:
      bins[cur_bin].append(items_sorted[i])
      remainder -= items_sorted[i]
      del(items_sorted[i])
      n -= 1
    else:
      i += 1
if len(bins) == len(items):
  print("only trivial packing is possible")
print("bins: %s" % bins)
```

Java:

```
// This is a so called Bin Packing Problem (https://en.wikipedia.org/wiki/Bin_packing_problem)
// There are exact/better solutions referenced above,
// e.g. Schreiber, Ethan L.; Korf, Richard E. (2013), Improved Bin Completion for Optimal Bin
Packing and Number Partitioning
// but they are more complicated. Here we'll consider 2 approximate but simple solutions (and
they are faster).

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class Packing10 {
  private int[] items; // = { 2, 4, 7, 4, 3, 5, 3, 1 };
  private int m = 10; // box length

  private void input() throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("enter items: ");
    String line = br.readLine().trim();
    items =
Arrays.stream(line.split(",")).map(String::trim).mapToInt(Integer::parseInt).toArray();
  }
```

```
  /**
   *  check all items < m
   *  this check is separated for clarity
   */
  private void check() throws IllegalArgumentException {
    int[] oversized = Arrays.stream(items).filter(item -> item > m).toArray();
    if(oversized.length > 0)
 throw new IllegalArgumentException(String.format("oversize items: %s",
Arrays.toString(oversized)));

  }

  /**
   * (1) Scan from largest to smallest items and check if the current item can still fit into
the current bin.
   *     If not, then put it into the next bin.
   *     This is an "on-line" algorithm.
   */
  public void test1() {
    int[] items_sorted =
IntStream.of(items).boxed().sorted(Comparator.reverseOrder()).mapToInt(i -> i).toArray();
    List<List<Integer>> bins = new ArrayList<>();
    int cur_bin = 0;
    int cur_item = 0;
    int remainder = m;
    bins.add(new ArrayList<>());
    while(cur_item < items_sorted.length) {
      if(items_sorted[cur_item] > remainder) {
        cur_bin++;
        bins.add(new ArrayList<>());
        remainder = m;
      }
      bins.get(cur_bin).add(items_sorted[cur_item]);
      remainder -= items_sorted[cur_item];
      cur_item++;
    }
    if(bins.size() == items.length)
      System.out.println("only trivial packing is possible");
    System.out.println(String.format("bins: %s", bins.toString()));
  }

  /**
   * (2) Scan from largest to smallest items and for each item, check if we can find item(s)
from the right that can fit
   *     into the current bin as well. The difference from (1) is that we not only checking the
item immediate to the right but
   *     smaller ones, too. And by scanning towards smaller items we ensure that we picked up
the largest item(s) that can still
   *     fit into the current bin alongside with the current item.
   */
  public void test2() {
    ArrayList<Integer> items_sorted = IntStream.of(items).boxed().
sorted(Comparator.reverseOrder()).mapToInt(i ->
i).boxed().collect(Collectors.toCollection(ArrayList::new));
    List<List<Integer>> bins = new ArrayList<>();
    int cur_bin = -1;
```

```
    int cur_item = 0;
    int n = items_sorted.size();
    while(cur_item < n) {
      cur_bin++;
      bins.add(new ArrayList<>());
      bins.get(cur_bin).add(items_sorted.get(cur_item));
      int remainder = m - items_sorted.get(cur_item);
      cur_item++;
      int i = cur_item;
      while(i < n) {
        if(remainder == 0)
          break;
        if(items_sorted.get(i) <= remainder) {
          bins.get(cur_bin).add(items_sorted.get(i));
          remainder -= items_sorted.get(i);
          items_sorted.remove(i);
          n--;
        }
        else
          i++;
      }
    }
    if(bins.size() == items.length)
      System.out.println("only trivial packing is possible");
    System.out.println(String.format("bins: %s", bins.toString()));
  }

  public static void main(String[] args) throws IOException {
    Packing10 test = new Packing10();
    test.input();
    test.check();

    test.test1();
    test.test2();
    System.out.println("end.");
  }
}
```