The PRIMES 2012 problem set

Dear PRIMES applicant,

This is the PRIMES 2012 problem set. Please send us your solutions as part of your PRIMES application by December 1, 2011. For complete rules, see `http://web.mit.edu/primes/apply.shtml`.

Note that this set contains four sections: "General math problems" (for all three tracks) and three sections corresponding to the three research tracks of PRIMES 2012 ("Advanced math", "Computer science", and "Computational biology"). Please solve as many problems as you can in the General math section, and also in the section(s) corresponding to the track(s) for which you are applying.

You can type the solutions or write them up by hand and then scan them. Please attach your solutions to the application as a PDF (preferred), DOC, or JPG file. The name of the attached file must start with your last name, for example, "smith-solutions." Include your full name in the heading of the file.

Note that there are separate submission instructions for the Computer science track. See these instructions in the Computer science section.

Please write not only answers, but also proofs (and partial solutions/results/ideas if you cannot completely solve the problem). Besides the admission process, your solutions will be used to decide which projects would be most suitable for you if you are accepted to PRIMES.

You are allowed to use any resources to solve these problems, *except other people's help.* This means that you can use calculators, computers, books, and the Internet. However, if you consult books or Internet sites, please give us a reference.

Note that some of these problems are tricky. We recommend that you do not leave them for the last day, and think about them, on and off, over some time (several days). We encourage you to apply if you solve at least three problems in the general math section, and at least two problems in the section of the relevant track. [1]

Enjoy!

General math problems

Problem G1. You draw 4 cards from the regular deck of 52 cards.

(a) What is the chance that all of these cards have different denominations (i.e., values)? Represent the answer as a fraction or a decimal up to the third digit.

(b) What is the chance all of these cards have different denominations, and in addition there is no neighbors (for example an 8 and a 9, a 10 and a jack, or a king and an ace are neighbors, but a 7 and a 9, or an ace and a 2 are not neighbors)?

Problem G2. Find the remainder of division of $5^{5^{555}}$ (i.e., 5 to the power 5^{555}) by 27.

Problem G3. Count geometrically different (i.e., inequivalent under rotation) colorings in red and blue of the faces of

(a) a cube;

(b) a regular octahedron.

Problem G4. One chooses at random an integer $1 \leq N < 10^{100}$ (with equal probability for all choices).

(a) What is the chance (to the third digit precision) that the leading (leftmost) digit of N^2 is 1? What is the chance that this digit is 9? Are they equal to each other?

(b) What are the exact values of these probabilities in the limit when 10^{100} is replaced by 10^k when k grows indefinitely?

Problem G5. (a) Show that the number $\sum_{n=0}^{\infty} \frac{1}{2^{n^2}}$ is irrational.

(b) Describe (as explicitly as you can) all strictly increasing sequences of nonnegative integers $b_0 < b_1 < \ldots$ for which

$$\sum_{n=0}^{\infty} \frac{1}{2^{b_n}}$$

is a rational number.

[1] We note, however, that there will be many factors in the admission decision besides your solutions of these problems.

Advanced math problems

Problem M1. (a) Find the polynomial $P(x)$ with integer coefficients and leading coefficient 1 of smallest degree, such that $P(\sqrt{2}+\sqrt{3}+\sqrt{6})=0$.

(b) Let p,q,r be three distinct primes. Find the polynomial $P(x)$ with integer coefficients and leading coefficient 1 of smallest degree, such that $P(\sqrt{p}+\sqrt{q}+\sqrt{r})=0$.

Problem M2. Let d be a positive integer. Let w be a word in letters x and y of length d (e.g., $xxyxy$ is a word of length 5). Let $a_n(w)$ be the number of words in x,y of length n which don't contain w as a subword (i.e., the number of words of length n which are not of the form awb where a,b are words).

(a) Find the generating function for $a_n(x^{d-1}y)$ (where x^m is x repeated m times). I.e., find the function given by the power series $\sum_{n=0}^{\infty} a_n t^n$, as a rational function of t.

(b) Show that $a_n(x^d)\neq a_n(x^{d-1}y)$ for some n, and compute the generating function of $a_n(x^d)$ (You may first consider the case $d=2$).

(c) Classify words w of length d for which $a_n(w)=a_n(x^{d-1}y)$ for all n (i.e. describe, as explicitly as you can, what these words are).

Hint. Try to find recursions for $a_n(w)$.

Problem M3. Let A be a matrix 100×100 whose entries are 0 or 1, each chosen randomly by flipping a coin (head=0, tail=1).

(a) What is the chance that the determinant of A is odd? (Compute up to third digit precision).

(b) Let A be an $n\times n$ matrix whose entries are determined by flipping a coin, and p_n be the probability that $\det A$ is odd. What is the limit of p_n as $n\to\infty$?

Hint. Regard the matrix as a matrix over the field of 2 elements.

Problem M4. Let $(a_n)_{n\geq 0}$ be a sequence of nonnegative real numbers such that

$$a_{n+1}\leq \frac{1}{2}(a_n+a_{n-1}). \qquad (*)$$

Show that $(a_n)_{n\geq 0}$ converges.

Hint: Show that for any $n\geq i-1$, one has $a_n\leq \max(a_i,a_{i-1})$. Set $a:=\operatorname{limsup}_{n\to\infty}a_n$, and deduce that one of any two consecutive terms of the sequence must be larger than or equal to a. Now assume that there is a subsequential limit $b<a$, with a_{n_k} converging to b, and show that the inequality (*) cannot hold for large enough k.

Problem M5. In his Care of Magical Creatures class, Hagrid showed his students magical amoebas. These creatures can inhabit cells of the first quadrant of an infinite checkerboard, labeled by (i,j), $i,j\in\mathbb{Z}_{\geq 0}$ (at most one amoeba per cell). If a magical amoeba occupies a cell (i,j) and the adjacent cells $(i+1,j)$ and $(i,j+1)$ above and to the right are empty, then it can divide, and the two daughter amoebas will inhabit the two adjacent cells (while the cell (i,j) becomes vacant). Initially, there is just one magical amoeba living at $(0,0)$. Is it possible that the amoebas will ever vacate the entire 3 by 3 square in the corner of the board (i.e., the cells with $0\leq i,j\leq 2$)?

Hint. Define a function on the set of configurations of amoebas that does not change when they divide.

Computational biology problems

Problem B1.

A bacteria in a certain population lives one or two days. On the next day after it is born, it divides with probability $p > 0$ and survives to the following day with probability $q > 0$ (otherwise it dies). On the second day, it divides with probability $r > 0$ (otherwise it dies).

(a) Find the condition on p, q, r under which the population will survive (if the initial number of bacteria is very large). In particular, determine if it will survive if:

(1) $p = q = r = 1/3$?

(2) $p = 1/3,\ q = r = 1/2$?

(b) Find the average rate of growth or decay of the population (i.e., how many times it grows or shrinks per day) as a function of p, q, r.

(c) If the population starts with 1 billion bacteria which are 1 day old, how soon, on average, will the population become extinct if $p = q = r = 1/4$?

Problem B2. PRIMES student Mary is working on a computational biology project, generating random DNA strings, 10 nucleotides long (the nucleotides A,C,G,T in each position are chosen independently and randomly, with probability 1/4 each). Being annoyed by her 12-year-old sister's loud music, Mary deletes all strings that contain a sequence GAGA in them.

(a) What percentage of the strings, on average, will Mary delete? (compute with precision 0.1%).

(b) What percentage of the strings, on average, will Mary delete, if the string consists of 20 nucleotides?

Hint. Find a recursion for the number a_N of GAGA-free strings of length N. You can also solve (a) directly, and use a computer to solve (b).

Problem B3. When DNA strands are left unattended, they want to pair up. There are four types of nucleotides: A, C, G and T. So mathematically the fragment of DNA is a string in the alphabet A, C, G, T. These nucleotides are matched to each other. When two DNA strands pair up, A on one strand always matches T and C matches G. So it is logical that if there are two complementary DNA pieces on the same fragment, they will find each other and pair up. They form a hydrogen bond. For example, a piece AACGT matches perfectly another piece TTGCA. Suppose a substring of DNA consists of a piece AACGT and somewhere later the reverse of the match: ACGTT. Such a string is called an inverted repeat. The DNA fragment we mentioned contains a string AACGT****ACGTT, where stars denote any nucleotides. Two pieces AACGT and ACGTT are complementary and not too far from each other in space. So it is easy for them to find each other and to bond to form a so-called stem-loop or a hairpin structure. For some particular loops the orientation in space becomes awkward and one of the nucleotides rips off, this might lead to a mutation and an illness.

Suppose a DNA strand consists of 100 million nucleotides. Suppose a strand is formed randomly from nucleotides and A appear with probability a, C with probability c, G with probability g, T with probability t.

(a) Find the probability that the strand contains a piece AACGTGTGGACGTT, that can form a hairpin structure as first five nucleotides can pair up with the last five. Provide the formula and calculate the answer if $a = c = g = t = 1/4$.

(b) Suppose the strand contains a piece *****GTGG*****, that can form a hairpin structure. The first five nucleotides can be any nucleotides, but they have to pair up with the last five to form a hairpin structure (this particular configuration is responsible for Sickle-cell anemia). Find the probability that the strand contains such a piece. Provide the formula and calculate the answer if $a = c = g = t = 1/4$.

Problem B4. Find the first 500 digits of Pi online and write a program to convert them to letters of the English alphabet. Please see the explanation in problems C2 and C3.

Problem B5. A test consists of 5 true or false questions. After the test (answering all 5 questions), John gets his score: the number of correct answers. John doesn't know any answer, but is allowed to take the same test several times. Can John work out a strategy that guarantees that he can figure out all the answers

(a) after the 5th attempt?

(b) after the 4th attempt?

(c) Find the smallest number of attempts needed if the test has 8 questions.

In (b) and especially (c), you may find a computer helpful.

Computer science problems: Pi Poetry

Overview.

The theme of this project is to find English words in the digits of Pi. To do this, you will:

(1) Compute the fractional digits of Pi.
(2) Convert the digits of Pi into a numeric base more suitable for word-finding — i.e., base 26.
(3) Transform the digits of Pi into an alphabetic String of letters.
(4) Find words in that particular encoding of Pi.

After the basic version of the code works, you will work on improving it to get better word coverage — changing the mapping of digits to letters so that you're more likely to find interesting words.

You are encouraged to implement this project using any programming language you want. It might be helpful to pick a language with robust mathematical libraries included (Java, Python, etc).

Problem C1: Pi Generation

This part of the problem set will generate an arbitrary number of digits of Pi, so we have data to search through. We will be implementing the Bailey-Borwein-Plouffe formula: `http://en.wikipedia.org/wiki/Bailey-Borwein-Plouffe_formula`. The particularly helpful portion of the page is subtitled "BPP digit-extraction algorithm for Pi". Because this algorithm generates arbitrary digits of Pi without generating previous digits, it is much easier to test than other Pi generation methods.

Take special note of the modular exponentiation step in the computation. Doing the naive modular exponentiation algorithm (computing $a^b \bmod m$ by multiply $a \times a \times \cdots \times a$, then taking the product modulo m) will not be fast enough to compute an interesting number of digits of Pi. You may be interested in the algorithms located here: `http://en.wikipedia.org/wiki/Modular_exponentiation`. Also take note that, in most languages, you'll encounter integer overflow problems (`http://en.wikipedia.org/wiki/Integer_overflow`) when computing large exponents. You should take special care to make sure that your solution avoids these problems.

Note that the BBP algorithm returns digits of Pi in base-16. Throughout Computer Science, base-16 is commonly referred to as hexadecimal (or hex). Hex is usually expressed using 0-9 and A-F to represent digits, with A = 10, B = 11, ..., F = 15.

You can find some digits of Pi in hexadecimal here: `http://www.super-computing.org/pi-hexa_current.html`. You can use this to check the answers you generate. (Alternately, if you find Problem C1 too difficult, you can use the digits from the link above and proceed to Problem C2. This way you can receive credit for subsequent problems that you complete.)

Problem C2: Transforming Pi

In order to find words in the digits of Pi, you will first need to convert the hex digits of Pi into something more useful.

There are many resources online that explain how to convert fractional digits from one base to another. Logically, you want to multiply the number repeatedly by the new base, and chop off the new integer part of the multiplication every time. Continue until you have as much precision as you want. For example, to convert .123456 from base-7 to base-12:

- Multiply by 12: 2.222202 Chop off integer part: 2
- Multiply by 12: 3.666333 Chop off integer part: 3
- Multiply by 12: 11.660661 Chop off integer part: 11
- Multiply by 12: 11.534535 Chop off integer part: 11 ...

So, the first few digits of .123456 base-7 expressed in base-12 are .2 3 11 11 ... (or .23BB...). Note that numbers with finite expansions expressed in one base may have infinite expansions in another. An easy way to see this is that .1 in base-3 is .33333333... in base-10. The function you write should let you choose how many digits of precision you get in the new base.

After your function is implemented, you should try converting Pi from base-16 to different bases. A quick way to check that your function is working correctly is that converting Pi from base-16 to base-10 should get you the digits of Pi that you are familiar with.

Problem C3: Converting Pi to Characters

We'll define an "alphabet" of base-N as a mapping from the digits $\{0, 1, \ldots, n-1\}$ to arbitrary strings of characters. For example, a very simple alphabet for base-26 would be $a[0] = A$, $a[1] = B$, $\ldots$, $a[25] = Z$.

You should write a function that takes in digits expressed in base-N, as well as an alphabet for base-N, and returns a String which maps every digit of Pi into the appropriate String of the alphabet, and concatenates the results together.

So, if we had digits that looked like $[0, 1, 2, 3, 4, 5, \ldots]$ in base-26, and they were fed into this function with the previously specified simple alphabet, the resulting output would be "abcdef...".

Try converting Pi from base-16 into base-26, then feed base-26 Pi into this function along with the simple alphabet we described above. You should get a $a - z$ string of characters.

Problem C4: Finding Words

Now that we have an alphanumeric string, we want to find words in it.

You should write a function that takes in a string (the haystack), as well as a list of strings (the needles). Your function should try to find each needle in the haystack, if it's present at all. It should print to the user all the needles that it found, along with the index at which each of those needles were found.

You should now be able to find words in the digits of Pi! You can go ahead and try to find your name, and other things you may be interested in. You can use the dictionary here `http://www.thefreedictionary.com/dictionary.htm` to check if a particular sequence of letters is a dictionary word.

You can either look for disjoint words that do not share any Pi digits in common, or overlapping words that share digits. Please make sure to specify in your submitted solution what method you employed.

Problem C5: Word Generation Revisited

You will likely have difficulty with finding most words in the digits of Pi. Part of the reason is that the letters we are finding are too random; "z"s occur with roughly the same frequency as "e"s. The problem is that there are not too many words in a random string of characters. To increase the number of words you can adjust the randomness to match some properties of real English. This problem will explore one way to improve your implementation.

One way you can do this is, given a source of English words, you can try to calculate how often each character occurs, and then weight those characters in appropriate proportions in your translation of Pi. Here `http://www.langmaker.com/wordlist/basiclex.htm` is a simple list of English words, though you can use any list that you want. For example, you can use the most recent essay you wrote for your English class. You should figure out how to read such a list into the programming language of your choice, so you can use it in your program.

For this part of the project, you can try to convert Pi into some very large base (base-200, for example). You should then write a function that produces a base-200 alphabet, and weight each character as often as they appear in the training data you built your alphabet from. So, if you built your alphabet from the list of basic English words linked above, you would expect a lot of "a"s and "e"s, but very few "q"s and "z"s. This would be reflected in the base-200 alphabet by maybe having 12-15 digits map to "a", but only 1-2 digits map to "z".

After you generate your base-200 alphabet, you should be able to feed it back into the function from Problem C3 with Pi expressed in base-200, and you'll get a new string to search for words in. This string should be able to find many more words than the naive base-26 implementation.

Extra: Even Better Word Generation

The above implementation of a word generator is but one of many things you could do. If you want to improve your string-finding ability even more, here are some ideas:

Things like "qx" never appear in English. Build a better word generator that reflects this fact.

Things like "ing" and "ed" appear a lot in English. Build a better word generator that supports this, too.

You could build word generator using different sources of training data. For example, if you specifically are looking for names, it could be that characters are distributed differently in the space of English names as opposed to just English words. You could find a list of names, and see if the word generator you build from that is better.

Submitting your solution

You should send MIT PRIMES a zip file containing the code that you write, with a detailed README.txt file explaining the environment you used to develop the code, and how to compile and run the code with example input files (if required) and example output files. We will look at your code to evaluate your programming style, so please include comments and make your code readable. If you have different versions of your program (for example, for Problem C5 or the Extra Work), please include a description for each version of the program in the README.txt file.

In addition, you should submit in a text file named SOLUTIONS.txt (included in the zip folder) the following:

- The number of digits you generated in Problem C1.
- The first 20 digits of Pi in base 7.
- The first 20 letters of Pi when converted to the English alphabet.
- The first 5 words you found, specifying whether the words are disjoint or overlapping.
- The character frequency table you used to find the words in Item 4.