MATH182 HOMEWORK #3
DUE July 12, 2020

Note: while you are encouraged to work together on these problems with your classmates, your final work should be written in your own words and not copied verbatim from somewhere else. You need to do at least seven (7) of these problems. All problems will be graded, although the score for the homework will be capped at $N :=$ (point value of one problem) $\times 7$ and the homework will be counted out of N total points. Thus doing more problems can only help your homework score. For the programming exercise you should submit the final answer (a number) *and* your program source code.

Exercise 1. *Obtain asymptotically tight bounds on* $\lg(n!)$ *without using Stirling's approximation. Instead, find a way to approximate the summation* $\sum_{k=1}^{n} \lg k$ *from above and below.*

Exercise 2. *Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem. Start at the left end of the array, and progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray of* $A[1\,..\,j]$*, extend the answer to find a maximum subarray ending at an index* $j+1$ *by using the following observation: a maximum subarray of* $A[1\,..\,j+1]$ *is either a maximum subarray of* $A[1\,..\,j]$ *or a subarray* $A[i\,..\,j{+}1]$*, for some* $1 \le i \le j{+}1$*. Determine a maximum subarray of the form* $A[i\,..\,j{+}1]$ *in constant time based on knowing a maximum subarray ending at index* j*.*

Your answer should consist of:

(1) Pseudocode for your algorithm.
(2) Proof of correctness.
(3) An analysis of the running time.

Exercise 3. *How quickly can you multiply a* $kn \times n$ *matrix by an* $n \times kn$ *matrix, using Strassen's algorithm as a subroutine? Answer the same question with the order of the input matrices reversed.*

Exercise 4. *Show how to multiply the complex numbers* $a+bi$ *and* $c+di$ *using only three multiplications of real numbers. The algorithm should take* a, b, c, d *as input and produce the real component* $ac-bd$ *and the imaginary component* $ad+bc$ *separately.*

Exercise 5. *Show that the solution of* $T(n) = T(n-1) + n$ *is* $O(n^2)$*.*

Exercise 6. *Solve the recurrence* $T(n) = 3T(\sqrt{n}) + \log n$ *by making a change of variables. Your solution should be asymptotically tight. Do not worry about whether values are integers.*

Exercise 7. *Use a recursion tree to determine a good asymptotic upper bound on the recurrence* $T(n) = T(n/2) + n^2$*. Use the substitution method to verify your answer.*

Exercise 8. *Use a recursion tree to determine a good asymptotic upper bound on the recurrence* $T(n) = T(n-1) + T(n/2) + n$*. Use the substitution method to verify your answer.*

Exercise 9. *Use the master method to give tight asymptotic bounds for the following recurrences:*

(1) $T(n) = 2T(n/4) + 1$
(2) $T(n) = 2T(n/4) + \sqrt{n}$
(3) $T(n) = 2T(n/4) + n$
(4) $T(n) = 2T(n/4) + n^2$

Exercise 10. *Professor Diogenes has n supposedly identical integrated-circuit chips that in principle are capable of testing each other. The professor's test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the professor cannot trust the answer of a bad chip. Thus, the four possible outcomes of a test are as follows:*

Chip A says	*Chip B says*	*Conclusion*
B is good	*A is good*	*both are good, or both are bad*
B is good	*A is bad*	*at least one is bad*
B is bad	*A is good*	*at least one is bad*
B is bad	*A is bad*	*at least one is bad*

(1) Show that if at least $n/2$ chips are bad, the professor cannot necessarily determine whip chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

(2) Consider the problem of finding a single good chip from among n chips, assuming that more than $n/2$ of the chips are good. Show that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(3) Show that the good chips can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the chips are good. Give and solve the recurrence that describes the number of tests.

Exercise 11. *Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are* equivalent *if they correspond to the same account.*

It's very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

Exercise 12. *Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a* local minimum *if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.*

You are given such a complete binary tree T, but the labeling is only specified in the following implicit *way: for each node v, you can determine the value x_v by* probing *the node v. Show how to find a local minimum of T using only $O(\log n)$* probes *to the nodes of T.*

Exercise 13 (Programming exercise)**.** *There are exactly ten ways of selecting three from five,* 12345*:*

$$123, 124, 125, 134, 135, 145, 234, 235, 245, \text{ and } 345$$

In combinatorics, we use the notation $\binom{5}{3} = 10$.

In general, $\binom{n}{r} = \frac{n!}{r!(n-r)!}$, where $r \leq n$, $n! = n \times (n-1) \times \cdots \times 3 \times 2 \times 1$, and $0! = 1$.

It is not until $n = 23$, that a value exceeds one-million: $\binom{23}{10} = 1144066$.

How many, not necessarily distinct, values of $\binom{n}{r}$, for $1 \leq n \leq 100$, are greater than one-million?

Exercise 14 (Programming exercise)**.** *Euler's Totient function,* $\varphi(n)$ *(see* `https://en.wikipedia.org/wiki/Euler%27s_totient_function`*) is used to determine the number of numbers less than* n *which are relatively prime to* n *(*d *is* ***relatively prime*** *to* n *if* $\gcd(d, n) = 1$*). For example, as* $1, 2, 4, 5, 7, 8$ *are all less than nine and relatively prime to nine,* $\varphi(9) = 6$*.*

n	*Relatively Prime*	$\varphi(n)$	$n/\varphi(n)$
2	1	1	2
3	1, 2	2	1.5
4	1, 3	2	2
5	1, 2, 3, 4	4	1.25
6	1, 5	2	3
7	1, 2, 3, 4, 5, 6	6	1.1666...
8	1, 3, 5, 7	4	2
9	1, 2, 4, 5, 7, 8	6	1.5
10	1, 3, 7, 9	4	2.5

It can be seen that $n = 6$ *produces a maximum* $n/\varphi(n)$ *for* $n \leq 10$*.*

Find the value of $n \leq 1000000$ *for which* $n/\varphi(n)$ *is a maximum.*

Exercise 15 (Programming exercise)**.** *It is possible to write five as a sum in exactly six different ways:*

$$4 + 1$$
$$3 + 2$$
$$3 + 1 + 1$$
$$2 + 2 + 1$$
$$2 + 1 + 1 + 1$$
$$1 + 1 + 1 + 1 + 1$$

How many different ways can one hundred be written as a sum of at least two positive integers?

Exercise 16 (Programming exercise)**.** *The most naive way of computing* n^{15} *requires fourteen multiplications:*

$$n \times n \times \cdots \times n \;=\; n^{15}$$

But using a "binary" method you can compute it in six multiplications:

$$\begin{aligned} n^2 &= n \times n \\ n^4 &= n^2 \times n^2 \\ n^8 &= n^4 \times n^4 \\ n^{12} &= n^8 \times n^4 \\ n^{14} &= n^{12} \times n^2 \\ n^{15} &= n^{14} \times n \end{aligned}$$

However it is yet possible to compute it in only five multiplications:

$$\begin{aligned} n^2 &= n \times n \\ n^3 &= n^2 \times n \\ n^6 &= n^3 \times n^3 \\ n^{12} &= n^6 \times n^6 \\ n^{15} &= n^{12} \times n^3 \end{aligned}$$

We shall define $m(k)$ *to be the minimum number of multiplications to compute* n^k*, for example* $m(15) = 5$*.*

For $1 \leq k \leq 200$*, find* $\sum m(k)$*.*

Exercise 17 (Programming exercise)**.** *Looking at the table below, it is easy to verify that the maximum possible sum of adjacent numbers in any direction (horizontal, vertical, diagonal, or anti-diagonal) is* $16 (= 8 + 7 + 1)$*.*

-2	5	3	2
9	-6	5	1
3	2	7	3
-1	8	-4	8

Now, let us repeat the search, but on a much larger scale:

First, generate four million pseudo-random numbers using a specific form of what is known as a "Lagged Fibonacci Generator":

- *For* $1 \leq k \leq 55$*,* $s_k = [100003 - 200003k + 300007k^3] (\mathrm{mod}\, 1000000) - 500000$*.*
- *For* $56 \leq k \leq 4000000$*,* $s_k = [s_{k-24} + s_{k-55} + 1000000] (\mathrm{mod}\, 1000000) - 500000$

Thus, $s_{10} = -393027$ *and* $s_{100} = 86613$*.*

The terms of s *are then arranged in a* 2000×2000 *table, using the first* 2000 *numbers to fill the first row (sequentially), the next* 2000 *numbers to fill the second row, and so on.*

Finally, find the greatest sum of (any number of) adjacent entries in any direction (horizontal, vertical, diagonal, or anti-diagonal).