About homework submission. You must submit a PDF file in HuskyCT. This will help our grader in the grading. I think typing a solution also helps you to think and allow you to edit more easily. If you don’t know how to typeset, you may write it up by hand and then scan into PDF. The deadline is the end of the day when the homework is due.

0. Basic Concepts.

This problem is only for self-study only; do not hand in. This problem checks your knowledge of some basic facts used in this course. Note: these do not cover all the needed background; but if you do not know how to answer some of these, it is time to review what you learned in CSE 2100 and 2500.

1. \( \ln(e^x) =? \) \( \log_{10}(1000) =? \) \( \log_2(1024) =? \)

2. \( \sum_{i=1}^{n} i =? \)

3. \( \sum_{i=1}^{n} \frac{1}{i^2} =? \)

4. What is approximately \( \sum_{i=1}^{n} \frac{1}{i} \)?

5. \( \sum_{i=1}^{n} \sum_{x=1}^{m} ix =? \)

6. We define unrooted tree as a undirected, acyclic graph. How many edges are there in an unrooted tree with \( n \) nodes?

7. How many permutations are there for \( \{1,2,3,4,5\} \)? How many (un-ordered) pairs are there from \( \{1,2,3,4,5\} \)?

8. How many ways of choosing \( k \) different elements from a set of \( n \) different elements are there? Suppose we need to consider all \( k = 0 \ldots n \), and for each \( k \) we need to consider all possible subsets with \( k \) different elements from a set of \( n \) different elements. How many subsets do we need to consider?

9. Let \( x \) be a random binary variable (i.e. \( x \) can be 0 or 1), where the probability of \( x = 1 \) is 0.6. What is the expectation of \( x \) (i.e. \( E(x) \))? Let \( x \) and \( y \) be two random variables, whose expected values are 1 and 2 respectively. What is \( E(x + y) \)?

10. Let \( S \) be a set with \( n \) elements. How many distinct subsets does \( S \) have?

1 Matrix Multiplication

Given two \( n \) by \( n \) matrices \( A \) and \( B \), write a simple algorithm to compute the product of \( A \) and \( B \). Then, analyze your algorithm to find out how many additions and multiplications your algorithm will take. Here, treat each addition or multiplication of two whole integers as one operation. You do not need to come up with the most efficient algorithm; the goal of this problem is getting you started on algorithm analysis.

2 Stable Matching: A Generalization

This problem is taken from the textbook (also see the book chapter posted on the class web page): Exercise 4 on p.23. For convenience, I re-phrase the problem as follows.

One generalization of stable matching discussed in class is to consider the situation of National Resident Matching Program, which matches medical students to hospitals. Here, there are \( m \) hospitals and \( n \) students. Each hospitals has one or more openings for residents, and each student can only work
for one hospital. We assume there are more students than the number of openings. That is, there will be students who can not find positions. Like before, each student has a ranking of (all) hospitals and each hospital has a ranking of (all) students. The problem is to find a stable assignment of students to hospitals, so that all openings are filled. We say an assignment is stable if neither of the following occurs:

1. First type of instability. There are two students, \( s, s' \), and a hospital \( h \), where \( s \) is assigned with \( h \), \( s' \) is free, and \( h \) prefers \( s' \) over \( s \).

2. Second type of instability. There are two students, \( s \) who is assigned to hospital \( h \), and \( s' \) who is assigned to \( h' \). But \( h \) prefers \( s' \) over \( s \) and \( s' \) prefers \( h \) over \( h' \).

Now show that there always exists a stable assignment by giving an efficient algorithm for this problem.

### 3 The Pancake Problem

A stack of \( n \) pancakes is placed in front of you. You have a spatula which you can insert anywhere into the stack and flip over all the pancakes above the spatula. You want to arrange the pancakes in order of their diameter (they are perfectly round), and you want to use as few flips as possible. As an example suppose \( n = 6 \), and the pancakes are numbered 1 through 6 in order of their diameter with 1 the smallest and 6 the largest. Suppose the original order is 346215, and the left end of the sequence represents the top of the stack. In one flip I can get 643215 (by flipping the first three pancakes: 346), then in the next flip 512346, then 432156, then 123456, so four flips are enough in this case. Let \( F(n) \) be the worst case number of flips needed to arrange a stack of \( n \) pancakes. Find an efficient (in a worst case sense) algorithm for this problem, where efficiency is measured by the worst case number of flips. Remember that your algorithm should work for pancakes with any order. To start you off you should easily be able to show that \( F(n) \) is at most 2\( n \).

### 4 A special sorting problem

This problem is somewhat related to the previous problem. You are given a list of numbers, which is a permutation of the integers 1, 2, 3, \ldots, \( n \). You want to sort them into the identity permutation (i.e. 1 being the first and \( n \) being the last). You are only allowed one type of operation: reversal. Here, you can pick any segment of the numbers and reverse them. For example, suppose you are given the following (ordered) list: 2, 3, 1, 4, 6, 5. You may reverse the portion e.g. from the second position to the fourth position (i.e. 3, 1, 4); after the reversal, you will obtain 2, 4, 1, 3, 6, 5. The objective is using the smallest number of reversals to sort the list into the identity permutation.

It is very hard computationally to figure out the exact minimum. So we now ask a simpler question: can you give an estimate on the minimum number of reversals (i.e. a lower bound) that are necessary for sorting a list of integers? A lower bound does not have to be the exact minimum but it must be valid: if I say a lower bound on the number of reversals is five for a list, then there exists no way of sorting the list using fewer than five reversals. Note that I don’t know whether one can sort with exactly five reversals; this is fine because five is a lower bound and I am not claiming five is the true minimum. The problem asks for some non-trivial lower bound. One can say zero is a lower bound (and it is indeed) but it is a trivial lower bound.

Now let me walk you through one approach for obtaining a non-trivial lower bound.

**Breakpoint** We say there is a breakpoint between two neighboring numbers in the list if these two numbers are not adjacent in values (i.e. either \( x \) and \( x + 1 \) or \( x + 1 \) and \( x \) for some \( x \)). For example, in the above example, there is no breakpoint between 2 and 3 but there is a breakpoint between 3 and 1.

**Question 1:** Now your task is obtaining a lower bound based on the number of breakpoints in the list with \( n \) numbers. Note that your lower bound should work for arbitrary list of the \( n \) numbers (i.e. the permutation of the integers from 1 to \( n \)). You need to justify why your lower bound is valid.

**Question 2:** Apply your lower bound on the list given as an example above: 2, 3, 1, 4, 6, 5. Then try to sort this list by hand: how many reversals do you need for this list in your best trial? Do you think your lower bound is tight?