BME 4800 and CSE 3800/5800 — Bioinformatics — Yufeng Wu — Fall 2017
Homework 1 ——- Due: 09/12 Tuesday by the end of day

---

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. I would recommend you to learn some typesetting software tools (such as Latex). Google it: Latex is the most popular typesetting tool in math and computer science.
The deadline is the end of the day when the homework is due.

This assignment covers some fundamental concepts and techniques that are important to know in order to understand the course materials. Due to the time constraints, I won't be able to teach these basic concepts. I expect most of you already know these. But just in case, please go through the following list of problems. If you have difficulty in some of these, then this is an indication you may have some difficulty in understanding some course topics. If this happens, I suggest you make up these deficiency on your own. 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 spend some time to address this.

**Note**: for this assignment *only*, **only submit the problems marked by ***. **CSE 5800 students**: you should also submit your solutions for the problem marked as G. **Extra credits**: sometimes I will assign extra credit problems. No solutions will be posted for these extra credit problems.

# 1 Basic algorithm design and analysis.

We will learn many bioinformatics algorithms in this course. Thus you need to have basic understanding of algorithm design and analysis. If you have troubles with the following questions, it is time to pick up a standard algorithm book such as "Introduction to Algorithms" by Cormen, et al. or "Algorithm Design" by Kleinberg and Tardos.

## 1.1 Asymptotic notations and basic algorithm analysis

### 1.1.1

Finding the smallest value within a $n$ by $n$ square matrix will take $\Theta(\underline{\qquad\qquad})$ time.

### 1.1.2

Is $2n^2 = O(n^2)$? Justify your answer: if your answer is yes, prove it; if your answer is no, explain why not.

### 1.1.3 * Basic algorithm analysis

Analyze the running time $T(n)$ for the following algorithm. Here, $n$ is an integer.

**while** $n > 1$ **do**
  $n \leftarrow \lfloor n/2 \rfloor$
**end while**

## 1.2 Algorithm design

### 1.2.1 * Divide and conquer

You are given a rooted tree $T$ with $n$ nodes. The tree may not be binary. You want to find the depth of $T$. Here, we define the depth of a node $v$ in the tree to be the maximum number of nodes along a path from $v$ to a leaf under $v$ (going downwards). The depth of $T$ is the depth of the root of $T$. Now give a divide and conquer algorithm for finding the depth of a tree. You need to analyze its running time.

### 1.3 Dynamic programming

We consider the longest increasing subsequence (LIS) problem. Given a list of $n$ numbers (i.e. sequence, denoted as $A$), we define subsequence as a sub-list of the numbers (with the original order). For example, let the sequence being 4; 2; 7; 5; 3; 5; 9; 6. A subsequence is 2, 5, 4, 6. We call a subsequence an increasing subsequence if this subsequence consists of numbers in the increasing order. Another example of increasing subsequence of the above example is 4, 7, 9. The LIS problem is to find the longest increasing subsequence (LIS). We first study how to compute the length of LIS. In this example, the length of LIS is 4.

Let us define $L(i)$ as the length of a LIS that *ends* at position $i$ (position means index of the array). Now design a dynamic programming algorithm based on this definition.

## 2 Probability

We will use probability extensively in this course. I expect you know basics of probabiity. The following problems are meant to test your knowledge in probability.

### 2.1 Basics

1. 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. $E(x)$)?

2. Let $x$ and $y$ be two random variables, whose expected values are 1 and 2 respectively. What is $E(x + y)$?

3. * We draw randomly five cards from the standard deck of 52 playing cards. What is the probability of drawing four aces among these five cards [1]?

4. * This question concerns conditional probability. There are three prisoners, $A$, $B$ and $C$, who are on the death row. The governor decided to pardon one of the three and picked randomly one of the three. The warden learned from the governor but decided to keep the choice secrete. Now $A$ heard of the news and came to see the warden. When told by the warden that who is to be pardoned won't be released for a couple of days, $A$ asked the following question: which of $B$ and $C$ will be executed? The warden thought for a while and told $A$ that $B$ will be executed. Here, the warden thought he didn't reveal any additional : since only one is pardoned, one of $B$ and $C$ will be executed; thus revealing $B$ is to be executed doesn't give $A$ information whether $A$ is pardoned or not. However, $A$ thinks differently: since $B$ is to be executed, then $A$ has 50% chance getting pardoned since there are only two choices $A$ and $C$.
   Now is $A$ correct? Use conditional probability to justify your answer.

5. * We have two biased coins $X_1$ and $X_2$ with probabilities of turning out heads as $p_1$ and $p_2$ respectively. Here $p_1 < p_2$. If we toss the two coins independently, due to chance, we can see $X_1$ to be head while $X_2$ to be tail. It is a little surprising that we can find a way of generating tosses where $X_2$ always succeeds (i.e. heads) when $X_1$ succeeds. To do this, we use a third biased coin $X_3$ with success probability $p_3$. Then we toss $X_1$ and $X_3$ independently. We let $X_2$ be head if either $X_1$ or $X_3$ is head. What should $p_3$ be to make this scheme work?

### 2.2 G: for graduate students only

You are playing a two-game match with an AI board game player. Here are the rules.

1. Each game has three outcomes: (human) win, lose, or draw. A win earns you 1 point, a tie earns you 0.5 point and a lose earns nothing.

[1] There are many probability calculations that can be done. For example, you can think about calculating the probability of having exactly one pair among the five cards.

2. If there is a tie in the first two games, a sudden-death game starts until you win or lose.

The AI player is very strong and your chance of a win is always less than 50%. We assume AI doesn't change the game playing style based on the current winning or losing situation. You have two strategies:

Timid You get a draw with probability $p_d$ and lose with probability $1 - p_d$. That is, you won't win using the timid strategy.

Bold Using this strategy, you win with probability $p_w$ and lose with probability $1-p_w$. Here $p_w < 0.5$.

You have different choices about when to use which strategy. Here we consider one intuitive playing strategy: play timid if and only if you are ahead in the score (note: this is strictly ahead; tie in scoring doesn't count).

Now, compute the probability of winning the match using the above strategy. You need to first give a general formula with $p_d$ and $p_w$. Then to see what your probability means, show the probability of winning when $p_d = 0.8$ and $p_w = 0.45$.

### 2.3 5% Extra Credits.

Consider the following problem. You are given one binary string $S$ (which has $n$ 0 or 1 at each position). $S$ is created by two (fixed but unknown) parental binary strings $S_1$ and $S_2$ (both of length $n$) in the following way. At position $i$ of $S$, a random parental string is chosen and the value of the parental string at position $i$ is the value of $S$ at position $i$. The only thing we know about $S_1$ and $S_2$ is that they are formed by mixing up from two ancestral pools $A$ and $B$ in the following way. At position $i$, ancestral pools $A$ and $B$ have fractions $f_A^i$ and $f_B^i$ of being of value 1. That is, the composition of the ancestral pools can vary at different positions. Here we assume $f_A^i$ and $f_B^i$ are known for each $i$. Then, $S_1$ (respectively $S_2$) are formed by first picking the ancestral pool $A$ with probability $m_1$ (respectively $m_2$) and picking $B$ with probability $1 - m_1$ (respectively $1 - m_2$); then for every position, we randomly draw a value from the chosen ancestral pool at that position. The problem we are interested is, given $S$, we want to estimate $m_1$ and $m_2$. One usual way of performing statistical inference is finding the values of $m_1$ and $m_2$ that maximize the conditional probability $p(S|m_1, m_2)$. This is called maximum likelihood estimate (MLE). Now compute $p(S|m_1, m_2)$ using the notations given above. Can you estimate $m_1$ and $m_2$ using the MLE approach? Justify your answer.

## 3 Programming practice

* Note: each student should do this assignment.

Suppose you have \$$n$ (say $n = 10$). You enter a casino and you want to play the slot machine to increase your asset to \$$N$ (say $N = 30$). Assume you can only bet any multiple of \$1 (up to the amount of dolloars you have), and the winning chance of the slot machine is $p$; if you win for a bet of \$$k$, you gain \$$k$ and otherwise you lose the \$$k$ of your bet. The play stops when you achieve the goal of reaching \$$N$ balance or when you have no money left. You have the following two strategies to choose from:

Timid strategy. Each time you just bet \$1 no matter what.

Bold strategy. Each time you bet the maximum that can get you to achieve your goal of reaching $N$. That is, if you have \$$n$ and $2n \leq N$, you would bet \$$n$; otherwise bet \$$(N - n)$.

Now write a program to test out which of the two strategies is better for the following cases: (i) $p = 0.45$, (ii) $p = 0.48$, (iii) $p = 0.5$, (iv) $p = 0.52$ and (v) $p = 0.55$. For each case, repeat 100 times and find out which strategy is better.

Submit your code along with the results you get. You should write down a conclusion on the choice of strategy. You can use any programming language you woud like to use.