Assignment 1

CS 577 *Discrete Review, Graphs & Asymptotic Analysis* Fall 2024

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Related Readings: `http://pages.cs.wisc.edu/~hasti/cs240/readings/`

Name: ______________________ Wisc id: ______________________

- **Purpose of Homework:**
  - Algorithm design and analysis, like any skill, can only be developed through consistent practice and feedback. Whether it's cooking, playing basketball, integration, gardening, interviewing, or teaching, theoretical knowledge alone is not sufficient. The comfortable feeling of "Oh, sure, I get it" after following a well-presented lecture or hearing a TA explain a homework solution is a seductive, yet dangerous trap. True understanding comes from doing the thing—by actually solving the problems yourself.
  - The homework assignments are your opportunity to practice. Lectures, textbooks, office hours, labs, and guided problem sets are designed to build intuition and provide justification for the skills we want you to develop. However, the most effective way to develop those skills is by attempting to solve the problems on your own. The process is far more important than the final solution.
  - Expect to get stuck. It's normal to have no idea where to start on some problems. That's why you have access to a textbook, lecture slides, and discussions. The journey of wrestling with the problem is an essential part of the learning process.

Scoring Guidelines

1a) 1 point	1b) 1 point	1c) 1 point	1d) 1 point
2a) 2 points	2b) 2 points		
3a) 4 points	3b) 4 points	3c) 4 points	
4) 5 points			
5) 5 points			
6) 5 points			
7) 5 points			
8) 10 points			
9) 2 points			
10a) 1 point	10b) 1 point	10c) 1 point	10d) 2 points
11a) 2 points	11b) 2 points		
12) 10 points			
13a) 5 points	13b) 5 points	13c) 10 points	13d) 10 points
14a) 2 points	14b) 2 points		
15a) 2 points	15b) 3 points	15c) 5 points	
16a) 5 points	16b) 5 points		

- **Total score:** 150 points. To get the full score, it is sufficient to score 100 points; the remaining points are for bonus and fun.
- Your score is computed for the assignment as:

$$\text{Score} = \frac{\min\{100, \text{Final score}\}}{100}$$

Homework Guidelines

- **Collaboration and Academic Integrity:**
  - You are encouraged to work together on homework problems, but you must list everyone you worked with for each problem.
  - You must write everything in your own words and properly cite every external source you use, including ideas from other students. The only sources that you are not required to cite are the official course materials (lectures, notes, homework solutions).
  - Plagiarism is strictly prohibited. Using ideas from other sources or people without citation is considered plagiarism. Copying verbatim from any source, even with citation or permission, is also considered plagiarism. Don't cheat.
- **Submission Instructions:**
  - Submit your homework solutions as PDF files on Gradescope. Submit one PDF file per numbered homework problem.
  - Gradescope will not accept other text file formats such as plain text, HTML, LaTeX source, or Microsoft Word (.doc or .docx).
  - Homework submitted as images (.png or .jpg) will not be graded.
  - Each submitted PDF file should include the following information prominently at the top of the first page: [your full name]_[course title]_[homework assignment number].pdf
- **Solution Writing:**
  - When writing an algorithm, a clear description in English is sufficient. Pseudo-code is not required.
  - Ensure that your algorithm is correct by providing a justification, and analyze the asymptotic running time of your solution. Even if your algorithm does not meet the requested time bounds, you may receive partial credit for a correct, albeit inefficient, solution.
  - Pay close attention to the instructions for each problem. Partial credit may be awarded for incomplete or partially correct answers.

Logic

1. Using a truth table, show the equivalence of the following statements.
   (a) $P \lor (\neg P \land Q) \equiv P \lor Q$
   (b) $\neg P \lor \neg Q \equiv \neg(P \land Q)$
   (c) $\neg P \lor P \equiv \text{true}$
   (d) $P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R)$

Sets

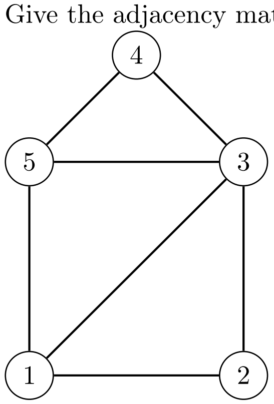
2. Based on the definitions of the sets A and B, calculate the following: $|A|$, $|B|$, $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$.
   (a) $A = \{1, 2, 6, 10\}$ and $B = \{2, 4, 9, 10\}$
   (b) $A = \{x \mid x \in \mathbb{N}\}$ and $B = \{x \in \mathbb{N} \mid x \text{ is even}\}$

Induction

3. Prove the following by induction.
   (a) $\sum_{i=1}^{n} i = n(n+1)/2$
   (b) $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$
   (c) $\sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$

Graphs and Trees

4. Give the adjacency matrix, adjacency list, edge list, and incidence matrix for the following graph.



5. How many edges are there is a complete graph of size n?
   Prove by induction that there are $h(n) = n(n-1)/2$ edges.

6. Draw all possible (unlabelled) trees with 4 nodes.

7. Show by induction that, for all trees, $|E| = |V| - 1$.

8. *Kleinberg, Jon. Algorithm Design (p. 108, q. 7).* Some friends of yours work on wireless networks, and they're currently studying the properties of a network of n mobile devices. As the devices move around, they define a graph at any point in time as follows:

   > There is a node representing each of the n devices, and there is an edge between device i and device j if the physical locations of i and j are no more than 500 meters apart. (If so, we say that i and j are "in range" of each other.)

   They'd like it to be the case that the network of devices is connected at all times, and so they've constrained the motion of the devices to satisfy the following property: at all times, each device i is within 500 meters of at least $\frac{n}{2}$ of the other devices. (We'll assume n is an even number.) What they'd like to know is: Does this property by itself guarantee that the network will remain connected?

   Here's a concrete way to formulate the question as a claim about graphs:

   Claim: Let G be a graph on n nodes, where n is an even number. If every node of G has degree at least $\frac{n}{2}$, then G is connected.

   Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

Counting

Justify your answers.

9. How many 3 digit pin codes are there?

10. A standard deck of 52 cards has 4 suits, and each suit has card number 1 (ace) to 10, a jack, a queen, and a king. A standard poker hand has 5 cards. For the following, how many ways can the described hand be drawn from a standard deck.
    (a) A royal flush: all 5 cards have the same suit and are 10, jack, queen, king, ace.
    (b) A straight flush: all 5 cards have the same suit and are in sequence, but not a royal flush.
    (c) A flush: all 5 cards have the same suit, but not a royal or straight flush.
    (d) Only one pair (2 of the 5 cards have the same number/rank, while the remaining 3 cards all have different numbers/ranks):

Proofs

11. Show that $2x$ is even for all $x \in \mathbb{N}$.
    (a) By direct proof.
    (b) By contradiction.

Find the error

12. Explain what is wrong with the following proof by induction. We will prove that all cows have the same color. Formally, we will show that for any $n \geq 1$, any collection of n cows all have the same color:

    The base case of $n = 1$ is clearly true so assume the statement holds for some set of cows of size $k \geq 1$. Now consider a set of cows of size $k+1$: $C_1, \ldots, C_{k+1}$. We know $C_1, \ldots C_k$ have the same color by the induction hypothesis and we know that $C_2, \ldots, C_{k+1}$ also have the same color by the induction hypothesis. Thus, the entire collection of cows $C_1, \ldots, C_{k+1}$ all have the same color. Therefore by induction, any collection of $n \geq 1$ cows all have the same color.

    Hint: Try the 2 case.

Recurrences

13. Solve the following recurrences. Show work and do not use the master theorem.
    (a) $c_0 = 1; c_n = c_{n-1} + 4$
    (b) $d_0 = 4; d_n = 3 \cdot d_{n-1}$
    (c) $T(1) = 1; T(n) = 2T(n/2) + n$ (An upper bound is sufficient.)
    (d) $f(1) = 1; f(n) = \sum_1^{n-1} (i \cdot f(i))$
    (Hint: compute $f(n+1) - f(n)$ for $n > 1$)

Asymptotic Analysis

14. *Kleinberg, Jon. Algorithm Design (p. 67, q. 3, 4).* Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.
    (a) $f_1(n) = n^{2.5}$
    $f_2(n) = \sqrt{2n}$
    $f_3(n) = n + 10$
    $f_4(n) = 10n$
    $f_5(n) = 100n$
    $f_6(n) = n^2 \log n$
    (b) $g_1(n) = 2^{\log n}$
    $g_2(n) = 2^n$
    $g_3(n) = n(\log n)$
    $g_4(n) = n^{4/3}$
    $g_5(n) = n^{\log n}$
    $g_6(n) = 2^{(2^n)}$
    $g_7(n) = 2^{(n^2)}$

15. *Kleinberg, Jon. Algorithm Design (p. 68, q. 5).* Assume you have a positive, non-decreasing function f and a positive, non-decreasing function g such that $g(n) \geq 2$ and $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.
    (a) $2^{f(n)}$ is $O(2^{g(n)})$
    (b) $f(n)^2$ is $O(g(n)^2)$
    (c) $\log_2 f(n)$ is $O(\log_2 g(n))$

Partial Sums

16. *Kleinberg, Jon. Algorithm Design (p. 68, q. 6).* You're given an array A consisting of n integers. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ — that is, the sum $A[i] + A[i + 1] + ... + A[j]$. (Whenever $i \geq j$, it doesn't matter what is output for $B[i, j]$.) Here's a simple algorithm to solve this problem.

```
for i = 1 to n
  for j = i + 1 to n
    add up array entries A[i] through A[j]
    store the result in B[i, j]
  endfor
endfor
```

(a) For some function f that you should choose, give a tight bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

(b) Although the algorithm provided is the most natural way to solve the problem, it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.