HW1: Computability

1. A NARCISSIST is a Turing Machine (TM) that on any input writes “IT IS ABOUT ME” at the beginning of the tape. Show that there is not Turing Machine that can decide if a given Turing Machine is a NARCISSIST, i.e., the language of TMs that are NARCISSISTS is undecidable.
2. Describe a TM that accepts only symmetric binary matrices (specify the input format on the TM tape; you can choose the format).
3. Show that the set of complex numbers with integer real and complex parts is countable, i.e., $\mathcal{C} = \{a + ib : a, b \in \mathbb{Z}\}$ is countable.

HW2: Divide-and-Conquer

1. Given 2n numbers so that the first n are sorted in ascending order and the last n are also sorted in ascending order, give an $O(logn)$ time algorithm to find the median of the 2n elements.

2. Suppose we run the median finding algorithm with groups of 3 elements. What would the worst-case run time be? What about with groups of 7 elements?

3. A matrix A is called *supersymmetric* if it can be divided into square blocks as below: $\begin{pmatrix} A_1 & A_2 \\ A_2 & A_1 \end{pmatrix}$, where A_1, A_2 are also supersymmetric, and all 1×1 matrices are supersymmetric. Give an algorithm to multiply $n \times n$ supersymmetric integer matrices A,B taking $O(n^{2.5})$ time or less.
4. Solve the following recurrences asymptotically (prove your answer, don't use an existing theorem):
   - $T(n) = 4T(n/3) + n$
   - $T(n) = T(0.2n) + T(0.75n) + \Theta(n)$
   - $T(n) = 2T(n/2) + T(n/4) + \Theta(n)$
   - $T(n) = aT(n/b) + \Theta(n^c)$ where $a > b^c$ for $a, b, c > 0$.

HW3: FFT

1. We are given an array A with n elements, where each element of A is an integer in the range $[0, n]$. We are also given an integer t. The algorithm must answer the following yes-or-no question:
   Do there exist indices i, j, k, not necessarily distinct, such that $A[i] + A[j] + A[k] = t$?
   Design an $O(n \log n)$ time algorithm for this problem.

2. Modify your algorithm to count the total number of ordered triples (i, j, k) for which the sum of entries is at most t.

HW4: Space, Time and Nondeterminism

1. Let PTIME be the class of languages that can be decided in polynomial time, i.e., for any language L in the class, there is a TM M and an integer k s.t. for every input string x of any length n, M decides whether $x \in L$ in time $O(n^k)$. Show that $SPACE(O(\log n)) \subseteq PTIME$.
2. Show that $NSPACE(n^2) \subsetneq SPACE(n^5)$, i.e., a strict subset.
3. Describe a TM D that takes as input the description of any TM M and input x for M, and simulates M on x and if M halts, it outputs the total number of steps taken in the simulation. What is the time complexity of your D as a function of the time complexity of M?

HW5: Dynamic Programming

1. Suppose you are given two strings, $S = s_1, s_2, ..., s_n, T = t_1, t_2, ..., t_m$. Give a polynomial-time algorithm to find the longest common substring.

   Explain the time complexity and argue the correctness of your algorithm. e.g. For S = zxcvb, T = abcxcvbde, the longest common substring is xcvb.

2. Suppose you are given a sequence of integers, A=$a_1, a_2, ..., a_n$.

   Give a linear-time algorithm to find the contiguous subsequence with the largest sum. e.g. For (-5, 6, 7, 1, 4, -8, 16) the contiguous subsequence with the largest sum is (6, 7, 1, 4, -8, 16) with sum 26.

3. Suppose a CS curriculum consists of n courses, all of them mandatory. Additionally, some of these courses require one or more prerequisite courses to have been taken in a previous semester. All such classes have a higher course number than their prerequisites (so there can never be a loop of classes that require each other). Assuming that a student can take any number of courses in a given semester, give an algorithm that takes the courses and prerequisites as input and computes the minimum number of semesters necessary to complete the curriculum.

4. Suppose you are given a graph $G = (V, E)$ with weights $w(v)$ for each vertex v, and that G is a tree (contains no cycles). Give an algorithm for finding the maximum weight independent set. Recall that an independent set is a set of vertices such that no edge exists between any pair of vertices in the independent set.

5. Suppose you are given a set of m vectors in n-dimensional space with integer weights, i.e. $V = \{v_1, v_2, ..., v_n\}, w(v)$. Give an algorithm to find a maximal linearly independent subset with minimum total weight. (A maximal linearly independent subset is a linearly independent set such that adding any other element will not keep it linearly independent). Assume that there is a subroutine to test linear independence which takes time T(k) for a set of k vectors. State your final complexity in terms of $m, n, T()$.

Homework 6: Ax=b

1. Solve the following linear system using Gaussian elimination. Show your work.

2. Solve the same system $\mod 3$. Show your work.
3. Given an efficient subroutine that takes integer A, b as input and finds a rational x such that $Ax = b$, give a simple, efficient algorithm to find a solution to $Ax = b \mod p$ for any prime p that does not divide $\det(A)$.

HW7: Flow

1. Given a graph $G = (V, E)$ with integer capacities C_e on its edges, and two special vertices $s, t \in V$ critical edge e of G is an edge with the property that increasing its capacity would increase the value of the maximum flow. Give an efficient algorithm to identify all critical edges of a graph.
2. Let $G = (V, E)$ be an undirected graph with capacities c_e on each edge $e \in E$. Let s, t be two of its vertices. Let $\mathbf{P}$ be the set of $s - t$ paths in G and $\mathbf{C}$ be the subsets of edges that are $s - t$ cuts. Show that $\max_{p \in \mathbf{P}} \min_{e \in p} c_e = \min_{C \in \mathbf{C}} \max_{e \in C} c_e$.

3. Given an undirected graph $G = (V, E)$ with non-negative integer capacities on edges, give an efficient algorithm to find the global min-cut, i.e., the smallest capacity set of edges whose removal disconnects the graph.
4. Let $G = (V, E)$ be a directed graph with capacities on its edges and two special vertices s, t. The capacity of a directed path from s to t is the smallest of the capacities of edges on the path. Give an efficient algorithm to find a path from s to t of maximum capacity.

HW8: Matchings

This homework is due at the beginning of class on Mon, October 29. Each solution should take up AT MOST one page (one side of one sheet of paper).

1. Let $G = (A, B, E)$ be a bipartite graph $G = (A, B, E)$ with $|A|$ not necessarily equal to $|B|$. Show that G has a matching of size $|A|$ iff for every $X \subseteq A, |N(X)| \geq |X|$.

2. Let $G = (V, E)$ be a graph in which the maximum matching is of size k. What is a polynomial-size certificate that proves that there is no matching of greater size?

3.(a) On the Greek island of Ikaria, one can go by road from any junction to any other junction; moreover each junction has an even number of roads to other junctions. Show that it is possible to start at any junction in Ikaria, travel every road exactly once and return to the starting point.

(b) In the small Chinese village of Xiao Shanghai, an intelligent postman realizes that each day he must visit every street of the village at least once. To minimize the total distance he travels, he measures the length of every street, then needs to find a tour that travels every edge at least once, starting at the post office and returning to the post office. You are an expert on matchings, and can find a mimimum-weight perfect matching in polynomial-time. Can you help the postman find a minimum length tour that visits every edge at least once, using your algorithm for minimum-weight perfect matching? (in

time polynomial in the number of edges of the village; assume that Xiao Shanghai is connected).

4. Suppose you have an algorithm to find a perfect matching of a graph if one exists. Given a graph $G = (V, E)$ and a number $r(u)$ for each vertex u, find a subset of edges F s.t. for each vertex u, exactly $r(u)$ edges of F are incident to u, or declare that such a subset of edges does not exist. For example if $r(u) = 2$ for every u, then F should have two edges incident to every vertex.

HW9: Linear Programs

1. Give a feasible solution of the following Linear Program OR show that it is infeasible.

$$x_1 + x_2 + x_3 + x_4 \geq 12$$
$$2x_1 - 6x_2 - 3x_3 + 2x_4 \geq 4$$
$$-x_1 + x_2 - x_3 - x_4 \geq 1$$
$$x_1, x_2, x_3, x_4 \geq 0$$

2. (a) Find the dual to the following standard minimum problem: Min $y_1 + 2y_2 + y_3$

s.t
$$y_1 - 2y_2 + y_3 \geq 2$$
$$-y_1 + y_2 + y_3 \geq 4$$
$$2y_1 + y_3 \geq 6$$
$$y_1 + y_2 + y_3 \geq 2$$
$$y_1, y_2, y_3 \geq 0$$

(b) show that $(y_1, y_2, y_3) = (2/3, 0, 14/3)$ is optimal for this problem and that $(x_1, x_2, x_3, x_4) = (0, 1/3, 2/3, 0)$ is optimal for the dual.

3. Show that for any convex cone generated by vectors $a_1, a_2, \ldots, a_m \in \mathbf{R}^n$, $C = \{x : x = \sum_{i=1}^{m} \alpha_i a_i, \alpha_i \geq 0\}$, any point $y \in \mathbf{R}^n$ in the space either belongs to the cone OR there is a halfspace through the origin that contains the cone and does not contain y, i.e., there is a vector $w \in \mathbf{R}^n$ s.t. $w^T a_i \geq 0$ for all $1 \leq i \leq m$ and $w^T y < 0$. (We used this fact in class to prove Farkas' lemma; don't use Farkas' lemma to prove it!)

HW10: Linear Optimization

1. A furniture plant produces 480 chairs, 400 tables and 230 cabinets every day; each of these products can be sold either natural or polished. The total number of furniture items that can be polished during a normal working day is 420; in addition, up to 250 products can be polished during overtime at a higher cost. The net profits are as follows:Find the production schedule that maximizes the total net profit, and give a proof that it is the optimal one.
2. Given a directed graph $G = (V, E)$ with a root vertex $r \in V$, an arborescence is a subset of edges that contains a directed path from the root to each vertex of the graph. Given nonnegative costs on the edges, write the problem of finding a minimum-cost arborescence as a linear program. What is a short proof that the minimum arborescence has cost at least some value C?
3. (*Bonus) Show how to solve the problem of finding a minimum-cost arborescence in a directed graph in polynomial time.

4. Let B be a box with side lengths $a_1, a_2, ..., a_n$. What is the volume of the largest ellipsoid that fits in the box and the volume of the smallest ellipsoid that contains the box?

HW11: NP-completeness I

P1. Consider the clique problem restricted to graphs in which every vertex has degree at most 3. Call this problem CLIQUE-3.
(a) Prove that CLIQUE-3 is in NP.
(b) What is wrong with the following proof of NP-completeness for CLIQUE-3?
We know that the clique problem in general graphs is NP-complete, so it is enough to present a reduction from clique-3 to clique. Given a graph G with vertices of degree at most 3, and a parameter g, the reduction leaves the graph and the parameter unchanged: clearly the output of the reduction is a possible input for the clique problem. Furthermore, the answer to both problems is identical. This proves the correctness of the reduction and, therefore, the NP-completeness of clique-3.
(c) It is true that the vertex cover problem remains NP-complete even when restricted to graphs in which every vertex has degree at most 3. Call this problem VC-3. What is wrong with the following proof of NP-completeness for CLIQUE-3?
We present a reduction from vc-3 to clique-3. Given a graph $G = (V, E)$ with node degrees bounded by 3, and a parameter b, we create an instance of clique-3 by leaving the graph unchanged and switching the parameter to $|V| - b$. Now, a subset $C \subseteq V$ is a vertex cover in G if and only if the complementary set $V \setminus C$ is a clique in G. Therefore G has a vertex cover of size $\leq b$ if and only if it has a clique of size $\geq |V| - b$. This proves the

correctness of the reduction and, consequently, the NP-completeness of clique-3.
(d) Give a polynomial time algorithm to solve CLIQUE-3.
P2. Reduce the 3-COLORING to the EXAM SCHEDULING problem.
EXAM SCHEDULING
Input: k exams $F_1, ..., F_k$; the number of possible exam time slots h; t students $s_1, ..., s_t$; and the subset of exams S_i that each student s_i is taking.
Output: A schedule of exams, mapping each exam to a time slot, such that no student is taking two exams during the same time slot.
3-COLORING
Input: A graph $G = (V, E)$.
Output: An assignment of each vertex to one of three colors, such that no edge connects two vertices of the same color.
Bonus Challenge: Try reducing EXAM SCHEDULING to 3-COLORING.
P3. A bowtie is a graph on an even number of vertices, say $2g$, in which there are 2 disjoint cliques of size g with exactly one edge between the 2 cliques.
BOWTIE
Input: An undirected graph $G = (V, E)$ and an integer goal g.
Output: YES or NO. YES means that G contains subsets S and T of V where the induced subgraph on $S \cup T$ is a bowtie with $2g$ vertices, and where S and T are disjoint, the set S is a clique on g vertices, T is also a clique on g vertices and there is exactly one edge connecting S and T; NO means there is no bowtie of $2g$ vertices in G.
Prove that BOWTIE is NP-complete.

HW12: NP-Completeness

1. Given a language $L \in NP$, show that there exists a polynomial p such that $x \stackrel{?}{\in} L$ can be decided by an algorithm with running time $O(2^{p(|x|)})$.

2. Consider the **DENSE SUBGRAPH** problem:
   Input: An undirected graph $G = (V, E)$ and integers k and l.
   Output: A subset S of vertices where S contains exactly k vertices and a total of **at least** l edges between pairs of vertices in S, or report NO if no such subset exists.Prove that the **DENSE SUBGRAPH** problem is NP-complete.
3. For each problem below, provide a polynomial-time algorithm, or show there does not exist one unless $P = NP$.

   - Given a graph $G = (V, E)$, partition V into two non-empty subsets, minimizing the number of edges between the parts.
   - Given a graph $G = (V, E)$, partition V into two non-empty subsets, maximizing the number of edges between the part.
     Hint: The following problem, namely **3-SAT'**, is NP-complete. Given a SAT formula with every clauses containing exactly 3 literals, decide whether there exist an assignment s.t. every clause has both true and false literals.