CS 466/666 Algorithm Design and Analysis **Last Updated:** July 17, 2026
Homework 4 **Due Date: Aug 5, 2026**

Please list your collaborators, including AI assistance, for *each* question. This will not affect your marks.

There are totally 60 marks, and the full mark is 50. This homework is counted 5% of the course. Please read the course outline for the late submission policy.

---

1. **Extreme Point Solution**

   (10 marks) We consider linear programming in the inequality form: $\max\langle c, x\rangle, Ax \leq b$. In L16, we gave the definitions of (1) vertex solutions (2) extreme point solutions and (3) basic solutions, and proved that (1) and (3) are equivalent. In this question, you are asked to prove that they are all equivalent.

2. **Bipartite Vertex Cover**

   (10 marks) Consider the following linear program for the minimum bipartite vertex cover problem, where $x(v)$ is a variable for a vertex v and $c(v)$ is the cost of vertex v.

$$\begin{aligned} \min \sum_{v \in V} c(v) \cdot x(v) & \\ x(u) + x(v) &\geq 1 \quad \forall uv \in E \\ x(v) &\geq 0 \end{aligned}$$

   Prove that any vertex solution to the linear program is an integral solution. You must work directly on the definition of a **vertex** solution, and cannot use other methods in the literature such as proving that the constraint matrix is total unimodular.

3. **Bicriteria Spanning Tree**

   (10 marks) In this problem we consider a generalization of the minimum spanning tree problem, in which there are two independent values on each edge, say length and cost, denoted by l_e and c_e. The goal of this problem is to find a minimum cost spanning tree with total length at most L. Consider the following linear program.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e & \\ \text{subject to} \quad & x(E(S)) \leq |S| - 1 & \forall\, S \subset V, \\ & x(E(V)) = |V| - 1, & \\ & \sum_{e \in E} l_e x_e \leq L, & \\ & x_e \geq 0 & \forall\, e \in E. \end{aligned}$$

   Design a pseudo-polynomial time algorithm to find a spanning tree with cost no more than the cost of any optimal solution, and the total length is at most $(1+\epsilon)L$ for any $\epsilon > 0$. The exponent of the running time may depend on $1/\epsilon$. (Hint: Guess and prune the "long" edges.)

4. **Balanced Coloring**

(10 marks) Given a hypergraph $G = (V, E)$ where each hyperedge $e \in E$ is a subset of V, our goal is to color the vertices of G using $\{-1, +1\}$ such that each hyperedge is as balanced as possible. Formally, given a coloring $\psi : V \to \{-1, +1\}$ on the vertices, we define $\Delta(e) = \sum_{v \in e} \psi(v)$ and $\Delta(G) = \max_{e \in E} |\Delta(e)|$. Prove that if the maximum degree of the hypergraph is d (i.e. each vertex appears in at most d hyperedges), then there is a coloring with $\Delta(G) \leq 2d - 1$.

You may find it useful to consider the following LP, where initially we set $B_e = 0$ for all $e \in E$.

$$\begin{aligned} \sum_{v \in e} x_v &= B_e \quad \forall\, e \in E \\ -1 \leq x_v &\leq 1 \quad \forall\, v \in V \end{aligned}$$

5. **Cooperative Game Theory**

(10 marks) Consider a set $N = \{1, \ldots, n\}$ of n players. Let $v : 2^N \to \mathbb{R}_+$ be a value function, i.e. for $S \subseteq N$, the value $v(S)$ is the total worth that the members of S can earn without any help from the players outside S. By default, we set $v(\emptyset) = 0$. Naturally, the total worth $v(N)$ is shared among all the players, and we would like to know how to share it so that no subset S has the incentive to deviate and obtain an outcome that is better for all of its members. Specifically, consider an allocation vector $x \in \mathbb{R}^n$ where x_i represents the payoff to player i for $1 \leq i \leq n$. We say that a subset S can improve upon an allocation if and only if $v(S) > x(S) = \sum_{i \in S} x_i$. We say that an allocation x is stable if

$$x(N) = v(N), \quad x(S) \geq v(S) \quad \forall S \subseteq N.$$

Use LP duality to give a necessary and sufficient condition for the existence of a stable allocation.

6. **Multiplicative Weight Update Method**

(10 marks) Consider the maximum flow problem from s to t on a directed graph, where each edge has capacity one. A fractional s-t flow solution with value k is an assignment of each edge e to a fractional value $x(e)$, satisfying that

$$\sum_{e \in \delta^{\text{out}}(s)} x(e) = \sum_{e \in \delta^{\text{in}}(t)} x(e) = k,$$

$$\sum_{e \in \delta^{\text{in}}(s)} x(e) = \sum_{e \in \delta^{\text{out}}(t)} x(e) = 0,$$

$$\sum_{e \in \delta^{\text{in}}(v)} x(e) = \sum_{e \in \delta^{\text{out}}(v)} x(e) \quad \forall v \in V - \{s, t\}, \text{ and}$$

$$0 \leq x(e) \leq 1 \quad \forall e \in E.$$

Use the multiplicative weights update method to solve this LP by reducing the flow problem to the problem of finding shortest paths between s and t. Analyze the convergence rate and the total complexity of your algorithm to compute a flow of value $k(1 - \epsilon)$ for any $\epsilon > 0$.