Midterm

Instructions

• This is a close-book/note exam. The exam length will be one hour and 20 minutes.

• **There are three questions. The third question is extra credit.**

• Collaboration is not permitted in this exam.

• Use of electronic devices is not permitted in this exam.

• No question will be answered by the teaching staff unless it is typo-related.

• Sign the honor code statement on the front of this cover sheet.

• **GOOD LUCK!**

Honor Code

In recognition of and in the spirit of the Honor Code, I certify that I will neither give nor receive unpermitted aid on this examination and that I will report, to the best of my ability, all Honor Code violations observed by me.

Name: ________________________________________________

Signature: ____________________________________________
1. Edge Cover

An edge cover $C$ of a graph $G(V,E)$ is a subset of $E$ such that for all $v \in V$ there exists $e \in C$ with $v \in e$. In other words, an edge cover is a set of edges covering all vertices.

a) Give an example of a graph with no edge cover.

b) Suppose $G$ does not have any isolated vertices, that is every vertex of $G$ is connected to at least one more vertex. Let $C^*$ be a minimum edge cover in $G$. Prove that

$$|C^*| + |M^*| \leq |V|$$

where $M^*$ is a maximum matching of $G$.

Solution:

a) Consider any graph with an isolated vertex (degree 0); this vertex can never be covered by any set of edges in the graph.

b) Let $S$ be the set of vertices not covered by $M^*$. Note that no edges exist between any pair of nodes in $S$ (i.e. $S$ is an independent set). Let $C$ be all edges in $M^*$ plus edges which connect $M^*$ to $S$. This is an edge cover. Then,

$$|C^*| \leq |C| = |M^*| + |S| = |M^*| + (|V| - 2|M^*|) = |V| - |M^*|$$

and rearrange.

2. A dot com Problem

Some of your friends have recently graduated and started a small company, which they are currently running out of their parents' garages in Santa Clara. They’re in the process of porting all their software from an old system to a new, revved-up system; and they’re facing the following problem.

They have a collection of $n$ software applications, $\{1,2,\ldots,n\}$, running on their old system; and they’d like to port some of these to the new system. If they move application $i$ to the new system, they expect a net (monetary) benefit of $b_i \geq 0$. The different software applications interact with one another; if applications $i$ and $j$ have extensive interactions, then the company will incur an expense if they move one of $i$ or $j$ to the new system but not both; let’s denote this expense by $x_{ij} \geq 0$.

So, if the situation were really this simple, your friends would port all $n$ applications, achieving a total benefit of $\sum_i b_i$. Unfortunately, there’s a problem....
Due to small but fundamental incompatibilities between the two systems, there’s no way to port application 1 to the new system; it will have to remain on the old system. Nevertheless, it might still pay off to port some of the other applications, accruing the associated benefit and incurring the expense of the interaction between applications on different systems.

So this is the question they pose to you: Which of the remaining applications, if any, should be moved? Give a polynomial-time algorithm to find a set \( S \subseteq \{2, 3, \ldots, n\} \) for which the sum of the benefits minus the expenses of moving the applications in \( S \) to the new system is maximized.

**Solution:**

We define a directed graph \( G = (V, E) \) with nodes \( s, v_1, v_2, \ldots, v_n \), where our sink \( t \) will correspond to \( v_1 \). Define an edge \((s, v_i)\) of capacity \( b_i \), and edges \((v_i, v_j)\) and \((v_j, v_i)\) of capacity \( x_{ij} \) for all pairs \( i \neq j \). Define \( B = \sum_i b_i \). Now, let \((S, S^C)\) be the minimum \( s-v_1 \) cut in \( G \). The capacity of \((S, S^C)\) is

\[
c(S, S^C) = \sum_{i \notin S} b_i + \sum_{i \in S, j \notin S} x_{ij} = B - \sum_{i \in S} b_i + \sum_{i \in S, j \notin S} x_{ij}
\]

Thus, finding a cut of minimum capacity is the same as finding a set \( S \) maximizing \( \sum_{i \in S} b_i - \sum_{i \in S, j \notin S} x_{ij} \).

3. Lecture Attendance Planning [Extra Credit]

A group of students want to minimize their lecture attendance by sending only one of the group to each of the \( n \) lectures. We have the following constraints:

- Each of the \( n \) lectures should be covered.
- Lecture \( i \) starts at time \( a_i \) and ends at time \( b_i \).
- It takes \( r_{ij} \) time to commute from lecture \( i \) to lecture \( j \).
- Assume all times \( r_{ij} \) as well as the duration of the lectures are in minutes and integers.

Minimize the number of students that will attend lectures i.e. develop a flow based algorithm to identify the minimum number of students needed to cover all \( n \) lectures.

**Solution:** We are going to solve this problem using a maximum matching on a bipartite graph. First observe that minimizing the number of students attending courses is equivalent to maximizing the number of classes that a given student can attend. If classes \( i \) and \( j \) are such that one can go
to class $i$, commute to class $j$ and still be on-time, then we only need one student to go to both classes. Let us now build the graph:

For each lecture $i$, set two nodes $x_i$ and $y_i$. $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ are our partitions. The edge $(x_i, y_j)$ exists if $i \neq j$ and one can go to class $i$ and then to class $j$. More formally $i \neq j$ and $a_j \geq b_i + r_{ij}$.

Building such graph takes at most $2n \times 3n^2$ steps, polynomial in $n$.

Let $M$ be a maximum matching in $G(X, Y, E)$. We claim that the minimum number of students needed is $n - |M|$. We can prove this by contradiction.

Assume you can go to all the lectures with $n - p < n - |M|$ students. Then that means that we can “reuse” $p$ students. Let $I = \{i_1, \ldots, i_p\}$ be the set of lectures where we are reusing a student (i.e. the set of lectures that at the end the student will go to another lecture). Let $J = \{j_1, \ldots, j_p\}$ be the set of lectures they are attending afterwards. It is easy to see that $M' = \{(x_{i_1}, y_{j_1}), \ldots, (x_{i_p}, y_{j_p})\}$ is a matching in $G$. But $p > |M|$, which is a contradiction.

We can find such matching using a flow algorithm, just like we saw in class.