Problem 1, CS38 Set 7, Matt Lim

Here is how we will format this problem as an integer linear program. Let C be the set cover. For this program, let $X_i = 1$ if $S_i \in C$ and 0 otherwise. Let u_k be the kth element of the universe U, and let $u_{ki} = 1$ if u_k is in S_i and 0 otherwise.

$$\textbf{maximize} \sum_{i \text{ from 1 to } n} -X_i$$

$$\textbf{such that}$$

$$\sum_{i \text{ from 1 to } n} X_i \cdot u_{ki} \geq 1 \text{ for all } 1 \leq k \leq m$$

$$X_i = 1 \text{ or } X_i = 0 \text{ for all } 1 \leq i \leq n$$

Let OPT_{ILP} be the value that optimally solving this linear program would return. Then we have that $-OPT_{ILP} = |min - set - cover|$. This is because we minimize the number of sets we choose, subject to the constraint that every element in the universe is covered by at least one of the chosen sets, which is exactly what getting a min-set-cover is.

Now we will format this problem as a linear program. Let u_k be the kth element of the universe U, and let $u_{ki} = 1$ if u_k is in S_i and 0 otherwise.

$$\textbf{maximize} \sum_{i \text{ from 1 to } n} -X_i$$

$$\textbf{such that}$$

$$\sum_{i \text{ from 1 to } n} X_i \cdot u_{ki} \geq 1 \text{ for all } 1 \leq k \leq m$$

$$0 \leq X_i \leq 1 \text{ for all } 1 \leq i \leq n$$

Let OPT_{LP} be the value that optimally solving this linear program returns. Let X^* be the optimal solution to this linear program. Then we have that $S = \{S_i : X_i^* \geq \frac{1}{k}, 1 \leq i \leq n\}$ is a set cover whose cardinality is at most k times the min-set-cover cardinality. To prove that S is a set cover, we will do the following. We have the constraint that $\sum_{i \text{ from 1 to } n} X_i \cdot u_{ki} \geq 1$ for all $1 \leq k \leq m$. This means that, for each u_j, there must be at least one set S_j that contains u_j that has a X_j value greater than or equal to $\frac{1}{k}$, since each element is contained in at most k sets (if this was not true, the constraint would not be met for each $u_j \in U$). And since we pick all those sets for every element, we cover every element in the universe. To prove that S is a set cover whose cardinality is at most k times the min-set-cover cardinality, we will do the following. We have that $-OPT_{LP} \leq -OPT_{ILP} = |min - set - cover|$, since the linear program is a relaxation of the integer linear program. We also have that $\frac{|S|}{k} \leq \sum_{i \text{ such that } S_i \in S} X_i^*$, because all such X_i^*s are greater than or equal to $\frac{1}{k}$. And since $\sum_{i \text{ such that } S_i \in S} X_i^* \leq -OPT_{LP}$ because $-OPT_{LP} = \sum_{i \text{ from 1 to } n} X_i^*$, we have the following:

$$\frac{|S|}{k} \leq -OPT_{LP} \leq -OPT_{ILP} = |min - set - cover|$$

$$\frac{|S|}{k} \leq |min - set - cover|$$

Thus we get our k-approximation.

Problem 2, CS38 Set 7, Matt Lim

Here is how we will format this problem as an integer linear program. For this program, let $x_v = 1$ if $v \in S$ and 0 otherwise.

$$\textbf{maximize} \sum_{e_{uv} \in E} e_{uv}$$

$$\textbf{such that}$$

$$0 \leq e_{uv} \leq min[x_u + x_v, 2 - (x_u + x_v)] \text{ for all } e_{uv} \in E \text{ such that each } e_{uv} \in E \text{ is a integer}$$

$$x_v = 0 \text{ or } x_v = 1 \text{ for all } v \in V$$

Note that solving this ILP gives us the value of the max cut. This is because the first constraint ensures that $e_{uv} = 1$ if $e_{uv} = (u, v)$ is across the cut and 0 otherwise, and the second constraint just ensures that a vertex is either in the cut or out of the cut. Then, with these constraints in place, we simply maximize the number of edges across the cut. So, we will let OPT_{ILP} be the value that solving this integer linear program would return, and we have that $OPT_{ILP} = |max - cut|$.

Now we will format this problem as a linear program.

$$\textbf{maximize} \sum_{e_{uv} \in E} e_{uv}$$

$$\textbf{such that}$$

$$0 \leq e_{uv} \leq min[x_u + x_v, 2 - (x_u + x_v)] \text{ for all } e_{uv} \in E$$

$$0 \leq x_v \leq 1 \text{ for all } v \in V$$

Let OPT_{LP} be the value that solving this linear program returns. Let OPT_R be the value that is obtained by rounding the linear program. We will round it using randomness, as suggested by the hint. So, for each value x_v, we will round it to 0 with 1/2 probability and round it to 1 with 1/2 probability. So basically, for each x_v, we flip a coin that determines whether or not it is 0 or 1. Then we have that $E[OPT_R]$, $E[OPT_R]$ being the expected value of OPT_R (in other words, the expected size of the cut), is given by the following, where the average of the value being summed is the probability of an edge $e = (u, v)$ being across the cut:

$$E[OPT_R] = \sum_{e_{uv} \in E} x_u(1 - x_v) + (1 - x_u)x_v = \frac{|E|}{2}$$

This is true because the average value of the term being summed is $\frac{1}{2}$ (equal probability that both terms are 0 (results in 0), both terms are 1 (results in 0), $x_u = 0$ and $x_v = 1$ (results in 1), $x_u = 0$ and $x_v = 1$ (results in 1)). Alternatively, we could get $E[OPT_R]$ by looking at the expected value of $OPT_{LP} = \sum_{e_{uv} \in E} min[x_u + x_v, 2 - (x_u + x_v)]$ with our randomized rounding. This is also just $\frac{E}{2}$, because the average value of the minimum (with our rounding) is just $\frac{1}{2}$ (it has the same four possibilities with equal probability as mentioned just above). Now we will bound the value OPT_{LP}. Note that OPT_{LP} is given as follows:

$$OPT_{LP} = \sum_{e_{uv} \in E} min[x_u + x_v, 2 - (x_u + x_v)]$$

where x_u and x_v are numbers between 0 and 1, inclusive. To bound OPT_{LP}, we will bound the term it sums:

$$B = min[x_u + x_v, 2 - (x_u + x_v)]$$

for all possible x_u and x_v. Consider all combinations of x_u and x_v such that $x_u + x_v \leq 1$. Then clearly, $B \leq 1$ for all such x_u and x_v, as the minimum will be the first term (or both terms, when the sum equals 1). Now consider all combinations of x_u and x_v such that $1 < x_u + x_v \leq 2$. Then clearly, $B \leq 1$ for all such x_u and x_v, as the minimum will be the second term. Note that $x_u + x_v$ cannot exceed 2, as both numbers are bounded above by 1. So we have that for all x_u and x_v,

$$B = min[x_u + x_v, 2 - (x_u + x_v)] \leq 1$$

Then we have the following:

$$OPT_{LP} = \sum_{e_{uv} \in E} min[x_u + x_v, 2 - (x_u + x_v)] \leq |E|$$

$$\frac{1}{2}OPT_{LP} = \frac{1}{2}\sum_{e_{uv}\in E} min[x_u + x_v, 2 - (x_u + x_v)] \leq \frac{|E|}{2}$$

This gives us the following inequality:

$$E[OPT_R] = \frac{|E|}{2} \geq \frac{1}{2}OPT_{LP}$$

Now notice that, since the linear program given above is less restrictive on the x_v values than the integer linear program (so basically, it has relaxed constraints), we have that $OPT_{LP} \geq OPT_{ILP} = |max - cut|$. Then we have that

$$E[OPT_R] \geq \frac{1}{2}OPT_{LP} \geq \frac{1}{2}OPT_{ILP} = \frac{1}{2}|max - cut|$$

$$E[OPT_R] \geq \frac{1}{2}OPT_{LP} \geq \frac{1}{2}|max - cut|$$

$$E[OPT_R] \geq \frac{1}{2}|max - cut|$$

Thus, we have formulated an LP relaxation and rounding scheme that yields an expected 2-approximation algorithm for this problem.

Problem 3, CS38 Set 7, Matt Lim

(a) For a graph $G = (V, E)$, let $m = |E|$ and $n = |V|$. We can find a maximal matching in $O(m)$ time using the following algorithm. Let the "value" of $v_1 = 1$, $v_2 = 2$, etc.

Maximal-matching$(G = (V, E))$

1. $S = \varnothing$
2. Initialize array A of size $|V|$ of all zeroes
3. **for** each edge $e = (u, v) \in E$
    4. **if** $A[u] == 0$ and $A[v] == 0$
        5. add e to S
        6. $A[u] = 1$
        7. $A[v] = 1$
8. return S

This clearly gives us a maximal matching. The reason we get a matching is line 4. This makes sure that no vertices are used in more than one edge. The reason we get a maximal matching is line 3 and line 4. These lines make it so that if we can add an edge e to S, we do it. And we try every edge. So by the end, we will have tried every edge e, and if it could have been inserted, we inserted it. So there is no edge that can be added to S that would keep it a matching (or else we would have added it). Note that this reasoning relies on the fact that, once we pass up on adding an edge, it will never be possible to add it at any further point in the algorithm, since can only add more edges to S after passing up an edge, which only leaves more vertices unavailable. In relation to our algorithm, this can be explained in the following way. Say we pass up on an edge $e = (u, v)$. This means either $A[u] == 1$ or $A[v] == 1$. Then, continuing on in the algorithm, these two values never get set back to zero. So there will never be a point when $A[u] == 0$ and $A[v] == 0$ after we pass up on e. So we can never add e once we pass up on it. Thus, after our algorithm tries every edge (after it finishes the for-loop), S, the maximal matching, cannot be enlarged by adding any edge.

This algorithm is $O(m)$ for the following reasons. First of all, the for-loop in line 3 iterates through all edges. Then, in the for-loop, everything we do is constant time, since we keep an array of which vertices we have included, and looking up things in arrays is constant time. The only other thing we do in the for-loop is add an element to a set, which is also constant time, and change two values in our array, which is also constant time. Thus our algorithm runs in time $O(m)$. Note that the array initialization is technically $O(n)$ - however, reading in the graph itself is $O(m+n)$, so we will just assume $m \geq n$ which retains the $O(m)$ time.

(b) Let M be a maximal matching, M^* a maximum matching, V the vertices in M, and V^* the vertices in M^*. We wish to prove that $|M| \geq \frac{1}{2}|M^*| \implies 2 \cdot |M| \geq |M^*|$. So, assume to the contrary that $2 \cdot |M| < |M^*|$. Notice that $2 \cdot |M| = |V|$. Also notice that each $v \in V$ can be adjacent to at most one edge in M^*, since M^* is a matching. So we have that, in total, all $|V|$ vertices in V are adjacent to at most $2 \cdot |M|$ edges in M^*. But since $|M^*| > 2 \cdot |M|$, this means there is at least one edge in M^* that is not incident to any vertex in M but is in the graph (since it is in M^*). So we can add this edge to M to enlarge it. But clearly, this is a contradiction. So we can conclude that $|M| \geq \frac{1}{2}|M^*|$ as desired.

Problem 4, CS38 Set 7, Matt Lim

Assume $a_1, a_2, \ldots, a_n$ are all less than or equal to 1 and greater than or equal to zero (otherwise we cannot pack some of them). Let OPT equal the minimum number of unit-capacity bins that can fit $a_1, a_2, \ldots, a_n$. Then we have that $sum(a_1, a_2, \ldots, a_n) \leq OPT$, since each bin will either have some empty space or be completely full. Let B be the number of bins the approximation algorithm gives us. Then we have that $B-1$ bins are greater than half full. Otherwise, we could have used at least one less bin by "combining" the contents of two bins (the greedy algorithm would have fit more stuff into one bin that was less than or equal to half full instead of using a new one and filling it less or equal to half full). This gives us the inequality $\sum_i a_i > \frac{B-1}{2}$, since $2 \cdot \sum_i a_i > B-1$, since $B-1$ bins are more than half full (so if you multiply their total contents by two, they will be overflowing). Now, we have two cases. If all B bins are greater than half full, then we get the inequality $\sum_i a_i > \frac{B}{2}$ (same logic as when $B-1$ are half full). If $B-1$ bins are greater than half full and 1 bin is less than half full, then consider the following. Let x be the fullness of B_k, the lowest capacity of the $B-1$ bins that are greater than half full. Let the other one bin (less than or equal to half full) be B_j. Then we have that $|B_j|$, which represents how full B_j is, is greater than $1-x$ (otherwise we could have put its contents in B_k). Now, consider the following. We have that $\sum_{i \text{ where } a_i \notin B_k, B_j} a_i > \frac{B-2}{2}$, since this inequality considers only bins more than half full. We also have that $\cdot \sum_{i \text{ where } a_i \in B_k, B_j} a_i > x + 1 - x = 1$. So overall, $\sum_i a_i > \frac{B-2}{2} + 1 \implies \sum_i a_i > \frac{B}{2}$. So in either case, we have $\sum_i a_i > \frac{B}{2}$. Then, we get that

$$OPT \geq \sum_i a_i > \frac{B}{2}$$

$$OPT > \frac{B}{2}$$

$$2 \cdot OPT > B$$

This gives us our approximation ratio of 2.