Math 182 HW4 Jun Ryu

Section 7

9) We design a flow network as the following: Suppose each patient is represented as x_i and each hospital as y_j and there exists an edge (x_i, y_j) iff patient i is within half-hour distance of hospital j. All of these edges will have capacity 1. Also, we connect all x_i's to a source s with edges capacity 1 each and we connect all y_j's to a sink t with edges capacity $\lceil \frac{n}{k} \rceil$. Now, we know we can send all patients to the correct hospitals iff there is an s-t flow of value n. This ensures all patients are treated and because of the hospitals being connected to a sink with capacity $\lceil \frac{n}{k} \rceil$, this prevents hospitals from being overcrowded. Now, the running time for this algorithm is poly.-time since this is a max-flow problem with $(n+k)$ nodes.

10) If e^* is not part of the max s-t flow, reducing its capacity does not change anything. Thus, we assume e^* is part of the max flow. Now, on this max flow, we reduce all capacities on edges that connect w to the sink and v to the source. This will just reduce the total flow by 1. Now, we need to determine if this is the actual max flow. The way we do this is we try to find an augmented path from s to t in the residual graph of our modified flow. If we fail to find one, we know our modified flow (1 less than the original) is maximum. If we do find one, we add that augmented path, giving us a new max flow.

18) a) We essentially add a source (s) and a sink (t) and connect s to all vertices in X and Y to v using directed edges. Now, we add edges from X to Y. Then, we set the capacity of each edge in matching to 0 and all other edges to 1.

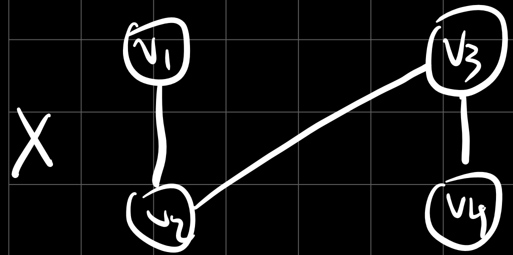
This gives us an algorithm that has a running time $O(|E|)$.

Now, we want to show that $|f| \geq |M| + K$ if there are at least K disjoint edges between X and Y, where f is the Ford-Fulkerson algorithm applied to G.

We have that these edges are also disjoint to edges in M. Thus, we can take the edges from M and create a flow of $|M| + K$, which gives us the desired result.

b) Take the following example:

$X = \{v_1, v_2\}$, $Y = \{v_3, v_4\}$



We have our original matching $\{(v_2, v_3)\} = M$

Now, $|M'| = |M| + k = 2$

so $M' = \{(v_1, v_2), (v_3, v_4)\}$

In this case $M \not\subset M'$.

24) We approach this by computing a min. cut with capacity $|C|$. Now increase the capacity of each edge in this cut by 1 and compute a new min. cut. If this min cut. has the same capacity as $|C|$, then we have this original cut C is not unique. Thus, we conclude that the graph only has a unique min. cut iff $|C| < |C_i|$, where C_i is the new modified cut as described above.

Section 8

3) We first observe that this problem is in NP since if we are given k counselors, we can check that they cover all sports in polynomial time.
We notice that this problem can be restructured into a vertex cover problem. Basically, we define a sport for each edge e and a counselor for each vertex v. Now, a counselor is qualified iff e has an endpoint equal to v. Now, if we want to find $k < m$ counselors that cover all sports, this gives us a vertex cover of size k. The converse is also true, giving us the property that the efficient recruiting algorithm $\leq_p$ Vertex Cover and vice versa. Thus, this problem is NP-complete.

5) This problem is in NP because given an arbitrary set H, we can check that it has size at most K and whether $H \cap B_i \neq \emptyset$. Now, again, we reduce this problem to a vertex cover problem. The way we do this is we define $A = \{a_1 \dots a_n\}$ to be the nodes of a vertex cover problem. For each edge $e_i = (u_i, v_i)$, we also define a set $B_i = \{u_i, v_i\}$. in our original problem. Now, we show that a vertex cover of size at most K gives us a hitting set of size at most K.

This works because if we consider a hitting set H of size at most K, we see that every set B_i is hit meaning every edge has at least 1 end in $H \rightarrow H$ is a vertex cover. Converse is also true if we just consider a cover to be a subset of A. Thus, this problem is NP-complete.

6) Again, we reduce this to a vertex cover problem. We model the problem as the following: suppose we want to find a vertex cover of size at most k. We have that each node represents a variable x_i. For each edge $e_i = (x_a, x_b)$, we create a clause $C_i = (x_a \vee x_b)$. Now, we show that these 2 problems yield the same answer. Suppose we take a vertex cover H. Since all edges are covered in H, we have that each clause has at least 1 variable set to 1. $\Rightarrow$ all clauses are satisfied.

In the converse case, it is also true because a vertex cover can be found by finding all edges that have an end equal to the value of 1. Thus, this problem is NP-complete.

7) We will essentially show that
3D Matching $\leq_p$ 4D Matching.

So take an instance of 3D Matching with sets X, Y, Z and collection C of ordered triples. Now, we extrapolate this to 4D where it has sets X, Y, Z, W of same sizes and a collection C^* of 4-tuples (w_i, x_j, y_k, z_ℓ), where $(x_j, y_k, z_\ell) \in C$. Now, we have that if $(x_j, y_k, z_\ell) \in C$, then we can easily let $(w_i, x_j, y_k, z_\ell) \in C^*$ and vice versa.

Thus, given any set of n disjoint triples in C, we can show that we can extrapolate that set to be a set of n disjoint 4-tuples in C^*. (and vice versa.) Thus, we have shown that 4D Matching is NP-complete.