HW 8 (due Wednesday, at noon, October 31, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.2**

Submission guidelines and policies as in homework 1.

1 (100 PTS.) Cry me a matching.

1.A. (20 PTS.) Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph where $|L| = |R| = n$. Furthermore, assume that there is a ***full matching*** M in G (i.e., every vertex of G appears in a matching edge of M). Prove, that for any subset $X \subseteq L$, we have that $|\mathrm{N}(X)| \geq |X|$, where

$$\mathrm{N}(X) = \{y \mid x \in X \text{ and } xy \in \mathsf{E}\}$$

is the ***neighborhood*** of X.

Solution:

Let U be the edges of M that are adjacent to the vertices of X. Since M is full, we have that $|X| = |U|$. Let Y be the right endpoints of the edges of U. We have that $|Y| = |U| = |X|$. Observe that $Y \subseteq \mathrm{N}(X)$, which implies that $|\mathrm{N}(X)| \geq |Y| \geq |X|$, as claimed.

1.B. (20 PTS.) Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph where $|L| = |R| = n$. Let M be a matching, and let $u \in L$ be a vertex that is not in M. Let R' be the set of all the vertices in R reachable via alternating paths from u, and let L' be all the vertices of L that are reachable from u via an alternating path (note that $u \in L'$). Assume that there is no augmenting path starting at u. Prove that $R' = \mathrm{N}(L')$ and $|L'| > |R'|$ (note, that $\mathrm{N}(L') \subseteq R$).

Solution:

If there is no edge adjacent to u, that the claim trivially holds since $L' = \{u\}$ and $R' = \emptyset$.

If any vertex $r \in R'$ is not adjacent to an edge of M, then there in an augmenting path from u to r, and the assumption is false. As such, it must be that all the edges in R' are adjacent to edges in the matching, which readily implies that $M^{-1}(R') = \{x \mid xy \in M \text{ and } y \in R'\}$ is of size $|R'|$. Observe that $M^{-1}(R') \subseteq L'$, and furthermore, $u \notin M$, which readily implies that $|L'| \geq |\{u\}| + |R'| > |R'|$, as claimed.

Combining the above two claims implies Hall's theorem.

Theorem 0.1 (Hall's theorem). *Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph, such that $|L| = |R|$. There is a perfect (i.e., full) matching in G $\iff$ for any set of vertices $X \subseteq L$, we have that $|\mathrm{N}(X)| \geq |X|$.*

Proof: If there is a perfect matching, then part (A) implies that $|\mathrm{N}(X)| \geq |X|$, for all X, as desired.

As for the other direction, if there is no prefect matching, then there is a vertex $u \in L$ that is not matching in the maximum cardinality matching M. The above implies that $|L'| > |R'| = |\mathrm{N}(L')|$, which is a contradiction (for $X = L'$). ■

1.C. (20 PTS.) For a number $d > 0$, a graph is ***d-uniform*** if every vertex has exactly d of edges incident to it. Let $\mathsf{G} = (L \cup R, \mathsf{E})$ Be a d-uniform graph. Prove that for every set $X \subseteq L$ we have that $|\mathrm{N}(X)| \geq |X|$.

Solution:

Let U be the set of edges adjacent to X. Observe that $|U| = d\,|X|$. Every vertex of $\mathrm{N}(X)$ can be adjacent to at most d of these edges. We conclude that $|\mathrm{N}(X)| \geq |U|/d = |X|$, as claimed.

1.D. (40 PTS.) Using the above, and only the above, prove that a bipartite d-uniform graph has a full matching.

Solution:

Consider a d-uniform bipartite graph $\mathsf{G} = (L \cup R, \mathsf{E})$. For any subset $X \subseteq L$, let E_X be the set of all edges adjacent to X in G. Clearly, there are $|X|d$ edges adjacent to these vertices (i.e., $|E_X| = |X|d$. In particular, let $Y = \Gamma(X) = \{v \mid u \in X, v \in R, uv \in \mathsf{E}\}$ be the set of all vertices of R adjacent to vertices of X. Let E_Y be the set of all edges adjacent to vertices of Y in G. We have that $E_X \subseteq E_Y$, and $d|X| = |E_X| \leq |E_Y| = d|Y|$.

Namely, $|X| \leq |\Gamma(X)|$, for any set of vertices $X \subseteq L$. Namely, the graph G complies with the condition of Hall's theorem, and contains a perfect matching.

2 (100 PTS.) Like a rolling matching.

You can use the above question in solving this question.

2.A. (30 PTS.) Prove that the edges of a k-uniform bipartite graph over n vertices (i.e., a graph where every vertex has degree k) can be assigned numbers $\{1, \ldots, k\}$, such that all adjacent edges to a vertex have different numbers. Describe an efficient algorithm, as fast as possible, for computing this numbering. What is the running time of your algorithm as a function of n and k.

Solution:

Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be the given k-uniform bipartite graph. We set $\mathsf{G}_0 = \mathsf{G}$ and $d_0 = k$.

For $i = 1, \ldots, k$, we do the following: The graph G_{i-1} is a d_{i-1} regular graph. As such, by (A), it contains a perfect matching. Let M_i be this perfect matching, and let $\mathsf{G}_i = (\mathsf{V}(\mathsf{G}_{i-1}), \mathsf{E}(\mathsf{G}_{i-1}) \setminus M_i)$. Clearly, the graph G_i is $d_i = d_{i-1} - 1$ regular, since every vertex lost exactly a single adjacent edge moving from G_{i-1} to G_i.

In the end of this process, we decomposed the original graph G into k perfect matchings $M_1, \ldots, M_k$. Now, coloring the edges of the ith matching M_i by color i results in the desired coloring.

As for the algorithm. The graph G_{k-i} have $m_i = n(k-i)/2$ edges. Running Hopcroft-Karp on such a graph takes $O(m_i\sqrt{n})$. As such, the overall running time of the algorithm is

$$\sum_{i=1}^{k} O(m_i\sqrt{n}) = O\left(\sqrt{n}\sum_{i=1}^{k} n(k-i)\right) = O\left(n^{3/2}\sum_{i=1}^{k} i\right) = O\Big(n^{3/2}k^2\Big).$$

2.B. (70 PTS.) Given a bipartite graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, the task is compute disjoint cycles that cover all the vertices of G if possible (or return that this is not possible). Put somewhat differently, we want to find a maximum subset of edges such that as many vertices as possible are adjacent to exactly two edges in this collection.

Observe that we seen in class how to solve the problem for the case that every vertex has to be adjacent to one such edge. Describe in detail how to modify the (slow) matching algorithm, seen in class, so that it works in this case. What are the augmenting paths in this case? What is the running time of your algorithm.

Prove the correctness of your algorithm.

(You can not use network flow to solve this problem.)

To clarify – the algorithm just needs to output a cover by cycles if it exists and it covers *all* vertices.

Solution:

We solve the more general problem of finding the largest k, for which there is a k-uniform subgraph that covers all the vertices of G.

The algorithm is going to try increasing values of k, till it fails to find a k-uniform graph. The largest such graph it found, is going to be output.

We modify the natural matching algorithm in the following way. We maintain a collection T of edges in the graph G, such that every vertex has at most k edges of T adjacent to it. A ***free*** vertex is adjacent to at most $k-1$ edges of T. At each iteration, we build the residual directed graph $\mathsf{H}(T)$ having all the edges of $\mathsf{E}(\mathsf{G}) \setminus T$ going from left to right, and all the edges of T going in the other direction. Compute the shortest path between two free vertices in $\mathsf{H}(T)$ (that are on different sides). This is an odd length path π – apply it to the T; that is $T \leftarrow T \oplus \pi$, and repeat till there are no free vertices.

Clearly, each such augmenting paths increases the degrees of the two free vertices, and keeps all other vertices to have the same degree.

There are two possibilities – if the algorithm fails to find an augmenting path, and there are still free vertices, then the algorithm outputs that there is no k-uniform subgraph. Otherwise, there are no free vertices, and the set of edges of T, form a k-uniform graph, as desired.

Lemma 0.2. *If G contains a k-uniform subgraph, the above algorithm would compute such a subgraph.*

Proof: Assume this is false, and the algorithm get stuck with a collection T, which is not the desired k-uniform subgraph. Specifically, every vertex of G is adjacent to at most k edges of T. Let X be the edges of the desired k-uniform subgraph.

Consider the directed graph J over the set of vertices $L \cup R$ with the edges $T \cup X$, where the edges of X are oriented from left to right, and the edges of T are oriented from right to left. Pick any vertex $v \in L \cup R$, such that there are strictly more edges of X adjacent to it than T (there must be such a vertex – otherwise, T defines the desired subgraph). Start walking in the graph J from v, deleting the edges we use, as we walk. This walk would get stack in a vertex u.

We claim that this vertex must be on the right side. Indeed, every vertex of L, has k edges of X going out of it, and at most k edges of T coming into it. As such, if the walk arrives to a vertex in L, it can always continue back to the right side. As such, since the graph is finite, the walk must get stuck in a vertex u on the right side, which is free. Indeed, if it is not free, then it is easy to argue that this walk can not get stuck in such a vertex.

Now, this walk w might not be simple, but it is easy enough to remove cycles from this walk. This leaves us with a simple path, and this path is clearly an alternating path for T, which is an augmenting path, but that is a contradiction to T being maximal ■

As for running time, the algorithm spends $O(m)$ time finding an augmenting paths. For the right value of k, the algorithm takes $O(knm)$ time. Of course, instead of starting the algorithm from scratch every time we increase k, we can just do it directly in the algorithm, hot starting it from the current set T. As such, the overall running time of the algorithm is $O(knm)$.

3 (100 PTS.) Vertex cover for half the edges.

3.A. (40 PTS.) Describe how to compute the smallest vertex cover for a given bipartite graph $\mathsf{G} = (L \cup R, \mathsf{E})$ with n vertices and m edges (hint: consider the maximum matching in this graph). What is the running time of your algorithm? Prove the correctness of your algorithm.
(Hint: Lookup Kőnig theorem and its proof.)

Solution:

Solution inspired by the Wikipedia page on Kőnig theorem (the easier solution is via network flow, but that "cheating").

Consider a maximum matching M in G. Let U_L be the set of vertices on L that are not in M. Let Z be the set of vertices that are reachable from the vertices of U_L via alternating paths. Observe that $U_L \subseteq Z$. Let

$$K = (L \setminus Z) \cup (R \cap Z)$$

Claim 0.3. *The set K is a vertex cover of the graph.*

Proof: Observe that $K \cap L = L \setminus Z \subseteq L \setminus U_L \subseteq L \cap \mathsf{V}(M)$. Consider an edge $e = uv$ adjacent to a vertex of $u \in L \setminus K = Z \cap L$. There is an alternating path π starting at a vertex of U_L arriving to u (by the definition of Z).

If $e \notin M$ then this path π can be extended to an alternating path that include e, and its right endpoint is in Z, which implies that K covers this edge.

If $e \in M$ then the last edge of the path π, must be e, which implies that its right endpoint is in Z, which implies that K covers this edge.

This implies that K is a vertex cover. ■

Claim 0.4. $|K| = |M|$.

Proof: Observe that $|K| \geq |M|$, since it must cover all the edges of M, and its a vertex cover.

Observe that all the vertices of K are directly covered by the matching M. Indeed, $K \cap L \subseteq L \setminus U_L \subseteq \mathsf{V}(M)$. If a vertex of $K \cap R = Z \cap R$ was not covered by M, then there would be an augmenting path ending at this vertex, which is a contradiction to M being maximum matching.

Consider an edge $e \in M$, If its right endpoint is in Z, then its left endpoint is in Z, which implies that only its right vertex is in K.

If the right endpoint of e is not in Z, then not both endpoints can be in K.

We conclude that every matching edge of M has only one endpoint in K, which implies the claim. ■

Clearly, there can not be a smaller vertex cover than M, which implies that K is indeed the desired minimum cover. The matching M can be computed in $O(m\sqrt{n})$ time, and K can be computed then in $O(m)$ time.

The overall running time of the algorithm is $O(\sqrt{n}m)$.

3.B. (60 PTS.) You are given a graph G (not necessarily bipartite), that has a vertex cover of size k. Describe a polynomial time algorithm that computes a set of k vertices, such that these k vertices cover at least $m/2$ edges of G, where m is the number of edges in G.

Solution:

Compute a two approximation to a max-cut of G. This can be done randomly by coloring the vertices by two colors, or by starting with an arbitrary coloring of a graph by two colors, and

greedily moving a vertex to the other side if it increases the number of edges in the cut. It is easy to see that this algorithm computes a cut with at least $m/2$ edges.

The running time of the deterministic algorithm is $O(mn)$. The randomized algorithm takes $O(m)$ time, but it might fail. (As long as the running time of your algorithm is polynomial, this does not matter, anyway.) The randomized algorithm can be modified to run in $O(m^2 \log m)$ time and succeed with high probability, by just repeatedly trying till getting a cut with at least $m/2$ edges.

Next, the bipartite subgraph represented by the approximate max-cut, has a vertex cover of size k, and it can be computed by the previous question in polynomial time (i.e., $O(m\sqrt{n})$), thus implying the claim.

There might be a much simpler way to do this.