ASSIGNMENT 6

CSCI 4113/6101
INSTRUCTOR: NORBERT ZEH
SOLUTIONS

QUESTION 1

We start with the hint in the assignment: A vertex cover C is minimal if and only if every vertex in C has an incident edge whose other endpoint is not in C.

First, we prove that any vertex cover C that satisfies this condition must be minimal. Let $C' \subset C$. Then there exists a vertex $v \in C \setminus C'$ and an edge $\{u, v\} \in G$ with $u \notin C$. Thus, $u, v \notin C'$, that is, C' does not cover the edge $\{u, v\}$. This shows that there is no proper subset of C that covers all edges of G, that is C is minimal.

Conversely, let C be a minimal vertex cover. If there exists a vertex $v \in C$ with $N(v) \subseteq C$, then $C \setminus \{v\}$ is also a vertex cover because every edge incident to v is still covered by its endpoint in $N(v)$. This contradicts the minimality of C. Thus, if C is minimal, then $N(v) \setminus C \neq \emptyset$, for all $v \in C$, that is, there exists an edge $\{u, v\}$ incident to v with $u \notin C$.

Testing whether a vertex cover is minimal is easy now. For every vertex $v \in C$, we inspect the edges incident to v, which takes $O(\deg(v))$ time. If we find an edge incident to v whose other endpoint is not in C, we move on to the next vertex in C or, if this was the last vertex, then we report that C is minimal. If we do not find such an edge incident to v, we report that C is not minimal. Since $\sum_{v \in V} \deg(v) = 2m$, this procedure to test whether a vertex cover is minimal takes $O(m)$ time.

QUESTION 2

(a) We use induction on the number of edges in G to prove this.
If G has no edges, then the algorithm returns $\mathcal{C} = \{\emptyset\}$. Since G has no edges, $\emptyset$ is a vertex cover of G, and its size is $0 \leq k$.
If G has at least one edge but $k = 0$, then the algorithm returns $\mathcal{C} = \emptyset$. In this case, it is trivially true that every set in $\mathcal{C}$ is a vertex cover of G of size at most k.
If G has at least one edge and $k > 0$, then the algorithm picks an edge $\{u, v\}$ and returns the set $\mathcal{C} = \{C \cup \{u\} \mid C \in \mathcal{C}_u\} \cup \{C \cup \{v\} \mid C \in \mathcal{C}_v\}$, where $\mathcal{C}_u$ and $\mathcal{C}_v$ are the sets returned by recursive calls on $(G - u, k - 1)$ and $(G - v, k - 1)$, respectively. Since $G - u$ and $G - v$ have fewer edges than G — at least the edge $\{u, v\}$ is missing from both graphs — the induction hypothesis shows that every set in $\mathcal{C}_u$ is a vertex cover of $G - u$ of size at most $k - 1$, and every set in $\mathcal{C}_v$ is a vertex cover of $G - v$ of size at most $k - 1$. Thus, all sets in $\mathcal{C}$ have size most k. For every vertex cover $C \in \mathcal{C}_u$, $C \cup \{u\}$ is a vertex cover of G because C covers all edges in $G - u$ and all edges in G not in $G - u$ are incident to u. By an analogous argument, $C \cup \{v\}$ is a vertex cover of G, for all $C \in \mathcal{C}_v$.

Thus, all sets in $\mathcal{C}$ are vertex covers of G of size at most k.

(b) Again, we use induction on the number of edges in G to prove this.
If G has no edges, then $\emptyset$ is a vertex cover of G. Thus, every non-empty vertex cover is not minimal, that is, $\emptyset$ is the unique minimal vertex cover of G. $\mathcal{C}$ contains this set.
If G has at least one edge and $k = 0$, then there is no vertex cover of G of size at most k. Thus, there is also no *minimal* vertex cover of G of size at most k, and $\mathcal{C} = \emptyset$ does contain all those minimal vertex cover.
If G has at least one edge and $k > 0$, then $\mathcal{C} = \{C \cup \{u\} \mid C \in \mathcal{C}_u\} \cup \{C \cup \{v\} \mid C \in \mathcal{C}_v\}$, where $\{u, v\}$ is an edge of G and $\mathcal{C}_u$ and $\mathcal{C}_v$ are the sets returned by recursive calls on $(G-u, k-1)$ and $(G-v, k-1)$, respectively. As argued in the proof of (a), $G-u$ and $G-v$ have fewer edges than G. Thus, the induction hypothesis shows that $\mathcal{C}_u$ and $\mathcal{C}_v$ contain all minimal vertex covers of $G-u$ and $G-v$, respectively, of size at most $k-1$.
Now consider a minimal vertex cover C of G of size at most k. Since C must cover the edge $\{u, v\}$, it must contain at least one of u and v. Assume it contains u. The case when it contains v is analogous. Then $C' = C \setminus \{u\}$ is a vertex cover of $G-u$ of size at most $k-1$. It must in fact be a *minimal* vertex cover of $G-u$ because, if there is a proper subset $C'' \subset C'$ that is a vertex cover of $G-u$, then $C'' \cup \{u\}$ is a vertex cover of G that is a proper subset of C, contradicting the minimality of C. Since C' is a vertex cover of $G-u$ of size at most $k-1$, it is an element of $\mathcal{C}_u$. Thus, $\mathcal{C}$ contains $C' \cup \{u\} = C$.

(c) As just argued, the recursive algorithm returns a set $\mathcal{C}$ containing all minimal vertex covers of size at most k. We start by proving that $|\mathcal{C}| \leq 2^k$. Again, we use induction on the number of edges of G.
If G has no edges, then $|\mathcal{C}| = 1 \leq 2^k$.
If G has at least one edge but $k = 0$, then $|\mathcal{C}| = 0 \leq 2^k$.
If G has at least one edge and $k > 0$, then $|\mathcal{C}| \leq |\mathcal{C}_u| + |\mathcal{C}_v|$. Once again, since $G-u$ and $G-v$ have fewer edges than G, the induction hypothesis shows that $|\mathcal{C}_u| \leq 2^{k-1}$ and $|\mathcal{C}_v| \leq 2^{k-1}$. Thus, $|\mathcal{C}| \leq 2^k$.
This bound on $\mathcal{C}$ immediately implies that we can discard all non-minimal vertex covers from $\mathcal{C}$ in $O^*\left(2^k\right)$ time because this requires inspecting each set in $\mathcal{C}$ and applying the algorithm from Question 1 to it.
It remains to prove that the construction of $\mathcal{C}$ itself takes $O^*\left(2^k\right)$ time. Specifically, we use induction on the number of edges in G to prove that the construction of $\mathcal{C}$ takes at most $T(n, m, k) = c \cdot 2^k \cdot (k+1) \cdot (n+m)$ time, for some constant $c > 0$.
If G has no edges, then we spend constant time to verify this, and then construct $\mathcal{C}$ in constant time. For c large enough, this cost is bounded by $T(n, m, k)$.
If G has at least one edge but $k = 0$, then we spend constant time to verify this, and then construct $\mathcal{C}$ in constant time. For c large enough, this cost is bounded by $T(n, m, k)$.
Finally, if G has at least one edge and $k > 0$, then we spend $O(n+m)$ time to construct $G-v$ and $G-w$. Once the recursive calls return, we need to construct $C \cup \{u\}$, for all $C \in \mathcal{C}_u$, and $C \cup \{v\}$, for all $C \in \mathcal{C}_v$. This takes $O(n)$ time per set, $O(|\mathcal{C}|n) = O\left(2^k n\right)$ time in total. Thus, the cost for this recursive call, excluding the recursive calls on $G-u$ and $G-v$, is bounded by $c \cdot 2^k \cdot (n+m)$, for c large enough. By the induction hypothesis, the resursive calls on $G-u$ and $G-v$ each take at most $T(n, m, k-1)$ time. Thus, the cost of the current invovation, including all recursive calls

it makes, is bounded by

$$c \cdot 2^k \cdot (n+m) + 2T(n,m,k-1) = c \cdot 2^k \cdot (n+m) + 2 \cdot c \cdot 2^{k-1} \cdot k \cdot (n+m) = c \cdot 2^k \cdot (k+1) \cdot (n+m) = T(n,m,k),$$

as claimed.

QUESTION 3

Let M be a maximal matching, let C be a minimal vertex cover C of G such that $C \subseteq V(M)$, let M_1 be a maximum matching of $G[C]$, and let M_2 be a maximal matching of $G[V(G) \setminus V(M_1)]$.

First, we prove that $M_1 \cup M_2$ is a maximal matching. M_1 and M_2 are both matchings. Since M_2 is a matching of $G[V(G) \setminus V(M_1)]$, no edge in M_2 shares an endpoint with an edge in M_1. Thus, $M_1 \cup M_2$ is a matching. If it is not a *maximal* matching, then there exists an edge $\{u,v\} \in G$ such that both u and v are unmatched by $M_1 \cup M_2$. Since every vertex in $V(M_1)$ is matched by M_1, we must have $u,v \in V(G) \setminus V(M_1)$. This is impossible though because M_2 is a maximal matching of $G[V(G) \setminus V(M_1)]$, that is, every edge of $G[V(G) \setminus V(M_1)]$ shares an endpoint with at least one edge in M_2. Thus, $M_1 \cup M_2$ is a maximal matching of G.

To bound the size of $M_1 \cup M_2$, note that, since C is a vertex cover of G, every edge has at least one endpoint in C. The edges in M_1 have both their endpoints in C and do not share any endpoints, as M_1 is a matching of $G[C]$. Since the edges in M_2 do not share any endpoints with edges in M_1, this leaves $|C| \setminus 2|M_1|$ vertices in C that can be endpoints of edges in M_2. Since every edge in M_2 has at least one endpoint in C and no two edges in M_2 share an endpoint, this shows that $|M_2| \leq |C| - 2|M_1|$. Thus, $|M_1 \cup M_2| = |M_1| + |M_2| \leq |C| - |M_1|$.

Now consider M. Split M into two subsets M_C and $\bar{M}_C$, where the edges in M_C have both their endpoints in C, and the edges in $\bar{M}_C$ have only one endpoint each in C. Since $C \subseteq V(M)$, every vertex in C is matched by M. Since M is a matching, this shows that $|C| = 2|M_C| + |\bar{M}_C|$, that is, $|M| = |M_C| + |\bar{M}_C| = |C| - |M_C|$.

Finally, note that M_C is a matching of $G[C]$, and M_1 is a *maximum* matching of $G[C]$. Thus, $|M_C| \leq |M_1|$ and $|M_1 \cup M_2| \leq |C| - |M_1| \leq |C| - |M_C| = |M|$.

QUESTION 4

The following algorithm decides whether there exists a maximal matching of size at most k in G: We enumerate all minimal vertex covers C of G of size at most $2k$. As shown in Question 1, this can be done in $O^*\left(2^{2k}\right) = O^*\left(4^k\right)$ time. For each such vertex cover C, we compute a maximum matching M_1 of $G[C]$ and a maximal matching M_2 of $G[V(G) \setminus V(M_1)]$. Let $M_C = M_1 \cup M_2$.[1] We remember the smallest of these matchings M_C we find. If it has size at most k, then we return it as proof that G has a maximal matching of size at most k. Otherwise, we answer that G does not have a maximal matching of size at most k.

As shown in Question 2, we find at most $2^{2k} = 4^k$ minimal vertex covers. For each, it takes polynomial time to find a maximum matching M_1 of $G[C]$ and a maximal matching of $G[V(G) \setminus V(M_1)]$. Thus, after the initial $O^*\left(4^k\right)$ cost of finding all minimal vertex covers of size at most $2k$, it takes $O^*\left(4^k\right)$ additional

[1] M_C has a different meaning in this question than it had in Question 3.

time to find the matchings M_C for all minimal vertex covers C we have computed, and to identify the smallest of them. Overall, the algorithm takes $O^*\left(4^k\right)$ time. We have to prove that it is correct.

The argument in the proof of Question 3 shows that the set M_C, for every C, is a maximal matching of G. Thus, if we find such a matching M_C of size at most k, then this does indeed show that G has a maximal matching of size at most k. We need to prove the converse: If G has a maximal matching of size at most k, then there exists a minimal vertex cover C of size at most $2k$ such that M_C is a matching of size at most k.

Question 3 provides the proof: Let M be a maximal matching of size at most k, and let $C \subseteq V(M)$ be a minimal vertex cover of G. Since $|C| \leq |V(M)| \leq 2k$, C is among the minimal vertex covers we find. The proof in Question 3 now shows that $|M_C| = |M_1 \cup M_2| \leq |M| \leq k$.

Note: *You may wonder why we insist on finding minimal vertex covers. The correctness of the answers in Questions 3 and 4 does not depend on the involved vertex covers being minimal. However, the running time of the algorithm does. Indeed, while we proved that we can enumerate all minimal vertex covers of size at most k in $O\left(2^k\right)$ time, this is no longer true if we want to enumerate all vertex covers of size at most k. Indeed, consider a graph without edges. Then any subset of at most k vertices is a vertex cover of size at most k, and there are $\binom{n}{k}$ vertex covers of size exactly k. Thus, $\binom{n}{k}$ is a lower bound on the running time of any algorithm that enumerates all vertex covers of size at most k in such a graph. A similar issue arises also in non-trivial graphs: If there exists a vertex cover C of size $k' < k$, then any superset of C of size k is also a vertex cover, and there exist $\binom{n-k'}{k-k'}$ such supersets.*