COMS 331: Theory of Computing, Spring 2023
Homework Assignment 8

Neha Maddali
Due at 10:00PM, Wednesday, April 12, on Gradescope.

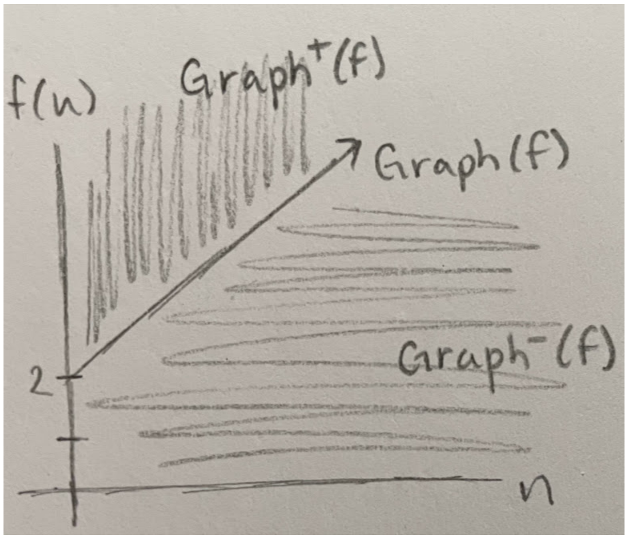
Problem 51.
To prove that $\forall B$ is co-c.e. given that B is co-c.e., we need to show that there exists a TM that can semi-decide $\forall B$. Let M be a TM that semi-decides B. We construct a new TM N that semi-decides $\forall B$ as follows:

On input x, N first runs M on all possible inputs w. If M halts and $\langle x, w\rangle \notin B$ for some w, then N rejects x. If M doesn't halt on some input w, N also doesn't halt on input x. If x is not in $\forall B$, then there exists some w such that $\langle x, w\rangle \notin B$. In this case, M will eventually halt on w and reject x, so N will also reject x. So, if N halts and rejects x, we can conclude that x is not in $\forall B$. Otherwise, if $x \in \forall B$, then for all possible inputs w, $\langle x, w\rangle \in B$. This means that M will either halt and accept x on all possible inputs w, or M will not halt on some input w. If M halts and accepts x on all possible inputs w, then N will also halt and accept x. If M doesn't halt on some input w, then N also doesn't halt on input x. Thus, N semi-decides $\forall B$ and so $\forall B$ is co-c.e. ■

Problem 52.
Let M be a TM that semi-decides A. Construct a new TM N that decides B as follows: On input x, for each i and for each string w of length i, if $\langle x, w\rangle \in A$, accept x and continue to the next string w, else reject x.

The language B decided by N consists of all strings x such that for every w, $\langle x, w\rangle \in A$. First, suppose that $x \in A$. Then, for every w, either $\langle x, w\rangle \in A$ or $\langle x, w\rangle \notin A$. If $\langle x, w\rangle \in A$ for every w, then x is in the language decided by N, because N checks all possible strings w and continues to the next one if $\langle x, w\rangle \in A$. Otherwise, there exists some w such that $\langle x, w\rangle \notin A$, so N will reject x. Now suppose that $x \notin A$. Then, there exists some w such that $\langle x, w\rangle \notin A$. Since N checks all possible strings w in lexicographic order and rejects x if $\langle x, w\rangle \notin A$ for some w, it will reject x. Thus A = $\forall B$ where B is the decidable language decided by N. ■

Problem 53.
Graph(f) is the linear line f(n)=n+2
Graph$^+$(f) is everything above the line f(n)=n+2, $\forall n \in N$
Graph$^-$(f) is everything below the line f(n)=n+2, $\forall n \in N$



Problem 54.a.
Graph(G) is not decidable. To prove this, use a reduction from the Halting problem. Assume that Graph(G) is decidable and consider a TM M takes input x. We can define a function f(n) as such: run M for n steps on input x, and output n+1 if M has not halted within n steps, and output the value it would have printed on the tape if it had halted within n steps. Then Graph(G) would be able to decide whether (n, f(n)) is in Graph(G), which is equivalent to deciding whether M halts within n steps on input x. But this contradicts the undecidability of the Halting problem we learned from lecture. This means that Graph(G) is not decidable. ■

Problem 54.b.
Graph(G) is c.e. Show that there exists a TM M that can generate all pairs (n, G(n)) for $n \in N$. M is as follows: on input k, (1) generate all pairs (n, m) for n, m $\leq k$. (2) For each such pair, check if m=G(n). (3) If m=G(n), output the pair (n,m). TM M will eventually generate all pairs (n, G(n)) for $n \in N$. Thus, Graph(G) is c.e. ■

Problem 54.c.
Graph(G) is not co-c.e. To prove this, we can show that it complement is not c.e. The complement of Graph(G) is the set of all pairs (n,k) such that $k \leq G(n)$. Assume that the complement of Graph(G) is c.e. and let M be a TM that enumerates it. Use M to compute G(n) as follows: for each n, we simulate the computation of M on all pairs (n,k) until the largest k is found such that (n,k) is enumerated by M. Then G(n) is equal to k+1. But this contradicts the fact that G is not computable, since we have just show how to compute it using a computably enumerable procedure. Therefore, the complement of Graph(G) is not c.e., which implies that Graph(G) is not co-c.e. ■

Problem 55.a.
Graph$^+$(G) is not decidable. To prove this, we can reduce the halting problem to Graph$^+$(G), which is known to be undecidable. Given a TM M with an input w, we can construct a function f: $N \to N$ that is defined as follows: For any n, if M halts on input w in n or fewer steps, then let f(n) be the max number of steps M takes on any input. Otherwise, let f(n)=0. Given M and w, construct f as described and consider the upper graph of f, Graph$^+$(f). We can see that (n,k) $\in$Graph$^+$(f) if and only if $k >$G(f(n)), since G(f(n)) is the maximum number of steps that M takes on any input, if it halts in n steps or fewer. Therefore, we can decide whether M halts on input w in n steps of fewer

by checking whether (n,k) $\in$Graph$^+$(f) for some k. So, Graph$^+$(G) is undecidable. ∎

Problem 55.b.
Graph$^+$(G) is c.e. To prove this, we can construct a TM M that enumerates all pairs (n,k) $\in$ Graph$^+$(G) like so: (1) for each n, simulate G(n) until the smallest k is found such that k $>$ G(n). (2) if such a k exists, output the pair (n,k). This TM will eventually enumerate all pairs (n,k) $\in$ Graph$^+$(G), since G is a rapidly growing function and k$>$G(n) can only happen for a finite number of n's. So, Graph$^+$(G) is c.e. ∎

Problem 55.c.
Graph$^+$(G) is not co-c.e. We can reduce the complement of the halting problem to Graph$^+$ (G). Given a TM M and an input w, construct a function f: $N \to N$ that is defined as follows: For any n, if M halts on input w in n or fewer steps, then let f(n)=1. Otherwise let f(n)=0. We can now show that there is a reduction from the complement of the halting problem to Graph$^+$(G). Given M and w, we construct f as described, and we consider the complement of the upper graph of f as the set $\{(n, k) \in N \times N | k \leq f(n)\}$. This set is the lower graph of a function g: $N \to N$, where g(n)=0 if M halts on input w in n steps or fewer, and g(n)=1 otherwise. Therefore, we can decide whether M halts on input w in n steps or fewer by checking whether (n,k) $\in$ Graph$^+$(G) for all $k \leq f(n)$. Therefore, Graph$^+$(G) is not co-c.e. ∎

Problem 56.a.
Graph$^-$(G) is not decidable. Given a TM M and an input x, we can construct a function f(n) that behaves like such: If M halts on input x in at most n steps, then f(n)=0. Otherwise, let k be the number of steps that M runs on input x before it halts. Then f(n)=k+1 for all $n > k$. f is a computable function, since we can simulate M on input x for n steps and determine whether it halts or not. Now, if we can decide (n,0) is in Graph$^-$(f) or not, we can determine whether M halts on input x or not. This is because (n,0) is in Graph$^-$(f) if and only if f(n)$>$ 0, which is true if and only if M halts on input x in at most n steps. We know that the halting problem is undecidable. So, there can't exist an algorithm that decides whether a given pair (n,k) is in Graph$^-$(G) or not. So, Graph$^-$(G) is not decidable. ∎

Problem 56.b.
Graph$^-$(G) is c.e. Construct a TM M that enumerates all pairs (n,k) where $k < G(n)$. M simulates G on all natural numbers in parallel, and whenever it finds a pair (n,k) such that $k < G(n)$, it outputs it. So Graph$^-$(G) is c.e. ∎

Problem 56.c.
Graph$^-$(G) is not co-c.e. To prove this, determine whether there exists an algorithm that can list all the pairs in $N \times N$ that are not in Graph$^-$(G). Let us assume there is such an algorithm. Then we can use it to decide whether a given pair (n,K) is in Graph$^-$(G) or not. Run the algorithm to see if (n,k) is listed or not. If it is listed, then it is not in Graph$^-$(G). Otherwise it is in Graph$^-$(G). But, as shown in 56a, Graph$^-$ is not decidable. Therefore, there can't exist an algorithm that can list all pairs in $N \times N$ that are not in Graph$^-$(G). So Graph$^-$(G) is not co-c.e. ∎