CS 124 Homework 6: Spring 2022

Your name: Lev Kruglyak

Collaborators: Sahil Kuchlous

No. of late days used on previous psets: 12
No. of late days used after including this pset: 12^+

Homework is due Wednesday at midnight ET. You are allowed up to **twelve** (college)/**forty** (extension school) late days through the semester, but the number of late days you take on each assignme nt must be a nonnegative integer at most **two** (college)/**four** (extension school).

Try to make your answers as clear and concise as possible; style may count in your grades. Assignments must be submitted in pdf format on Gradescope. If you do assignments by hand, you will need to scan in your results to turn them in.

You can collaborate with other students that are currently enrolled in this course in brainstorming and thinking through approaches to solutions but you should write the solutions on your own: you must wait one hour after any collaboration or use of notes from collaboration before any writing in your own solutions that you will submit.

For all homework problems where you are asked to give an algorithm, you must prove the correctness of your algorithm and establish the best upper bound that you can give for the running time. Generally better running times will get better credit; generally exponential time algorithms (unless specifically asked for) will receive no or little credit. You should always write a clear informal description of your algorithm in English. You may also write pseudocode if you feel your informal explanation requires more precision and detail, but keep in mind pseudocode does NOT substitute for an explanation. Answers that consist solely of pseudocode will receive little or not credit. Again, try to make your answers clear and concise.

Problem 1.

(a) **(10 points)** Prove that 1247536987 is composite by finding a witness in the form of a positive integer $a < 1247536987$ such that $a^{1247536987-1} \not\equiv 1 \pmod{1247536987}$. Give any information necessary to show that your witness in fact witnesses.

(b) **(10 points)** The number 75361 is a Carmichael number. Prove that it is composite by finding a witness in the form of a nontrivial square root of 1.

(a) We claim that 2 is a valid witness. Observe that it is nonzero modulo 1247536987, so if 1247536987 were prime then we would have

$$2^{1247536987-1} \equiv 1 \mod 1247536987,$$

yet this isn't the case. Using the efficient modular exponentiation algorithm, we get that

$$2^{1247536987-1} \equiv 448996154 \mod 1247536987.$$

(b) A simple check reveals that $6852^2 \equiv 1 \mod 75361$. This means that $1, 6852$, and $-1 = 75360$ are roots of $x^2 - 1$ modulo 75361, which contradicts the primality of 75361, so 75361 is composite.

Problem 2.

(a) **(15 points)** Adam has generated an RSA private key (p, q) and published $n = pq$. Adam wanted to be sure that his primes were both big enough, so he decided on the largest positive integer X his system could handle, picked some small positive integer δ, and generated both primes in the interval $[X - \delta, X]$. Give an $O(\delta \log^{124} n)$ algorithm to find p and q given n. (Your algorithm isn't given δ or X: it should work for all n, but the runtime may depend on what δ and X were.)

(b) **(5 points)** The public n that Adam generated as above was 15624998178250052834113. Factor it.

(a) Since both p and q are chosen from the interval $[X - \delta, X]$, it follows that $n = pq$ must be in the interval $[(X - \delta)^2, X^2]$, and so $X - \delta \leq \sqrt{n} \leq X$. We don't know where in the interval $\sqrt{n}$ is or even how big it is, so we can start with $\lfloor \sqrt{n} \rfloor$ and "spiral outwards" to try to find the prime factors. More explicitly: We start some integer k at 0. Then letting $p = \lfloor \sqrt{n} \rfloor + k$ and if n is divisible by p, $q = n/p$. Then we check if p and q are primes using some $O(\log^{124} n)$ algorithm such as the recently discovered AKS algorithm. If both p and q are primes, we are done, since $pq = n$. We do the same thing for $p = \lfloor \sqrt{n} \rfloor - k$ to hit both sides of the interval. (We can avoid an infinite loop) by breaking if $p < 0$ or something.) Notice that by the assumptions of the problem, the largest k can become is $\frac{\delta}{2}$, so we have to perform $O(\delta)$ primality checks. Since each primality check is $O(\log^{124} n)$, the overall algorithm is $O(\delta \log^{124} n)$. (Also note that all multiplications and divisions can be done in $O(\log^{124} n)$ so we don't have to worry about this.)

(b) Implementing this algorithm in Python, we get the factorization

$$15624998178250052834113 = 124999992197 \times 124999993229.$$

Problem 3. **PokéAllocation Problem:** In light of the April 10th, 2022 Mudkip Community Day, Tarun is bringing back his mudkip sales! Below is a reminder of how Tarun allocates his mudkips.

> Tarun has M extra mudkips that he would like to give to his trainer friends $t_0, \ldots, t_{|T|-1}$. Each trainer t_i would pay Tarun some positive integer p_i dollars for some positive integer m_i mudkips. The offers are all-or-nothing: each trainer t_i will walk away with either 0 or m_i mudkips. Let a "profile" be a set of values of M , $|T|$, p_i, and m_i as above.

This year, Tarun is hoping to buy two mudkip plushies (so they have friends)! Each of them is priced at $\$P$ for some positive integer P, for $\$2P$ total. Tarun wants to know whether or not it's possible, given a profile, to receive a profit of exactly $\$2P$; this is called the **PokéAllocation Problem**.

PokéGov Problem: In Tarunville, a self-governing city of happily coexisting Pokémon, the Pokémon live in $|R|$ regions $r_0, \ldots, r_{|R|-1}$. For example, in the Lake Tarun region (he names everything in his city after himself), water-type Pokémon, like mudkips, wander freely. In region r_i, there are p_i Pokémon.
Tarun would like two Pokémon representatives to help him make decisions (he was originally going to pick one, but if the Pokémon disagreed with Tarun, we would get Poké-gridlock leading to PokéGov-shutdown). To do this, he wants to divide the regions into two groups such that the two groups have equal populations. Then, each group would elect a single representative! This problem is called the **PokéGov Problem** and is NP-hard.

(a) **(2 points)** Show that the **PokéAllocation Problem** is in NP.

(b) **(2 points)** Show that the **PokéGov Problem** is in NP.

(c) **(15 points)** Give a polynomial-time algorithm that's a reduction from the **PokéGov Problem** to the **PokéAllocation Problem**: your algorithm should take as input an instance of the **PokéGov Problem** and output an instance of the **PokéAllocation Problem** with the same yes/no answer.

(a) To verify a solution to **PokéAllocation**, we simply check if Tarun's plan to sell mudkips adds up to a profit of exactly $\$2P$. This can be done in constant time.

(b) Given some partition of the regions into groups of equal population, we simply have to add up the populations in both regions and see if they are equal. This takes constant time, so **PokéGov** is NP.

(c) Suppose we have some input for **PokéGov** consisting of pairs (r_i, p_i) for $0 \le i \le |R| - 1$. Let P be half of the total population, i.e. $P = \frac{1}{2}\sum_{i \in R} p_i$. (If not divisible by 2, we can pass in some bogus input to **PokéAllocation** since clearly **PokéGov** is impossible.) Then, we simply set $M = |R|$ with $(m_i, p_i) = (1, 2p_i)$ where the RHS p_i is the number of mudkips in the r_i region. We claim that there is a one to one solution between solutions of this problem to solutions to **PokéGov**, this will prove that this in fact a reduction.

First, suppose there were a valid solution to **PokéGov**, say some set $S \subset R$ with $\sum_{i \in S} p_i = \sum_{i \in R-S} p_i$. Then $\sum_{i \in S} p_i = \frac{1}{2}\sum_{i \in R} p_i = P$. So if we sold to trainers t_i for all $i \in S$, we would sell $|S|$ mudkips for $\sum_{i \in S} 2p_i = 2P$, so **PokéAllocation** would return true. Similarly, if **PokéAllocation** returned true, it means that there is some collection $S \subset R$ of mudkips with $\sum_{i \in S} 2p_i = 2P$ so $\sum_{i \in S} p_i = P$. Then clearly S and $R - S$ are a valid solution to **PokéGov**. This proves that the answers to the two are equivalent, making this a valid reduction.

Problem 4. Let $G = (V, E)$ be a graph. A *vertex cover* of G is a set $C \subseteq V$ such that all edges in E have at least one endpoint in C. That is, each edge is adjacent to at least one vertex in the vertex cover. The **(Minimum) Vertex Cover** problem is, given a graph G and a number K, to determine if G has a vertex cover of size at most K. The lecture 18 notes have a proof that Minimum Vertex Cover is NP-complete.

(a) **(5 points)** We know that that all NP-complete problems reduce to each other. It would be nice if this meant that an approximation for one NP-hard problem would lead to another, but this is not the case. Consider the case of Minimum Vertex Cover, for which we have a 2-approximation; that is, we have an algorithm to find a vertex cover of size within a factor of 2 of optimal. (We'll see this in lecture next week, but it's not relevant to this problem.) A set C is a vertex cover in a graph $G = (V, E)$ if and only if $V - C$ is an independent set in V. Explain why this does not yield an approximation algorithm that is within a constant factor of optimal for Maximum Independent Set. That is, show that for every constant $c > 1$, there exists a graph and a 2-approximation of its Minimum Vertex Cover such that the corresponding independent set is not within a factor of c of the Maximum Independent Set.

(b) **(15 points)** Prove that it's NP-hard to approximate the size of the maximum independent set in a graph to within 124 vertices.

(c) **(20 points)** Prove that if there exists a polynomial time algorithm for approximating the maximum independent set in a graph G to within a factor of 2, then for every $\epsilon > 0$, there is a polynomial time algorithm for approximating the maximum independent set in a graph to within a factor of $(1 + \epsilon)$. The degree of the polynomial may depend on ϵ.

(a) Consider the graph consisting of N pairs of vertices each connected by a single edge. (They are all disjoint) Then clearly the minimum vertex cover is of size N, with one vertex covered in each pair. Then the vertex cover where all vertices but one is a 2-approximation of the minimal vertex cover. The corresponding independent set has size 1, which has a ratio of $\frac{1}{N}$ to the maximal independent independent set. This can be made arbitrarily small as $N \to \infty$ so we have our desired counterexample.

(b) Let $m(G)$ be the size of the maximum independent set in a graph G. Given some graph G and integer $k > 1$, the 124-MIS problem can be described as a function

$$\mathrm{MIS}_{124}(G, k) = \begin{cases} \text{yes} & m(G) \geq k + 124 \\ \text{no} & m(G) \leq k - 124 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Here "undefined" simply means that the function can be either "yes" or "no", we don't care. Let $\bigsqcup_N G$ be the graph which is a disjoint union of N copies of G, i.e. no edges between them. Then clearly $m\left(\bigsqcup_N G\right) = N \cdot m(G)$. Recall that the regular MIS problem is defined as $\mathrm{MIS}(G, k) = [m(G) \geq k]$. Now we claim that

$$\mathrm{MIS}(G, k) = \mathrm{MIS}_{124}\left(\bigsqcup_{248} G, 248k - 124\right).$$

Since MIS is NP-hard, this reduction shows that 124-MIS is NP-hard as well. To prove this, notice that on the RHS, the condition $m(G) \geq k + 124$ becomes

$$m\left(\bigsqcup_{248} G\right) \geq (248k - 124) + 124 \implies m(G) \geq k$$

and the condition $m(G) \leq k - 124$ becomes

$$m\left(\bigsqcup_{248} G\right) \leq (248k - 124) - 124 \implies m(G) \leq k - 1.$$

Thus $\mathrm{MIS}_{124}\left(\bigsqcup_{248} G, 248k-124\right)$ is never "undefined", and is in fact equal to $[m(G) \geq k]$, which is exactly $\mathrm{MIS}(G,k)$. This completes the proof.

(c) Suppose we had a polynomial time algorithm 2-MIS for finding an independent set of size at least $m(G)/2$ for some input graph G. We claim then that for any $\epsilon > 0$ there is a polynomial time algorithm for determining an $(1+\epsilon)$-approximation of the maximal independent set. First we'll prove a lemma:

Lemma. For any graph $G = (V,E)$ and positive integer n, we have $m(G^n) = m(G)^n$ where $G^n = G \times \cdots \times G$.

Proof. Suppose $X \subset V^n$ is an independent set in G^n. Notice that $\pi_i(X) \subset V$ is an independent set in G, where π_i is the projection of V^n onto the i-th component. This is because if there were an edge connecting two points x_1, x_2 in $\pi_i(X)$, by definition of cartesian product of a graph we would have an edge connecting $(\ldots, x_1, \ldots)$ and $(\ldots, x_2, \ldots)$ in G^n. Recall that $|X| \leq \prod_i |\pi_i(X)|$ so if X is a maximal independent set of G^n then $|X| \leq m(G)^n$, since each $\pi_i(X)$ is an independent set of G and so $|\pi_i(X)| \leq m(G)$. So $m(G^n) \leq m(G)^n$. To prove the other direction, let Y be a maximal independent of G. Then clearly $Y \times \cdots \times Y$ is an independent set of G^n, again by definition of the cartesian product. So $m(G^n) \geq m(G)^n$. This implies that $m(G^n) = m(G)^n$. □

Then for any positive integer n, by the Lemma, running 2-MIS(G^n) will give us an independent set in G^n of size at least $m(G)^n/2$. Let X be the output set of this algorithm, and consider the projections $\pi_1(X), \ldots, \pi_n(X)$. Since $|X| \geq m(G)^n/2$ and $|X| \leq \prod_i |\pi_i(X)|$, we have

$$\frac{m(G)^n}{2} \leq \prod_i |\pi_i(X)|.$$

This means that for the i with the maximal $|\pi_i(X)|$, we have $|\pi_i(X)| \geq \frac{m(G)}{\sqrt[n]{2}}$. To summarize all of the results, given some graph G and $\epsilon > 0$, set n to $\left\lceil \frac{1}{\log_2(1+\epsilon)} \right\rceil$. Then compute 2-MIS$(G^n)$, call the output say X. Select the largest size set $\pi_i(X)$, then this is a $(1+\epsilon)$-approximation of the maximal independent set of G. This is true by the preceding results and since $\sqrt[n]{2} \leq 1+\epsilon$.

Problem 5. **(25 points)** Gossip Girl and Chatter Charlie are at it again! They've been joined by $m-2$ of their gossip blogger friends. As before, there's a set of tips available to them: each tip is both an edge connecting two of the n students at Constance Billard High School and a subset of the m gossip bloggers. (For instance, a tip might be "Alice and Bob are friends. This tip is for Gossip Girl and Gossip Bloggers 17 and 124.") We'd like to decide whether there's a subset S of size k of the tips such that, for each of the gossip bloggers x, the subset of S consisting of the tips for x is enough to connect the whole school. Prove that this problem is NP-complete.

We'll prove that this problem is NP-complete by showing that it's at least as hard as set cover, which is NP-complete. So suppose we have some set X and set cover $\mathcal{U} \subset \mathcal{P}(X)$. We want to find the smallest subset of $\mathcal{U}$ which covers X. Consider a school with only two students and $|X|$ bloggers. For each set $S \subset \mathcal{U}$ we add a tip which connects the two students, and gives it to the bloggers corresponding to the elements of the set S. Then the answer to the gossip girl problem for this school and the set cover problem are the same. This is because a solution to the gossip girl problem must connect the graph, so every blogger must have a proof that the two students are connected. So each set of tips which is a valid gossip girl set must correspond to a set cover. Since in both problems we are looking for the minimum such set, it follows that the answer to the problems is the same with this reduction. Thus the gossip girl problem is NP-complete.