CS 170 HW 15

Due on

1 Study Group

List the names and SIDs of the members in your study group.

2 Reduction Essentials

Assume A and B are search problems, and A reduces to B in polynomial time. In each part you will be given a fact about one of the problems. Determine what, if anything, this allows you to determine about the other problem. *Answer each part in one sentence.*

(a) A is in **P**

(b) B is in **P**

(c) A is **NP**-hard

(d) B is **NP**-hard

Solution:

(a) Nothing

(b) A is in **P**

(c) B is **NP**-complete

(d) Nothing

3 NP-complete problems

Prove that the following problems are NP-hard. In each case, specify which problem you are reducing from, which problem you are reducing to. Briefly, but precisely describe how you transform an instance of one problem to another.

(a) Directed Rudrata Cycle.
Input: A directed graph $G = (V, E)$.
Goal: Find a directed cycle that visits every vertex in V exactly once.

(b) Californian Cycle
Input: A directed graph $G = (V, E)$ with each vertex colored *blue* or *gold*, i.e., $V = V_{blue} \cup V_{gold}$. **Goal** Find a *Californian cycle* which is a directed cycle through all vertices in G that alternates between blue and gold vertices. (*Hint: Directed Rudrata Cycle*)

(c) 4-SAT
Input: n boolean variables $\{x_1,$
$ldots, x_n\}$ and clauses $\{C_1,$
$ldots, C_m\}$ with each having exactly

four distinct literals. For example, the following is an instance of $4 - SAT$.

$$(x_1 \vee x_2 \vee \overline{x_4} \vee \overline{x_5}) \wedge (x_3 \vee \overline{x_4} \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_3 \vee x_4 \vee x_5)$$

. Note that all the 4 literals within a clause have to be distinct. **Goal:** Find an assignment to the variables $x_1, \ldots, x_n$ that satisfies all the clauses.

Solution:

(a) We reduce from
textsc{Rudrata Cycle}. To convert an undirected graph to a directed graph, we replace each undirected edge (u, v) with two directed edges, (u, v) and (v, u).

(b) We reduce Directed Rudrata Cycle to Californian Cycle, thus proving the NP-hardness of Californian Cycle.

Given a directed graph $G = (V, E)$, we construct a new graph $G' = (V', E')$ as follows:

- For each $v \in V$, create a blue node v_b with an edge to a gold node v_g (in G').
- For each $(u, v) \in E$, add edge (u_g, v_b) to E'. Another way to view this is that for each node $v \in V$, we are redirecting all its incoming nodes to v_g, and all its outgoing nodes originate from v_b (in G').

Here are some common attempts that may look reasonable, but are incorrect:

(a) Color the graph (does not matter how), then find a Californian cycle.
(b) For all $v \in V$, add a blue node $v' \in V'$. For every edge $(u, v) \in E$, add $(u, w) \in E'$ and $(w, v) \in E'$, where w is a new gold node.
(c) For all $v \in V$, create blue node v_b and gold node v_g in G'. For each edge $(u, v) \in E$, create edges (u_b, v_g) and (u_g, v_b) in G'.
(d) Same as the previous construction, but also add edges (v_g, v_b) and/or (v_b, v_g).
(e) Make two copies of G: one blue, one gold, each one with the same edges as in G. Then add edges between corresponding nodes of opposite color.

4 Hashing

Given a prime p and $a, b \in \{0, \ldots, p-1\}$, define the function $h_{a,b}(x) = ax + b \bmod p$ where $x \in \{0, \ldots, p-1\}$. Show that $H = \{h_{a,b}\}_{a,b \in \{0,\ldots,p-1\}}$ is a pairwise independent hash function family, i.e., show that for every $x \neq y$ and $c, d \in \{0, \ldots, p-1\}$ it holds that

$$\Pr_{h_{a,b} \leftarrow H} \Big[h_{a,b}(x) = c \wedge h_{a,b}(y) = d \Big] = \frac{1}{p^2} \ .$$

The notation $h_{a,b} \leftarrow H$ means that $h_{a,b}$ is chosen uniformly at random from H, meaning a and b are chosen independently uniformly at random from $\{0, \ldots, p-1\}$.

Solution: For this entire solution, all equations are mod$p$.

$h(x) = c$ and $h(y) = d$ if and only if $ax + b = c$ and $ay + b = d$. This is true if and only if a, b solve the system of equations

$$ax + b = c$$
$$ay + b = d$$

Solving this, we have

$$a = (c - d)(x - y)^{-1}$$
$$b = (cy - dx)(y - x)^{-1}$$

We are guaranteed that the multiplicative inverses exist because p is prime, and we know $x \neq y$. Thus, there is only one value of a and one value of b that satisfy these equations, which are chosen independently at random: the probability of this occurring is $1/p^2$.

5 Streaming Algorithms

In this problem, we assume we are given an infinite stream of integers x_1,x_2, ..., and have to perform some computation after each new integer is given. Since we may see many integers, we want to limit the amount of memory we have to use in total. For all of the parts below, give a brief description of your algorithm and a brief justification of its correctness.

(a) Show that using only a single bit of memory, we can compute whether the sum of all integers seen so far is even or odd.

(b) Show that we can compute whether the sum of all integers seen so far is divisible by some fixed integer N using $O(\log N)$ bits of memory.

(c) Assume N is prime. Give an algorithm to check if N divides the product of all integers seen so far, using as few bits of memory as possible.

(d) Now let N be an arbitrary integer, and suppose we are given its prime factorization: $N = p_1^{k_1} p_2^{k_2} \ldots p_r^{k_r}$. Give an algorithm to check whether N divides the product of all integers seen so far, using as few bits of memory as possible. Write down the number of bits your algorithm uses in terms of $k_1, \ldots, k_r$.

Solution:

(a) We set our single bit to 1 if and only if the sum of all integers seen so far is odd. This is sufficient since we don't need to store any other information about the integers we've seen so far.

(b) Set $y_0 = 0$. After each new integer x_i, we set $y_i = y_{i-1} + x_i \mod N$. The sum of all seen integers at step i is divisible by N if and only if $y_i \equiv 0 \mod N$. Since each y_i is between 0 and $N - 1$, it only takes $\log N$ bits to represent y_i.

(c) We can do this with a single bit b. Initially set $b = 0$. Since N is prime, N can only divide the product of all x_is if there is a specific i such that N divides x_i. After each new x_i, check if N divides x_i. If it does, set $b = 1$. b will equal 1 if and only if N divides the product of all seen integers.

(d) We can do this with $\lceil \log_2(k_1) \rceil + \lceil \log_2(k_2) \rceil + \cdots + \lceil \log_2(k_r) \rceil$ bits. For each i between 1 and r, we track the largest value $t_i \leq k_i$ such that p^{t_i} divides the product of all seen numbers. We start with $t_i = 0$ for all i. When a new number m is seen, we find the largest t_i' such that $p^{t_i'}$ divides m and set $t_i = \min\{t_i' + t_i, k_i\}$ We stop once $t_i = k_i$ for all i as this implies that N divides the product of all seen numbers.