# CS 180: Homework 6

Jonathan Woong

804205763

Winter 2017

Discussion 1B

Monday $20^{\text{th}}$ March, 2017

# Problem 1

Suppose that you are given as input a list of $n$ birthdays (in the format $MMDDYYYY$) give an algorithm to check if two people in the list have the same birthday. Your algorithm should always be correct and run in expected time $O(n)$.

Let $U$ be the set of all possible birthday combinations $b$.

Let $h(b)$ be the hasing function that maps a birthday $b$ to an integer in $\{1, \ldots, n\}$.

Let $\mathcal{D}$ be the database that stores hashed birthdays obtained by $h(b)$.

Let $L$ be the list of birthdays, with all $b \in L$.

ALGORITHM:

1. For $i = 1, \ldots, n$: If $Lookup(L[i])$ on $D$ returns true: RETURN $TRUE$.

   Else: $Insert(L[i])$ into $D$.

2. RETURN $FALSE$.

## Problem 2

Suppose you are writing a plagiarism detector. Students submit documents as part of a homework and each document is an (ordered) sequence of words. For some parameter $m$ decided by the provost, we say two documents are copies of each other if one of them uses a sequences of $m$ words (in that given order) from the other. Give an algorithm which given an integer $m$ and $N$ documents $D_1, \ldots, D_N$ as input, flags all submissions which are copies of some other submission. Your algorithm should run in expected $O(m + N + \text{total-length of documents})$ time (i.e., expected $O(m + N + n_1 + \ldots + n_N)$ time where $n_j$ is the length of $j$'th document) and be always correct.

INPUT:

$N =$ number of documents

$D = \{D_1, \ldots, D_N\}$ documents

$m =$ length of sequence

Let $\mathcal{D}$ be the dictionary that stores $(key, value)$ pairs using hashing with chaining.

Let $key$ match to sequences of $m$ words.

Let $value$ match to integers.

Let $r = \sum_i n_i$.

ALGORITHM:

1. Initialize array $C$ of length $N$, where $C[i] = 0\ \forall i = \{1, \ldots, N\}$.
2. For $i = 1, \ldots, N$:

    For $j = 1, \ldots, n_i - m$:

        Let $key = (D_i[j, j+m])$.

        Compute $value = Lookup(key)$.

        If $value = NULL$: $Insert(key)$ into $\mathcal{D}$.

        Else if $value \neq i$:, set $C[i] = 1$ and $C[value] = 1$.
3. RETURN $C$.

## Problem 3

Consider the problem FIND-CLIQUE defined as follows: "Given a graph $G$ and a number $k$ as input, find a clique of size $k$ in $G$ if one exists." Recall the CLIQUE decision problem from class: "Given a graph $G$ and a number $k$, does $G$ contain a clique of size $k$?". Give a polynomial-time reduction from FIND-CLIQUE to CLIQUE.

We reduce FIND-CLIQUE to CLIQUE by using a polynomial number of calls to a black-box that solves CLIQUE.

Let K-CLIQUE be the altered CLIQUE algorithm such that the input graph $F$ into K-CLIQUE is always size $k$; K-CLIQUE only needs to check if $F$ is a clique.

Divide $G = (V, E)$ into $j$ subgraphs of size $k$. The maximum number of subgraphs of size $k$ is $n^k$.

Let $g_i \in \{g_1, \ldots, g_j\}$ be the subgraphs of $G$ of size $k$.

ALGORITHM:

1. For $i = 1, \ldots, j$:

   Let $clique =$ K-CLIQUE$(g_i)$.

   If $clique = TRUE$, RETURN $g_i$.

2. RETURN $FALSE$.

This algorithm reduces the problem into at most $n^k$ instances of CLIQUE.

## Problem 4

Consider the problem LPS defined as follows: "Given a matrix $A \in \mathbb{R}^{n\times n}$, a vector $b \in \mathbb{R}^n$ and an integer $k > 0$, does there exist a vector $x \in \mathbb{R}^n$ with at most $k$ non-zero entries such that $A \cdot x \geq b$". Here $A \cdot x$ denotes the usual matrix-vector product and for two vectors $u, v$, we say $u \geq v$ if for every $i$, $u_i \geq v_i$. Give a polynomial-time reduction from 3SAT to LPS.

We can show that 3SAT $\leq_p$ LPS by showing that VERTEX-COVER $\leq_p$ LPS.

Since 3SAT $\leq_p$ VERTEX-COVER, if VERTEX-COVER $\leq_p$ LPS, then by transitivity 3SAT $\leq_p$ LPS.

Let M-LPS be defined as: Given a matrix $A' \in R^{m\times n}$ and a vector $b' \in R^m$ and an integer $k > 0$, does there exist a vector $x \in R^n$ with at most $k$ non-zero entries such that $A' \cdot x \geq b'$.

Show that VERTEX-COVER $\leq_p$ M-LPS:

1. Given an instance of vertex cover with graph $G$ and integer $k$, define an instance of M-LPS as: Let $G = (V, E)$ where $E = \{e_1, \dots, e_m\}$ are the edges in $G$ and $V = \{v_1, \dots, v_m\}$ are the vertices. Let $A'$ be the $m \times n$ matrix where $A'_{ij} = 1$ if edge $e_i$ is adjacent to vertex $v_j$ and 0 otherwise. Let $b' \in R^n$ be the vector with all entries being 1.

CLAIM: $G$ has a vertex-cover of size $k$ iff there is a vector $y$ with at most $k$ non-zero entries such that $A'y \geq b$.

PROOF:

FORWARD: Suppose there is a vertex cover $Y \subset [n]$ of size at most $k$. Let $y = 1(Y)$ be the vector with $y_j = 1$ if $j \in Y$ and 0 otherwise. For any $i \in [m]$, $(A'y)i = \sum_{j=1}^{n}(A'_{ij}y_j) = \sum_{j\in Y} A'_{ij} \geq 1$ (since $e_i$ should be adjacent to at least one vertex of $Y$).

$\therefore A'y \geq b'$

BACKWARD: Suppose there is a vector $y \in R^n$ with at most $k$ non-zero entries such that $A'y \geq b'$. Let $Y = \{j : y_j \neq 0\}$. $|Y| \leq k$ and $Y$ is a vertex-cover. This is because for an index $i \in [m]$, $(A'y)_i \geq b_i = 1$ and $(A'y)_i \neq 0$. We know that $(A'y)_i = \sum_{j=1}^{n} A'_{ij}y_j$, and for $(A'y)_i$ to be non-zero, one of the summands of $A'_{ij}y_j$ should be non-zero. This is only true when the edge $e_i$ is adjacent to a vertex in $Y$.

$\therefore Y$ is a vertex-cover.

This shows that $G$ has a vertex-cover of size $k$ iff there is a vector $y$ with at most $k$ non-zero entries such that $A'y \geq b'$. Since an instance of M-LPS can be built in polynomial time, we can use a black-box for M-LPS to solve VERTEX-COVER.

$\therefore$ VERTEX-COVER $\leq_p$ M-LPS.

2. Consider an instance of M-LPS given by $A', b', k$. If $m = n$, we have an instance of $LPS$.

If $m > n$, build a matrix $A \in R^{m \times m}$ by adding $m - n$ columns of length $m$ filled with zeroes to the matrix $A'$. A new instance of $LPS$ will be specified by $A, b', k$. We can check that the instance of M-LPS has a solution iff our constructed instance $A, b', k$ has a solution.

If $m < n$, build a matrix $A \in R^{n \times n}$ by adding $n - m$ rows of length $n$ filled with zeroes to the matrix $A'$. Let $b \in R^n$ be the vector obtained by adding $n - m$ zeroes to the entries of $b'$. A new instance of LPS is specified by $A, b, k$. We can check that the instance of M-LPS has a solution iff our constructed instance $A, b', k$ of LPS has a solution.

We can use a black-box for LPS to solve M-LPS because the instances of LPS can be constructed in polynomial time.

$\therefore$ M-LPS $\leq_p$ LPS.

Arguments 1 and 2 show that VERTEX-COVER $\leq_p$ LPS.

## Problem 5

For this problem we need the notion of multi-variate polynomials over variables $x_1, \ldots, x_n$ and how they are specified. To review some terminology, we say a *monomial* is a product of a real-number co-efficient $c$ and each variable $x_i$ raised to some non-negative integer power $a_i$: we can write this as $cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$. A polynomial is then a sum of a finite set of monomials.

We say a polynomial $P$ is of degree at most $d$, if for any monomial $cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$ appearing in $P$, $a_1 + a_2 + \cdots a_n \leq d$. (For example, the degree of the previous polynomial is 7). One can represent a $n$-variable polynomial of degree $d$ by at most $(n+1)^d$ numbers.

Consider the problem Poly-Root defined as follows: "Given a polynomial with integer coefficients of degree at most 6 as input, decide if there exists a $x \in \mathbb{R}^n$ such that $P(x) = 0$." Show that 3SAT reduces to Poly-Root. You don't have to write down the coefficients of the polynomials explicitly in your reduction - you can leave them as summations if it is more convenient for you.

For a term $y$, let $P_y$ be the polynomial defined as:

If $y = x_i$ for a variable: $P_y = 1 - x_i$.

Else if $y = \overline{x_i}$: $P_y = x_i$.

For every clause $C$, let $P_C$ be the polynomial obtained by multiplying the polynomials $P_y$ for every term $y$ that appears in $C$.

Given a 3SAT instance $\phi = C_1 \vee \cdots \vee C_k$, let $P_\phi = P_{C_1}^2 + \cdots + P_{C_k}^2$.

CLAIM: $\phi$ is satisfiable iff there exists an $x \in R^n$ such that $P_\phi(x) = 0$.

PROOF:

FORWARD: Suppose that $\phi$ is satisfiable and let $a$ be a satisfying assignment for $\phi$. Then $P_{C_j}(a) = 0$ for every clause $C_j$, since at least one of the terms in $C_j$ would be satisfied by $a$.

$\therefore P_\phi(a) = 0$.

BACKWARD: Suppose there exists a vector $b \in R^n$ such that $P_\phi(b) = 0$. Define an assignment $a$ as:

For each $i \in [n]$:

If $b_i \in \{0, 1\}$: set $a_i = b_i$.

Else: set $a_i$ to 1.

CLAIM: $a$ is a satisfying assignment for $\phi$.

PROOF: If $P_\phi(b) = 0$, then $P_{C_j}(b) = 0$ for every $1 \leq j \leq k$. Let $C_j = y_{j1} \vee \cdots \vee y_{j3}$. Since $P_{C_j} = P_{y_j1}P_{y_j2}P_{y_j3}$, $P_{C_j}(b) = 0$ iff one of $P_{y_j1}(b), P_{y_j2}(b), P_{y_j3}(b) = 0$. Any of these polynomials is zero iff the corresponding value of $b \in \{0, 1\}$ and satisfies the associated term.

$\therefore$ From our definition of $a$, $a$ satisfies the clause $C_j$. Since this applies to every clause, $a$ satisfies the CNF formula $\phi$.

The arguments show that $\phi$ has a satisfying assignment iff the polynomial $P_\phi$ has a root. Since we can build $P_\phi$ in polynomial time and it has degree at most 6, we can use a black-box for POLY-ROOT to solve 3SAT. $\therefore$ 3SAT $\leq_p$ POLY-ROOT.