# CS 170 HW 14

# Due on 2019-05-06, at 9:59 pm

## 1 Study Group

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

## 2 Convex Hull

Given $n$ points in the plane, the *convex hull* is the list of points, in counter-clockwise order, that describe the convex shape that contains all the other points. Imagine a rubber band is stretched around all of the points: the set of points it touches is the convex hull.

In this problem we'll show that the convex hull problem and sorting reduce to each other in linear time.

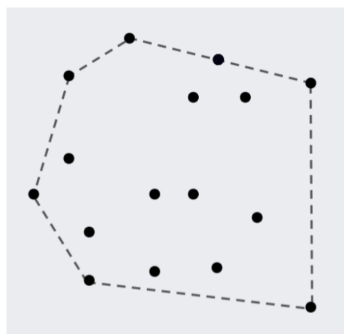

Figure 1: An instance of convex hull: the convex hull is the seven points connected by dashed lines. Note that three or more points in the convex hull can be collinear.

(a) Fill in the following algorithm for convex hull; you do not need to prove it correct. What is its runtime? For simplicity, in this part you may assume no three points are collinear.

**procedure** ConvexHull(list of points $P[1..n]$)
  Set $low$ := the point with the minimum $y$-coordinate, breaking ties by minimum $x$-coordinate.
  Create a list $S[1..n-1]$ of the remaining points sorted increasingly by the angle between the vector $point - low$ and the vector $(1, 0)$ (i.e the x-axis) .
  Initialize $Hull := [low, S[1]]$
  **for** $p \in S[2..n-1]$ **do**
    *<fill in the body of the loop>*
  Return $Hull$

(b) Now, find a linear time reduction from sorting to convex hull. In other words, given a list of real numbers to sort, describe an algorithm that transforms the list of numbers into a list of points, feeds them into convex hull, and interprets the output to return the sorted list. Then, prove that your reduction is correct.

## 3 Decision vs. Search vs. Optimization

The following are three formulations of the VERTEX COVER problem:

- As a *decision problem*: Given a graph $G$, return TRUE if it has a vertex cover of size at most $b$, and FALSE otherwise.
- As a *search problem*: Given a graph $G$, find a vertex cover of size at most $b$ (that is, return the actual vertices), or report that none exists.
- As an *optimization problem*: Given a graph $G$, find a minimum vertex cover.

At first glance, it may seem that search should be harder than decision, and that optimization should be even harder. We will show that if any of the above variants of Vertex-Cover can be solved in polynomial time, all the other variants are also solvable in polynomial time.

For the following parts, describe your algorithms precisely; justify correctness and the number of times that the black box is queried (asymptotically).

(a) Suppose you are handed a black box that solves VERTEX COVER (DECISION) in polynomial time. Give an algorithm that solves VERTEX COVER (SEARCH) in polynomial time.

(b) Similarly, suppose we know how to solve VERTEX COVER (SEARCH) in polynomial time. Give an algorithm that solves VERTEX COVER (OPTIMIZATION) in polynomial time.

## 4 2-SAT and Variants

In this problem we will explore the variant of 2-SAT called Max-2-SAT. Recall that 2-SAT is in instance of SAT where each clause contains at most 2 literals (hereby called a 2-clause). As seen previously, 2-SAT can be solved efficiently via a graph theoretic techniques and is hence in $\mathsf{P}$. We will show that a subtle modification of the original 2-SAT problem renders the problem computationally hard.

The problem of Max-2-SAT is defined as follows. Let $C_1, \ldots, C_m$ be a collection of 2-clauses and $k$ a non-negative integer. We want to determine if there is some assignment which satisfies at least $k$ clauses.

The problem of Max-Cut is defined as follows. Let $G$ be an undirected unweighted graph, and $k$ a non-negative integer. We want to determine if there is some cut with at least $k$ edges crossing it. Max-Cut is known to be $\mathsf{NP}$-complete.

Show that Max-2-SAT is $\mathsf{NP}$-complete by reducing from Max-Cut. Prove the correctness of your reduction.