CS 170 HW 10

Due on 2018-11-04, at 9:59 pm

1 (★) Study Group

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

2 (★★★) Existence of Perfect Matchings

Prove the following theorem: Let $G = (L \cup R, E)$ be a bipartite graph. Then G has a perfect matching if and only if, for every set $X \subseteq L$, X is connected to at least $|X|$ vertices in R. Note that you must prove both directions. (Hint: Use the max-flow-min-cut theorem.)

3 (★★) A Reduction Warm-up

In the Rudrata path problem (aka the Hamiltonian Path Problem), we are given a graph G and want to find if there is a path in G that uses each vertex exactly once.

Is the following argument correct? Please justify your answer.

We will show that Undirected Rudrata Path can be reduced to Longest Path in a DAG. Given a graph G, use DFS to find a traversal of G and assign directions to all the edges in G based on this traversal (i.e. edges will point in the same direction they were traversed and back edges will be omitted). This gives a DAG. If the longest path in this DAG has $|V| - 1$ edges then there is a Rudrata path in G since any simple path with $|V| - 1$ edges must visit every vertex.

4 (★★★) 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 one can be solved in polynomial time, so can the others.

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.

5 (★★★) 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.

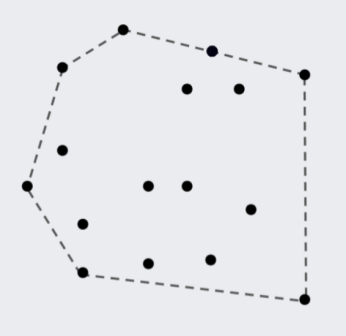

Figure 1: An instance of convex hull: the convex hull is the seven points connected by dashed lines. Note that three or more points in the convex hull can be collinear.

(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.

6 (★★★) More Reductions

Given a vector of non-negative integers $[a_1, a_2, \ldots, a_n]$, consider the following problems:

1 **Partition**: Determine whether there is a subset $P \subseteq [n]$ ($[n] := \{1, 2, \cdots, n\}$) such that $\sum_{i \in P} a_i = \sum_{j \in [n] \setminus P} a_j$

2 **Subset Sum**: Given some integer k, determine whether there is a subset $P \subseteq [n]$ such that $\sum_{i \in P} a_i = k$

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, as well as fixed numbers W and V there is some subset P such that $\sum_{i \in P} w_i \leq W$ and $\sum_{i \in P} v_i \geq V$

For each of the following clearly describe your reduction, justify runtime and correctness.

(a) Find a linear time reduction from Subset Sum to Partition.

(b) Find a linear time reduction from Subset Sum to Knapsack.

7 (★) Runtime of NP

True or False (with brief justification): Suppose we can show for some fixed k, an NP-complete problem P has a time $O(n^k)$ algorithm. Then every problem in NP has a $O(n^k)$ time algorithm.