CS170–Spring 2019 — Homework 14 Solutions

Ran Liao, SID 3034504227

May 3, 2019

Collaborators:Jingyi Xu, Renee Pu

1 Study Group

Name	SID
Ran Liao	3034504227
Jingyi Xu	3032003885
Renee Pu	3032083302

2 Convex Hull

(a) **Procedure**

while a right turn is needed for the the last two points in *Hull* to reach p

    remove the last point in *Hull*

add p to the end of *Hull*

Runtime

Each points can be added to the *Hull* at most once, and can be removed from *Hull* at most once. Therefore, the runtime in the nested loop is $O(n)$. Since sorting points will cost $O(n \log n)$ time. The overall runtime is $O(n \log n)$.

(b) **Reduction**

Suppose $x_1, x_2, \ldots, x_n$ are the input numbers to sorting algorithm.

A reduction R will construct the following points; $(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)$. And feed it to the convex hull algorithm.

Starting at point produced by convex hull algorithm with most negative x coordinate, counter-clockwise order of hull points yields items in ascending order.

Proof

Since all points R construct will lie on the function $f(x) = x^2$. And region $\{x : x^2 \geq x\}$ is convex. Therefore, all points are on convex hull and will be output by the convex full algorithm in counter clock wise order.

3 Decision vs. Search vs. Optimization

(a) **Psudocode**

SOLVE-SEARCH-PROBLEM($G = (V, E), b$){

$S \leftarrow$ empty set

$V_r \leftarrow V$

$E_r \leftarrow E$

for v in V\{

$e \leftarrow$ edges that connected by v in E_r

$E' \leftarrow E_r \setminus e$

$V' \leftarrow V_r \setminus v$

$G' \leftarrow (V', E')$

if SOLVE-DECISION-PROBLEM($G', b-1$){

$S \leftarrow S \cup v$

$b \leftarrow b - 1$

$E_r \leftarrow E'$

$V_r \leftarrow V'$

if $b == 0$ or $|E_r| == 0$

return S

}

}

return *none*

}

Proof of Correctness

I'd like to prove S will always cover all edges in $E \setminus E_r$.

In the beginning, both S and $E \setminus E_r$ are empty. This is trivial.

In the middle iteration, all edges in e can be covered by v. Therefore, if the remaining edges can be covered with at most $b-1$ vertices, it's safe to add e into S. And the above property still holds.

Therefore, when the algorithm ends, S will contain a valid vertex cover if solution exists.

Time Complexity

The black box will be called at most $O(|V|)$ times.

(b) We can use binary search idea to identify the minimal vertex cover with search problem black box. We stop when we find there's a vertex cover with size k, but there's no vertex cover with size $k-1$. Therefore, k is the optimal vertex cover size. And the search problem black box will give us the actual vertices in vertex cover. Since we use binary search, the black box will be called $O(\log |V|)$ times.

4 2-SAT and Variants

(a) **Reduction**

Given an instance $G = (V, E)$ for Max-Cut, we construct the following clauses for each edge $e = (u, v) \in E$.

$$x_u \vee x_v \qquad \overline{x_u} \vee \overline{x_v}$$

(b) **Proof**

For each cut in G, it's naturally to consider the vertex in one set is assigned with $true$, the other is assigned with $false$. Therefore, if edge $e = (u, v)$ cross between these two sets, both clauses mentioned above will be true. If it doesn't, one of the above clause will be true, the other will be false. Suppose there're k crossing edges. In total, there're $2k + |E| - |k| = |E| + k$ clauses will be satisfied.