COS 330: Great Ideas in Theoretical Computer Science

Fall 2026

Precept 3

Price Reduction

PROBLEM-SOLVING SESSION

Today is a **problem-solving session**. There is a single problem, with multiple parts, on the fine-grained reductions we saw in lecture: work on it on your own or in a small group, and ask your TA when you get stuck. Solutions will be shared after precept.

It's not extremely important that you remember the exact definitions from lecture; the precept is somewhat self-contained. If you remember the principles of fine-grained reductions, you should be able to solve the problems by reading the definitions carefully and thinking about what they mean.

Problem 1: It Takes Three to Tango

As we saw in lecture, 3-SUM is a very nice "hard" problem that can be used to establish the hardness of many other problems. Here is the definition in case you forgot:

Definition (3-SUM)**.** Given a list of n distinct integers $a_1, \ldots, a_n$, decide whether there are indices i, j, k (not necessarily distinct) such that $a_i + a_j + a_k = 0$.

Note: this is slightly different from the version we saw in lecture, where the three indices had to be distinct. Here an index may be used more than once, so, for example, $a_1 + a_1 + a_2 = 0$ counts as a solution. We use this version because it makes the reductions below cleaner, but the two versions are equivalent: each one can be reduced to the other, so everything we say about one applies to the other.

As we saw in lecture, it is conjectured that 3-SUM cannot be solved in time $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$. In this problem, we will show that another problem is "3-SUM hard" in the sense that if it can be solved in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$, then so can 3-SUM.

Definition (Colorful-3-SUM)**.** Given three lists of n distinct integers each, $A = (A_1, \ldots, A_n)$, $B = (B_1, \ldots, B_n)$, and $C = (C_1, \ldots, C_n)$, decide whether there are indices $i, j, k \in \{1, \ldots, n\}$ such that $A_i + B_j + C_k = 0$.

In other words, Colorful-3-SUM is 3-SUM where the three numbers must come one from each of the three lists (think of the lists as having three different colors).

We will actually show something stronger: that the two problems are equivalent up to (n^2, n^2)-reductions, so that if one can be solved in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$, then so can the other.

(a) Show how to (n^2, n^2)-reduce 3-SUM to Colorful-3-SUM. In other words, prove that if Colorful-3-SUM can be solved in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$, then 3-SUM can be solved in time $O(n^{2-\varepsilon'})$ for some $\varepsilon'(\varepsilon) > 0$ (this just means that ε' is a function of ε, and is positive whenever ε is).

Small "Hint"

This is supposed to be the easy direction. If it seems too easy (after you read and understand all the definitions), it's because it actually is.

(b) We will soon show how to (n^2, n^2)-reduce Colorful-3-SUM to 3-SUM, but let's first see why a naive approach does not work.

A first idea for solving Colorful-3-SUM using 3-SUM is to simply concatenate the three lists into a single list of $3n$ integers and run 3-SUM on it. (For this to even be a valid 3-SUM instance, the three lists must not share any integer; let's ignore this issue for now.) Show that this does not work: give a Colorful-3-SUM instance A, B, C that has *no* solution, but whose concatenation *does* have a 3-SUM solution. (No proof is needed here: just give the counterexample, and check that it fails in both ways.)

(c) Show how to (n^2, n^2)-reduce Colorful-3-SUM to 3-SUM. In other words, prove that if 3-SUM can be solved in time $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$, then Colorful-3-SUM can be solved in time $O(n^{2-\varepsilon'})$ for some $\varepsilon'(\varepsilon) > 0$.
