CS330 – Reductions

Purpose: A central tool in computer science is using the solution of one problem to solve another problem. In order to do this, you need to somehow turn the problem you need to solve into an instance of a problem whose solution we have at hand. This is called a "reduction".

Knowledge: This activity will help you become familiar with the following content knowledge:

- How to perform reductions

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 60 minutes.

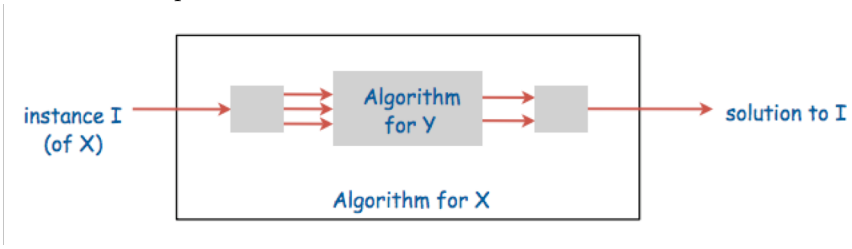
1. Consider the following two problems:

   Problem 1: Given a list of values $X = \{x_1, x_2, ...x_n\}$, return the list in sorted order.

   Problem 2: Given two lists of values $X = \{x_1, x_2, ...x_n\}$ and $Y = \{y_1, y_2, ...y_n\}$, return a pairing of all the elements of X and Y such that the least element of X is paired with the least element of Y and so on.

   Describe how we can use a solution for Problem 1 to solve Problem 2. What is the order of growth of your algorithm?

2. Definition: A problem X reduces to a problem Y if you can use an algorithm that solves Y to help solve X.

   Draw the picture for the previous example and describe exactly what is contained in the two smaller gray boxes. The first small box is the procedure for turning the instance of problem X into an instance of problem Y. The second small box is the procedure for turning the solution for problem Y into the solution for problem X.

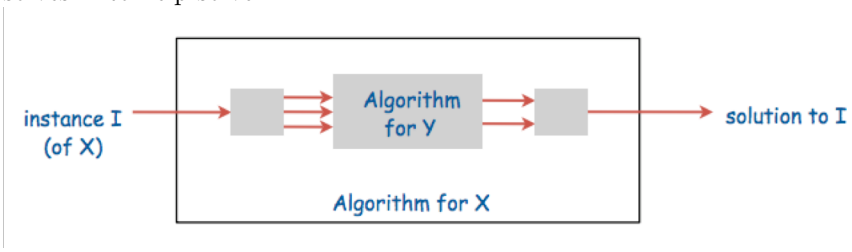
   The cost of solving X would be the cost of the reduction plus the cost of solving Y times the number of times the algorithm for Y is used.

3. A little more challenging is to come up with an algorithm for Problem 1 above that uses Problem 2 in the solution. In other words reduce sorting to pairing.

   Hint: Let $X = \{x_1, x_2, ...x_n\}$ and $Y = \{1, 2, ...n\}$

4. Since this reduction tells us that we can use the pairing algorithm to perform a sort, what does this tell you above the lower bound for pairing? Is it possible to find a O(n) pairing algorithm?

5. Definition: A problem X reduces to a problem Y if you can use an algorithm that solves Y to help solve X.



   If we know that X has a large time complexity and we have this reduction, what does this tell you about the time complexity of Y? Can the time complexity of Y be smaller than the time complexity of X? Why?

6. What about the reverse? If we know that Y has a large time complexity and we have this reduction, does this tell you anything about the time complexity of X? Can X have a smaller time complexity than Y?