(1) **(Best-case time)** (10 points) Show how to take nearly any algorithm (even a poor one), and modify it so that it has good best-case time.

To answer this question, describe a procedure that (1) takes as input a problem statement $\mathcal{P}$ together with an algorithm $\mathcal{A}$ that solves $\mathcal{P}$, and (2) outputs a new algorithm $\mathcal{A}'$ for $\mathcal{P}$ whose best-case running time is "as good as possible." An intelligent human being (such as yourself) should be able to carry out your procedure.

(Note: The original algorithm $\mathcal{A}$ might be very inefficient. So what does this tell you about best-case time?)

If we take the current algorithm and add a base case that checks to see if the given input is already a solution to the problem then simply return the input.

For any algorithm A that solves a problem P we can add a base case that checks for the most trivial case solution to the problem (A') and return the known answer for that particular configuration. If the input to the algorithm doesn't meet the base case to satisfy algorithm A' then continue to use A to solve the problem