Homework #6

Find the ODD Coin

1. You have a large number of coins and a pan balance. You may put any number of coins in each pan of the balance. The balance will tell you if the set of coins in one pan weighs the same as the set of coins in the other pan, or it will tell you which set is heavier.
   Somewhere among the coins is one coin which has different weight than the other coins. All the coins, except this one odd coin, have exactly the same weight.
   The problem is to find the odd coin

2. Design a divide and conquer algorithm to solve this coin problem.
   You may assume that the number of coins is a power of 3.

3. Give a difference equation for the number of weighings used by your algorithm.
   Solve this difference equation.
   Give a difference equation for the run time of your algorithm.
   Solve this equation to “Big Oh” order.

4. To simulate the balance, you should have a routine which takes as input two subranges of integers which indicate the two subsets of coins you want to compare, and your routine should return one of “heavier”, “lighter”, or “equal” depending on the location of the odd coin and its weight relative to the other coins. Your balance routine will need to know about the position and weight of the odd coin, but it can only communicate this information back to your program by use of the three words “heavier”, “lighter”, and “equal”. You should use a random number generator to decide the position and weight of the odd coin.
5. Now run your program for various numbers of coins, but use several different runs for each number of coins. Plot both the running time and the number of calls to balance as functions of the number of coins.

6. Let \( n \) be the number of coins.

   Does your program always take the same time and make the same number of weighings for each problem with \( n \) coins?

   How well does your run time and weighing data fit with the predictions from your difference equations?

**Newton Algorithm**

Newton’s method can be used to design algorithms for a variety of problems like square root and division. Here, we want you to use Newton’s method to design an algorithm which computes the square root of the reciprocal.

**INPUT:** a number \( A \).

**OUTPUT:** a number \( 1 / \sqrt{A} \).

1. Give a function \( f(x) \) which has \( 1 / \sqrt{A} \) as a root, i.e. \( f(1 / \sqrt{A}) = 0 \).

   Notice that \( A \) has to appear somewhere in \( f(x) \).

2. Find a formula for the derivative \( f'(x) \).

3. Use \( f(x) \) and \( f'(x) \) to set up the Newton iteration

\[
   x_{n+1} = N(x_n).
\]

4. If you are given \( A \), what value should you use for \( x_0 \) the starting point for your Newton iteration?

5. Pick a specific value for \( A \) and a value for \( x_0 \) and show the sequence of values \( \langle x_n \rangle \).

6. How quickly is the sequence \( \langle x_n \rangle \) converging? Is this sequence increasing or decreasing?

   Can you find an approximate formula \( g(\_\_\_) \), so that

\[
   x_{n+1} - x_n = g(x_n - x_{n-1})?
\]

   For the calculations in this problem, you can either write a program and execute it, **OR** you can use a calculator.