CSci 243 Homework 7

Due: Wednesday, November 2, end of day
Carlo Mehegan

1. (10 points) What is the largest problem size n that we can solve in no more than **one hour** using an algorithm that requires $f(n)$ operations, where each operation takes 10^{-9} seconds (this is close to a today's computer), with the following $f(n)$?

   (a) $\log_2 n \Rightarrow 2^{3.6\times10^{12}}$

   (b) $\log_2^4 n \Rightarrow 2^{\sqrt[4]{3.6\times10^{12}}}$

   (c) $3n \Rightarrow 1.2\times10^{12}$

   (d) $n\log_2 n$

   (e) $n\log_2^2 n$

   (f) $n^2 \Rightarrow \sqrt{3.6\times10^{12}}$

   (g) $(3n)^3 \Rightarrow \frac{\sqrt[3]{3.6\times10^{12}}}{3}$

   (h) $2^n \Rightarrow \log_2 3.6\times10^{12}$

   (i) $n!$

   (j) n^n

2. (10 points) Use pseudocode to describe an algorithm that determines whether a given function from a finite set to another finite set is one-to-one. Assume that the function $f: A \to B$ is given as a set of pairs $\{(a_i, f(a_i)), \forall a_i \in A = \{a_1, \dots, a_m\}$. You may also assume that you are given $B = \{b_1, \dots, b_n\}$.

```
Algorithm isInjective (f, B)
    previous_values = []
    for each pair in f:
        x = f(a_i) //the second value in the pair
        if x is in B and x is not in previous_values:
            previous_values.append(x)
        else if x is in previous_values and x is in B:
            return false
    return true
```

3. (5 points) Describe an algorithm that produces the maximum, the minimum, and the mean, of a set of n real numbers, passing through the numbers once.

   1. Create variables for *max*, *min*, and *sum*. Take the first element and set it as the initial value of *max*, *min*, and *sum*.

   2. Take the next element and compare it to *min* and *max*. If its value is smaller than *min*'s value, then set the value of *min* to the value of the element. If its value is larger than *max*'s value, set the value of *max* to the value of the element. Increment *sum* by the value of the element.

   3. Repeat step 2 until every element has been passed through.

   4. Return *min* and *max* as the minimum and maximum values of the set. Divide *sum* by *n* and return this as the mean value of the set.

4. (10 points) Consider the following pseudocode that finds x in a list of sorted numbers by using ternary search. The algorithm is simlar to binary search, only it splits the current list into three parts (instead of two) and checks which part x may be in. Thus at each step, the algorithm removes 2/3 of the items in the current list. Find the complexity of the algorithm. Is it faster or slower asymptotically than binary search?

```
Algorithm TernarySearch(x int, a(1)...a(n) int, output: loc int)
i=1;
j=n;
remaining = j-i+1;
while (remaining > 1)
   interval = floor(remaining/3);
   m1 = i+interval;
   m2 = i+2*interval;
   if (x <= a(m1))
      j=m1;
   else if (x<=a(m2))
      i=m1+1;
      j=m2;
   else
      i=m2+1;
   endelsif
   remaining = j-i+1;
endwhile

if (remaining == 1 and x not equal a(i)) return loc=0;
else return loc = j;
```

Complexity of binary search: $\log_2 n$

Complexity of ternary search: $2\log_3 n$

$\log_2 n \Leftrightarrow 2\log_3 n$

$\log_2 n \Leftrightarrow 2\frac{\log 2}{\log 3}\log_2 n$

$1 < 2\frac{\log 2}{\log 3}$

So, ternary search is slower asymptotically than binary search