CSci 243 Homework 6

Due: 10:00 am, Wednesday, October 14
Alexander Powell

1. (10 points) Use pseudocode to describe an algorithm based on the linear search that determines the correct position in which to insert a new element in an already sorted list.

   Solution:

   The following algorithm linearly searches through an inputted array until it finds the correct index to insert the new element. It should be noted that indexing is zero based.

```
Algorith GetInsertionIndex(input: Array array[0]...array[n-1],
                           number to insert x)
index = 0
while (i < n)
   if (x < array[i])
      return i
   i = i + 1
return i
```

2. (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;
```

Solution:

Both binary and ternary searches have logarithmic complexity. In a binary search there are $2 \times \log_2 n + 1$ compares in the worst case and in a ternary search there are $4 \times \log_3 n + 1$ compares in the worst case. Thus, we need to evaluate $\log_2 n$ and $2\log_3 n$. We can write $2\log_3 n$ as $\frac{2}{\log_2 3} \times \log_2 n$, and since $\frac{2}{\log_2 3}$ is greater than 1, we know that ternary search does more comparisons in the worst case.

So, we can define the complexity of ternary search to be $O(\log_3 n)$ and it is asymptotically slower than binary search.

3. For each of the following program fragments, give an analysis of the running time. You may use summations to evaluate the running times of nested loops. Show your work.

   (a) (6 points)

```
sum = 0
for i = 1 to n*n
   for j = 1 to n
      sum ++
```

Solution:

This pseudocode can be rewritten as the following double summation:

$$\sum_{i=1}^{n^2}\sum_{j=1}^{n} 1 = \sum_{i=1}^{n^2} n = n^3$$

Therefore, the runtime complexity of the program can be described as cubic, or order $O(n^3)$.

(b) (6 points)

```
sum = 0
for i = 1 to n
   for j = n-i+1 to n
      sum ++
```

Solution:
The above pseudocode can be represented with the following double summation:

$$\sum_{i=1}^{n}\sum_{i=n-i+1}^{n} 1 = \sum_{i=1}^{n}(n-(n-i+1)+1) = \sum_{i=1}^{n} i$$

We know from the gaussian summation formula, we can write the summation in the closed form:

$$\frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

So the program has a quadratic runtime complexity, or $O(n^2)$.

(c) (8 points)

```
sum = 0
for i = 1 to n
   for j = 1 to i * i
      for k = 1 to j
         sum ++
```

Solution:
We can write the psuedocode with a tripple summation as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{i*i}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i*i} j = \sum_{i=1}^{n}\left(\frac{i^2(i^2+1)}{2}\right)$$

$$\frac{1}{2}\sum_{i=1}^{n}\left(i^4+i^2\right) = \frac{1}{2}\left(\sum_{i=1}^{n} i^4 + \sum_{i=1}^{n} i^2\right)$$

This sum can be written in the closed form below:

$$\frac{1}{2}\left(\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} + \frac{n(n+1)(2n+1)}{6}\right)$$

From expanding this, we know the highest power in the formula will be n^5, so we can say the runtime complexity of the program is $O(n^5)$.