Homework-2

Due Date: 01/31/2017

Problem 1: Find the order of growth of the running time of the following programs

a.

```
int x = 1, i;
for(i = 0; i < N; i++)
  x++;
```

$O(N)$

b.

```
int x = 1, i, j;
for(i = 0; i < N; i++)
  for(j = 1; j < R; j++)
    x = x * j;
```

$O(NR)$.

c.

```
public static int f2(int N)
  int x = 1;
  while(x < N)
    x = x * 2;
  return x;
}
```

$O(\log_2(N))$.

d.

```
int x = 0, i;
for(i = 0; i < N; i++)
  x += f2(N);
```

$O(N\log(N))$

e.

```
int x = 0, i, j;
for(i = 1; i <= N; i++)
  for(j = 1; j <= N+R; j+=i)
    x += j;
```

$$t = 3 + \sum_{i=1}^{N} \left(\frac{N+R}{i}(1+1) \right) = 3 + \sum_{i=1}^{N} \frac{2(N+R)}{i}.$$

$$= 3 + 2(N+R) \sum_{i=1}^{N} < 3 + 2(N+R)\sqrt{N}$$

$$= O((N+R)\sqrt{N}) = O(\max\{N,R\}\sqrt{N}).$$

$$\approx O(N\sqrt{N}).$$

f.

```
public static int f7(int N) {
   if (N == 1) return 0;
   return 1 + f7(N/2);
}
```

$\log_2(N)$.

So. $c = f < a < d < e < b$

Problem 2: Find the order of growth of the running time of the following programs

a.

```
public static int f1(int N) {
    int x = 0;
    for (int i = 0; i < N; i++)
        x++;
    return x;
}
```

$O(N)$

b.

```
public static int f2(int N, int R) {
    int x = 0;
    for (int i = 0; i < R; i++)
       x += f1(i);
    return x;
}
```

$O(NR)$

c.

```
public static int f3(int N, int R) {
    int x = 0;
    for (int i = 0; i < R; i++)  R
       for (int j = 0; j < N; j++)  N
          x += f1(j);  N
    return x;
}
```

$O(N^2R)$

d.

```
public static int f4(int N, int R) {
    int x = 0;
    for (int i = 0; i < N; i++)  N
       for (int j = 1; j <= R; j += j)  log R
          x++;
    return x;
}
```

$O(N\log_2 R)$

e.

```
public static int f5(int N, int R) {
    int x = 0;
    for (int i = 0; i < N; i++)  N
       for (int j = 1; j <= R; j += j)
          x += f1(j);
    return x;
}
```

$$\sum_{j=1}^{\log_2 R} 2^j = 1+2+4+8\cdots R$$

$$= \frac{1(1-2^{(\log_2 R)})}{1-2} = 2^{\log_2 R}-1$$

$$= R-1$$

$O(N(R-1)) = O(NR)$

$a < d < b = e < c$

Problem 3: Find the order of growth of the running time of the following programs $x(n-1)$ for $f3(n-1)$

suppose need time $x(n)$ for $f3(n)$.

a.

```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++) N.
      x += f3(N-1); x(N-1)
   return x;
}
```

then: $x(n) = n \cdot x(n-1)$. $\Rightarrow \frac{x(n)}{x(n-1)} = n$.

and $x(1) = 1$.

$\Rightarrow x(n) = n!$

b.

```
public static int f6(int N) {
   if (N == 0) return 1;
   return f6(N-1) + f6(N-1) + f6(N-1);
}
```

suppose need time $x(n)$ for $f6(n)$

then: $x(n) = 3x(N-1)$. $\Rightarrow \frac{x(n)}{x(n-1)} = 3$.

and $x(0) = 1$.

So: $x(n) = 1 \cdot 3^{n-1} \Rightarrow O(3^n)$

c.

```
public static int f7(int N) {
   int x = 0;
   while (N > 0) {
      x++;
      N = N / 2;
   }
   return x;
}
```

This will cause dead loop in [illegible] O(

in other language: $O(\log_2 N)$.

d.

```
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n/2);
 }
```

$O(\log_2 N)$

e.

```
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n-1);
}
```

$O(N)$.

f.

```
void silly(int n, int x, int y) {
   for (int i = 0; i < n; ++i) {   n.
      if (x < y)
         for (int k = 0; k < n * n; ++k){  n^2.
             System.out.println("k = " + k);
         }
      else
         System.out.println("i = " + i);
   }
}
```

$O(n\cdot n^2) = O(n^3)$.

g.

```
void silly(int n) {
   for (int i = 0; i < n; ++i) {
      for (int j = 0; j < i; ++j) {   i
         System.out.println("j = " + j);
      }
      for (int k = 0; k < n * 3; ++k) { 3n.
         System.out.println("k = " + k);
      }
    }
 }
```

Since we have $i < n < 3n$.

So: $O(n\cdot(3n)) = O(3n^2)$

$= O(n^2)$.

i

```
void sunny(int n, int x) {
     for (int k = 0; k < n; ++k)   n
        if (x < 50) {
           for (int i = 0; i < n; ++i)  n      ]
              for (int j = 0; j < i; ++j) i     }  n^2/2.
                 System.out.println("x = " + x); }
        } else {
           System.out.println("x = " + x);   1
        }
 }
```

$O(\frac{n^2}{2}\cdot n) = O(n^3$

j.

```
void warm(int n) {
     for (int i = 0; i < 2 * n; ++i) {  2n.
        j = 0;
        while (j < n) {
           System.out.println("j = " + j);  n/5
           j = j + 5;
        }
     }
 }
```

$O(2n\cdot\frac{n}{5}) = O(\frac{2n^3}{5}) = O(n^3)$

k.

```
int silly(int n, int m) {
    if (n < 1) return m;
    else if (n < 10)
      return silly(n/2, m);
    else
      return silly(n - 2, m);
}
```

$O(1)$

$O(n)$

$O(n^2)$

$O(n^2)$

l.

```
void happy(int n) {
    for (int i = n*n; i > 0; i--) {
       for (int k = 0; k < n; ++k)
          System.out.println("k = " + k);
       for (int j = 0; j < i; ++j)
         System.out.println("j = " + j);
       for (int m = 0; m < 5000; ++m)
         System.out.println("m = " + m);
    }
  }
```

$n^2.$ $\quad i = n^2 > n.$

$n.$ $\quad O(n)$

$i = n^2.$ $\quad O(n^2)$

$5000.$ $\quad O(1).$

$O(n^2 \cdot n^2) = O(n^4)$

so: $d = c < e < g = k < f < i < j < l < b < a$

$\log n \quad \log n \quad n \quad n^2 \quad n^2 \quad n^3 \quad n^3 \quad n^3. \quad n^4 \quad 3^n \quad n!$

Problem 4. Programming: Submit python files of solutions of the following problems.

a. Devise an experiment to verify that the list index operator is O(1)

b. Devise an experiment to verify that get item and set item are O(1) for dictionaries

c. Devise an experiment that compares the performance of the `del` operator on lists and dictionaries.

d. Given a list of numbers in random order, write a algorithm that works in O(nlog(n)) to find the kth smallest number in the list