Tutorial 1: Complexity

Artificial Intelligence

G.Guérard

SOME SOLUTIONS MAY HAVE FALSE CLUES – SO BE CAREFUL

Exercise 1

Determine the time complexity of the Check algorithm:

```
bool Check(int Mat[3][3]) {
        bool Tab[9] = { FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE } ;
        for(int i = 0; i < 3; i + +) {
                for(intj = 0; j < 3; j + +) {
                        int value = Mat[i][j] ;
                        Tab[value-1] = TRUE ;
                }
        }
        for(int i = 0; i < 9; i + +) {
                if( !Tab[i]) {
                        return FALSE ;
                }
        }
        return TRUE ;
}
```

Solution

Check correctness and completeness, especially what happened during the first iteration.

Exercise 2

Analyze the complexity of Algorithm. Write another algorithm that does exactly the same thing as Algorithm but with a strictly better asymptotic time complexity.

```
int Algorithm(Array A<int>){
        int i = 1
        int j = 1
        int m = 0
        int c = 0
        while (i <= |A|){
                if (A[i] == A[j])
                        c = c + 1
```

```
                j = j + 1
                if (j > |A|){
                        if (c > m)
                                m = c
                        c = 0
                        i = i + 1
                        j = i
                }
        }
        return m
}
```

Solution

Complexity is O(|A|²). You can improve the algorithm by: first, sort the table in O(n log n) and then check occurrences in O(n), thus complexity becomes O(n log n).

Exercise 3

What is the (asymptotic) running time of each of the following algorithms, as a function of n? Justify your answers.

a)

```
for i = 1 to n do
        for j = 1 to 2n + 1 do
                print ("Hello World)
        end for
end for
```

b)

```
for i = 1 to 10 do
        for j = 1 to n do
                print ("Hello World")
        end for
end for
```

c)

```
for i = 1 to n do
        for j = i to n do
                print ("Hello World")
        end for
end for
```

d)

```
for i = 1 to n do
        for j = 1 to 2 ∗ i + 1 do
                print ("Hello World")
        end for
end for
```

e)

```
for i = 1 to n ∗ n do
        for j = 1 to i do
                print ("Hello World")
        end for
end for
```

f)

```
for (i = 0 to m) do
        t ← 1
        while (t < m) do
                print ("Hello world")
                t ← t ∗ 2
        end while
end for
```

Solution

a) $O(n^2)$
b) $O(n)$
c) $O(n^2)$
d) $O(n^2)$
e) $O(n^4)$
f) $O(m \log(m))$.

Exercise 4

Let $T_A(n)$, $T_B(n)$ and $T_C(n)$ denote the running time of algorithms A, B and C, respectively.

```
Instruction A
if (n < 100) then
        Instruction B
else
        for (j = 1 to n) do
                Instruction C
        end for
end if
```

What is the running time of this algorithm, under each of the following sets of assumptions? Justify your answers.

a) $T_A(n) = O(n)$, $T_B(n) = O(n^2)$ and $T_C(n) = O(\log n)$.
b) $T_A(n) = O(n^2)$, $T_B(n) = O(n^2)$ and $T_C(n) = O(\log n)$.
c) $T_A(n) = O(n^2)$, $T_B(n) = O(n^3)$ and $T_C(n) = O(\log n)$.

Solution

Since large values of n determine the asymptotic running time, we can neglect the if part of the if statement. Therefore, in all cases, the running time of this algorithm is O(TA(n) + nTC(n)).

a) $O(n \log n)$
b) $O(n^2)$
c) $O(n^2)$

Exercise 5

What is the worst-case complexity of the each of the following code fragments?

Two loops in a row:

```
1. for (i = 0; i < N; i++) {
2.     sequence of statements
3. }
4. for (j = 0; j < M; j++) {
5.     sequence of statements
6. }
```

How would the complexity change if the second loop went to N instead of M?

A nested loop followed by a non-nested loop:

```
7. for (i = 0; i < N; i++) {
8.     for (j = 0; j < N; j++) {
9.         sequence of statements
10.        }
11.    }
1. for (k = 0; k < N; k++) {
12.    sequence of statements}
```

A nested loop in which the number of times the inner loop executes depends on the value of the outer loop index:

```
13.    for (i = 0; i < N; i++) {
14.        for (j = N; j > i; j--) {
15.            sequence of statements
16.        }
   }
```

Solution

1. The first loop is O(N) and the second loop is O(M). Since you don't know which is bigger, you say this is O(N+M). This can also be written as O(max(N,M)). In the case where the second loop goes to N instead of M the complexity is O(N). You can see this from either expression above. O(N+M) becomes O(2N) and when you drop the constant it is O(N). O(max(N,M)) becomes O(max(N,N)) which is O(N).
2. The first set of nested loops is $O(N^2)$ and the second loop is O(N). This is $O(\max(N^2,N))$ which is $O(N^2)$.
3. This is very similar to our earlier example of a nested loop where the number of iterations of the inner loop depends on the value of the index of the outer loop. The only difference is that in this example the inner-loop index is counting down from N to i+1. It is still the case that the inner loop executes N times, then N-1, then N-2, etc, so the total number of times the innermost "sequence of statements" executes is $O(N^2)$.

Exercise 6

1. Give an analysis of the running time (Big-Oh notation) for each of the following 4 program fragments. Note that the running time corresponds here to the number of times the operation sum++ is executed. sqrt is the function that returns the square root of a given number.

(a)
```
sum = 0;
for(i=0;i<sqrt(n)/2;i++)
 sum++;
for(j=0 ;j<sqrt(n)/4;j++)
 sum++;
for(k=0;k<8+j;k++)
 sum++;
```

(b)
```
sum = 0;
for(i=0;i<sqrt(n)/2;i++)
  for(j=i;8+i;j++)
   for(k=j;k<8+j;k++)
    sum++;
```

(c)
```
sum = 0;
for(i=1;i<2*n;i++)
 for(j=1;j<i*i;j++)
   for(k=1;k<j;k++)
     if (j % i == 1)
       sum++;
```

(d)
```
sum = 0;
for(i=1;i<2*n;i++)
 for(j=1;j<i*i;j++)
   for(k=1;k<j;k++)
     if (j % i)
       sum++;
```

2. If it takes 10ms to run program (b) for n=100, how long will it take to run for n=400 ?

3. If it takes 10ms to run program (a) for n=100, how large a problem can be solved in 40ms ?

Solution

A and b O(sqrt(n)), c and d in O(n^5)

With cross product: sqrt(100)=x and y=10ms; sqrt(400)=2x take 20ms

In 40ms, a can run n=1600.