Algorithm Analysis

Suzan Ece Ada

Code Complexity

Primitive Operations:

- Addition etc.
  - 1 operation
- Calling a method or returning from a method
  - 1 operation
- Index in an array
  - 1 operation
- Comparison
  - 1 operation

[3]

Code Complexity

Loops:

- FOR
  - (# of iterations) * (running time of statements inside the loop)
- Nested loops
  - product of loop complexities
- Consecutive Statements
  - The sum of running time of each segment
- if/Else
  - The testing time + Larger of the running time of the cases

Q1 (Big-Oh)

```
bool checkForOddOrEven(int x{
if (x%2)
    print("Odd");
else
    print("Even");
}
```

Q2 (Big-Oh)

```
int findMax(int[] x){
 int max=0;
 for(int i=0;i<x.length;i++){
    if(x[i] > max)
        max = x[I];
 }
return max;
}
```

Q3 (Big-Oh)

```
bool checkForDuplicates(int[] x){
for(int i=0;i<x.length;i++){
     for(int j=0;j<x.length;j++){
          if (i == j) continue;
          if(x[i] == x[j])
               return true;
     }
}
return false;
}
```

[2]

Q4 (Big-Oh)

```
int calculateStrangeSum(int[] x){
 int h=x.length;
 int sum = 0;
 while(h>0){
    sum += x[h];
    h = h/2; }
 return sum;
}
```

[2]

Q5 (Big-Oh)

```
int mixed(int[] x){
print(x.length);
for(int i=0;i<x.length;i++){
    for(int j=0;j<x.length;j++){
        if (i == j) continue;
        if(x[i] == x[j])
            print("Duplicate");
}}
}
```

Q6

Order the following functions by growth rate:

$$N,\ \sqrt{N},\ N^{1.5},\ N^2,\ N\log N,$$

$$N\log\log N,\ N\log^2 N,\ N\log(N^2),\ 2/N,\ 2^N,\ 2^{N/2},\ 37,\ N^2\log N,\ N^3.$$

Indicate which functions grow at the same rate.

[1]

Q7

Suppose $T_1(N) = O(f(N))$ and $T_2(N) = O(f(N))$. Which of the following are true?

a. $T_1(N) + T_2(N) = O(f(N))$

b. $T_1(N) - T_2(N) = o(f(N))$

c. $T_1(N) / T_2(N) = O(1)$

d. $T_1(N) = O(T_2(N))$

Q8

Give an analysis of the running time (Big-Oh will do).

```
(1)  sum = 0;
     for( i = 0; i < n; i++ )
         sum++;

(2)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
             sum++;

(3)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n * n; j++ )
             sum++;

(4)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
             sum++;

(5)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i * i; j++ )
             for( k = 0; k < j; k++ )
                 sum++;

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

[1]

Q9

Implement the code in Java, and give the running time for several values of N.

```
(1)  sum = 0;
     for( i = 0; i < n; i++ )
         sum++;

(2)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
             sum++;

(3)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n * n; j++ )
             sum++;

(4)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
             sum++;

(5)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i * i; j++ )
             for( k = 0; k < j; k++ )
                 sum++;

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

[1]

Q10

Compare your analysis with the actual running times.

```
(1)  sum = 0;
     for( i = 0; i < n; i++ )
         sum++;

(2)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
             sum++;

(3)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < n * n; j++ )
             sum++;

(4)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
             sum++;

(5)  sum = 0;
     for( i = 0; i < n; i++ )
         for( j = 0; j < i * i; j++ )
             for( k = 0; k < j; k++ )
                 sum++;

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

[1]

Q11

Programs A and B are analyzed and found to have worst-case running times no greater than $150N \log_2 N$ and N^2, respectively. Answer the following questions, if possible:

a. Which program has the better guarantee on the running time

- for large values of N ($N > 10{,}000$)?
- for small values of N($N<100$)?

b. Which program will run faster on average for $N = 1{,}000$?

[1]

Q12 Maximum Subsequence Sum Problem

Given (possibly negative) integers A_1 , A_2 , . . . , A_N , find the maximum value of k=i A_k . (For convenience, the maximum subsequence sum is 0 if all the integers are negative.) [1]

Example:

For input −2, 11, −4, 13, −5, −2 ?

[1]

Maximum Subsequence Sum Problem

Input Size	Algorithm Time 1 $O(N^3)$	2 $O(N^2)$	3 $O(N \log N)$	4 $O(N)$
$N = 100$	0.000159	0.000006	0.000005	0.000002
$N = 1{,}000$	0.095857	0.000371	0.000060	0.000022
$N = 10{,}000$	86.67	0.033322	0.000619	0.000222
$N = 100{,}000$	NA	3.33	0.006700	0.002205
$N = 1{,}000{,}000$	NA	NA	0.074870	0.022711

Figure 2.2 Running times of several algorithms for maximum subsequence sum (in seconds)

[1]

Maximum Subsequence Sum Problem
Cubic Maximum Contiguous Subsequence Sum

```
public static int maxSubSum1( int [ ] a )

{

int maxSum = 0;

/**

*TODO

*/

return maxSum;

}
```

[1]

Maximum Subsequence Sum Problem
Quadratic Maximum Contiguous Subsequence Sum

```
public static int maxSubSum2( int [ ] a )
{
int maxSum = 0;
/**
*TODO
*/
return maxSum;
}
```

[1]

Maximum Subsequence Sum Problem
Recursive Maximum Contiguous Subsequence Sum

```
private static int maxSumRec( int [ ] a, int left, int right ){

/**

*TODO

*/

return max3( maxLeftSum, maxRightSum,maxLeftBorderSum + maxRightBorderSum );

}

public static int maxSubSum3( int [ ] a ){

return maxSumRec( a, 0, a.length - 1 );

}
```

[1]

Maximum Subsequence Sum Problem
Linear Time Maximum Contiguous Subsequence Sum

```
public static int maxSubSum4( int [ ] a )

{

int maxSum = 0; thisSum=0;

/**

*TODO

*/

return maxSum;

}
```

[1]

References

[1]Mark Allen Weiss. 2012. Data Structures and Algorithm Analysis in Java. Mark Allen Weiss. Pearson Education.

[2]Ozlem Simsek, CMPE250 Algorithm Analysis Slides

[2]CMPE250 Algorithm Analysis Lecture Slides