Practical Exercise 10 – Algorithms Analysis and Big-O Notation

Overall Objective

To estimate algorithm efficiency using the Big-O notation and determine the complexity of various types of algorithms.

Description

1. Why is a constant factor ignored in the Big-O notation? Refer to Ch24 slide 17

The Big *O* notation estimates the execution time of an algorithm in relation to the input size.

If the time is not related to the input size, the algorithm is said to take *constant time* with the notation *O(1)*.

For example, a method that retrieves an element at a given index in an array takes constant time, because it does not grow as the size of the array increases.

2. Why is a non-dominating term ignored in the Big-O notation? Refer to Ch24 slide 8

Consider the algorithm for finding the maximum number in an array of *n* elements.

If *n* is 2, it takes one comparison to find the maximum number.

If *n* is 3, it takes two comparisons to find the maximum number.

In general, it takes *n-1* times of comparisons to find maximum number in a list of *n* elements.

Algorithm analysis is for large input size. If the input size is small, there is no significance to estimate an algorithm's efficiency.

As *n* grows larger, the *n* part in the expression *n-1* dominates the complexity.

The Big *O* notation allows you to ignore the non-dominating part (e.g., -1 in the expression *n-1*) and highlight the important part (e.g., *n* in the expression *n-1*).

So, the complexity of this algorithm is *O(n)*.

3. What is the order of each of the following functions? Refer to Additional slide 21&22

   a. $\frac{(n^2+1)^2}{n} = \frac{(n^4+2n^2+1)}{n} = n^3+2n+\frac{1}{n} = n^3 = O(n^3)$
   (because n cube grow fastest)

   b. $\frac{(n^2+\log_2 n)^2}{n} = \frac{n^4+2n\log_2 n+(\log_2 n)^2}{n} = n^3 = O(n^3)$
   (because n cube grow fastest)

   c. $n^3+100n^2+n = n^3 = O(n^3)$ (because n cube grow fastest)

   d. $2^n+100n^2+45n = 2^n = O(2^n)$ (because 2 power n grow fastest)

   e. $n2^n+n^2 2^n = n^2 2^n = O(n^2 2^n)$ (because n square 2 power n grow fastest)

   f. $3n^4+2n^2+10000 = 3n^4 = O(n^4)$ (**NOTE:** *the 3 need to be omitted*)

4. What is the number of iterations in the following loop?

```
int count = 5;
while (count < n) {
   count = count + 3;
}
```

Formula for loop

$$= \frac{n - startValue}{step}$$

$$= \frac{n-5}{3}$$

$$= n$$

$$= O(n)$$

```
int sum = 1 + 2 + 3 … + 100
```

Answer $= O(1)$
Because it is only calculated once.

Always pay attention on the loop. If without loop is always O(1).

5. Use the Big-O notation to estimate the time complexity of the following methods:

   a.
```
public static void method1(int n) {
     for(int i = 0; i < n; i += 2) {
          System.out.print(Math.random() + " ");
     }
}
```

Refer to Ch24 slide 11 or Additional slide 13, 21&22

Answer

$= \frac{n - startValue}{step}$

$= \frac{n - 0}{2}$

$= \frac{n}{2}$

$= O(n)$

b.
```
public static void method2(int n) {
    for(int i = 0; i <= n; i++) {
        for(int j = 0; j < i; j++) {
            System.out.print(Math.random() + " ");
        }
    }
}
```
Refer to Ch24 slide 13 or Additional slide 19, 20, 21&22

Answer

$= outerLoop \times innerLoop$

$= \frac{n-0}{1} \times \frac{n-0}{1}$ (for inner loop take the worst case, which is n)

$= n^2$

$= O(n^2)$

c.
```
public static void method3(int[] m) {
      for(int i = 0; i < m.length; i += 2) {
           System.out.print(m[i] + " ");
      }

      for(int i = m.length - 1; i >= 0; i -= 2) {
           System.out.print(m[i] + " ");
      }
}
```
Refer to Ch24 slide 15

Answer

$= loop_1 + loop_2$

$= \frac{n-0}{2} + \frac{n-0}{2}$

$= \frac{n}{2} + \frac{n}{2}$

$= n$

$= O(n)$

6. Put the following growth functions in order:

$$\frac{5n^3}{4032},\quad 44\log n,\quad 500,\quad \frac{2^n}{45},\quad 10n\log n,\quad 2n^2,\quad 3n$$

Refer to Ch24 slide 30 or Additional slide 25

Answer:

$$500 \;\text{->}\; 44\log n \;\text{->}\; 3n \;\text{->}\; 10n\log n \;\text{->}\; 2n^2 \;\text{->}\; \frac{5n^3}{4032} \;\text{->}\; \frac{2^n}{45}$$

7. Design/describe an algorithm for the following tasks, and analyse the time complexity of the algorithm.

   a. Compute the sum of all numbers from n1 to n2 for (n1 < n2).

      First method(use loop):
      ```
      sum = 0
      for(int i = n1; i <= n2; i++) sum += i
      ```
      Complexity $= O(n)$

      Second method (use formula):
      $$sum(n_1, n_2) = \frac{n_2(n_2 - 1)}{2} - \frac{n_1(n_1 - 1)}{2}$$
      Complexity $= O(1)$

b. Find the occurrence of the largest element in an array.

```
largest = xs[0]
for(i=1 ; i < xs.length ; i++)
      if(largest < xs[i])
            largest = xs[i]

occurrence = 0
for(i=0 ; i < xs.length ; i++)
      if(xs[i] == largest)
            occurrence++
```

Complexity = $n + n = 2n = O(n)$

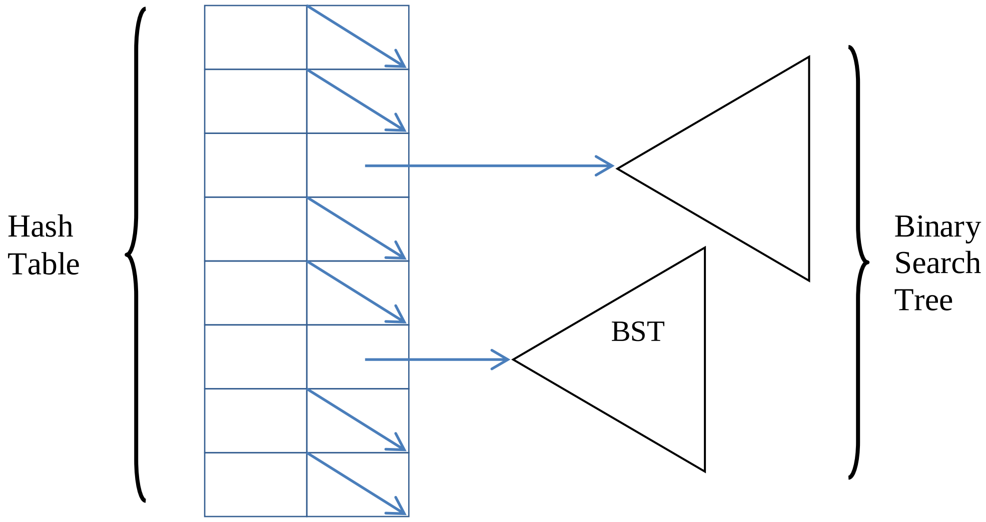
c. Remove duplicate element in an array.

```
for(i=0; i < array.length; i++)
   for(j=i; j < array.length; j++)
         if(array[j] == array[i])
               remove(array[j])
```

Complexity = $n \times n = n^2 = O(n^2)$

8. Give a diagram of conceptual design of a student database of N records that supports the following search-time constraints:

Worst case search-time:	$O(\log(N))$
Best and average search-time:	$O(1)$



Hash table of N entries

9. What is dynamic programming? Give an example. (refer to recursive and non-recursive Fibonacci solutions)
   Refer to Ch24 slide 36