Data Structures and Algorithms

HUS - HKI (2023 - 2024)

Assignment 1

Lecturer: Vũ Đức Minh - Trần Bá Tuấn

§ Algorithm Analysis §

(1) Trình bày sơ lược về độ phức tạp của thuật toán.

(2) Một số ví dụ

Example 1. *Find upper bound for:*

a) $f(n) = 4n + 8$

b) $f(n) = 3n^2 + 2$

c) $f(n) = n^4 + 100n^2 + 80$

d) $f(n) = 5n^3 - 5n^2$

e) $f(n) = 508$

Câu hỏi chung là: Độ phức tạp của thuật toán là gì?

Example 2. *Tính* $S = \frac{n * (n-1)}{2}$

Example 3.

```
for i in range(0, n):
    print('Current Number', i)
```

Example 4.

```
for i in range(0, n):
    print('Current Number', i)
    break
```

Example 5.

```
def function(n):
    i = 1
    while i <= n:
        i = i * 2
        print(i)
function(100)
```

Example 6.

```
for i in range(0, n):
    for j in range(0, n):
        print("Value of i, j", i, j)
```

Example 7.

```
public void function(int n){
        int i, j, k, count = 0;
        for(i = n/2; i <= n; i++)
            for(j = 1; j + n/2 <= n; j++)
                for(k = 1; k <= n; k = k * 2)
                    count++;
 }
```

Example 8.

```
public void function(int n) {
        int i, j, k, count = 0;
        for(i = n/2; i <= n; i++)
            for(j = 1; j <= n; j = 2 * j)
                for(k = 1; k <= n; k = k * 2)
                    count++;
}
```

(3) Bài tập vận dụng

Exercise 1. *Độ phức tạp của thuật toán là gì?*

a) $T(n) = nlogn + 3n + 2$

b) $T(n) = nlog(n!) + 5n^2 + 7$

c) $T(n) = 1000n + 0.01n^2$

d) $T(n) = 100nlogn + n^3 + 100n$

e) $T(n) = 0.01nlogn + n(logn)^2$

Exercise 2. *Độ phức tạp của thuật toán các đoạn code dưới đây là gì?*

a)

```
// Returns the sum of the integers in given array.
 public static int example1(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        total += arr[j];
    return total;
 }
```

b)

```
// Returns the sum of the integers with even index in given array.
 public static int example2(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j += 2) // note the increment of 2
        total += arr[j];
    return total;
 }
```

c)

```
// Returns the sum of the prefix sums of given array.
 public static int example3(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        for (int k=0; k <= j; k++) // loop from 0 to j
            total += arr[j];
    return total;
 }
```

d)

```
// Returns the sum of the prefix sums of given array.
 public static int example4(int[ ] arr) {
     int n = arr.length, prefix = 0, total = 0;
     for (int j=0; j < n; j++) { // loop from 0 to n-1
          prefix += arr[j];
          total += prefix;
     }
     return total;
 }
```

e)

```
 // Returns the number of times second array stores sum of prefix sums from
  ↪ first.
 public static int example5(int[ ] first, int[ ] second) {
 // assume equal-length arrays
     int n = first.length, count = 0;
     for (int i=0; i < n; i++) { // loop from 0 to n-1
          int total = 0;
          for (int j=0; j < n; j++) // loop from 0 to n-1
              for (int k=0; k <= j; k++) // loop from 0 to j
                  total += first[k];
          if (second[i] == total) count++;
     }
     return count;
}
```