Algorithms & Data Structures
SI3 - Polytech Nice-Sophia - Edition 2018

Lab #1: Proof by Induction, algorithm analysis, recursion

This lab will give you practice about inductive proofs, algorithm analysis and recursion

Part 1: Proof by induction

Prove (by induction) the following formulas:

a. $\sum_{i=1}^{N}(2i-1) = N^2$
b. $\sum_{i=1}^{N} i^3 = \left(\sum_{i=1}^{N} i\right)^2$

Part 2: Function growth

Order the following functions by growth rate. Indicate which functions grow at the same rate:

$N,\ \sqrt{N},\ N^{1.5},\ N^2,\ N\log N,\ N\log\log N,\ N\log^2 N,\ N\log(N^2),\ 2/N,\ 2^N,\ 2^{N/2},\ 37,\ N^2\log N,\ N^3.$

Part 3: Running time complexity

For each of the following six program fragments give an analysis of the running time:

```
(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++;
```

Part 4: Complexity growth

An algorithm takes 0.5 ms for input size 100. How long will it take for input size 500 if the running time is the following (assume low-order terms are negligible):

a. linear
b. $O(N \log N)$
c. quadratic
d. cubic

Part 5: Recursion

In this part, you will practice some recursive functions. You have to write the following methods:

- `binary`: this method takes an integer `N` as input, and prints out all the binary words of length `N`. For example, `binary(3)` will print out `000 001 010 011 100 101 110 111`
- `words`: this method takes two integers `x` and `y` as input, and prints out all the words made of `x` letters `'A'` and `y` letters `'B'`. For example, `words(2,3)` will print out `AABBB ABABB ABBAB ABBBA BAABB BABAB BABBA BBAAB BBABA BBBAA`
- `permutations`: this method takes an integer `n` as input, and prints out all permutation of `( 1, 2, .., n )`. For example, `permutations(3)` will print out `(1,2,3) (1,3,2) (2,1,3) (2,3,1) (3,1,2) (3,2,1)`
- `sum`: this method takes an array `A` of positive integers and an integer `N`, and return a boolean value. The method returns `true` if `N` can be computed by adding some (or all) of the values inside the array. For example, if the array `A` is `[3, 5, 7, 11]` and `N` is `21`, then `sum(A,N)` returns `true` (because 3 + 7 + 11 = 21) but `sum(A,13)` returns `false`. Each value inside the array can be used at most once in the sum.

For each methods give the running time complexity.

Supporting file

- Lab1.py