Exercises - Temporal Complexity

TC1031 - Data structures and fundamental algorithms
Tecnologico de Monterrey, campus Santa Fe
Instructor: Dr. Leonardo Chang

August 17, 2020

Exercise 1

Which of the following functions are $O(n^2)$? No explanation is required, but you might want to prove your answer to yourself to convince yourself that you are correct.

a) $f_1(n) = \frac{1}{20}n^3 + 4n^2$

b) $f_2(n) = 2n^2 + 13n$

c) $f_3(n) = 7n + 30 \log n$

d) $f_4(n) = \sin(n) + 15$

e) $f_5(n) = n \log(n)$

f) $f_6(n) = 2^{2^{10}}$

g) $f_7(n) = 2^n$

Exercise 2

Which of the following functions are $\Omega(n^2)$? No explanation is required, but you might want to prove your answer to yourself to convince yourself that you are correct.

a) $f_1(n) = \frac{1}{50}n^3 + n^2$

b) $f_2(n) = 8n^2 + 6n$

c) $f_3(n) = 7n + 2\log n$

d) $f_4(n) = n^2(\sin(n) + 3)$

e) $f_5(n) = n^2 \log(n)$

f) $f_6(n) = n!$

g) $f_7(n) = n^2(\frac{1}{\log n})$

Exercise 3

You are handed a scenic black-and-white photo of the skyline of a city. The photo is n-pixels tall and m-pixels wide, and in the photo, buildings appear as black (pixel value 0) and sky background appears as white (pixel value 1). In any column, all the black pixels are below all the white pixels. In this problem, you will design an efficient algorithm that finds the location of a tallest building in the photo.

The input is an $n \times m$ matrix, where the buildings are represented with 0s, and the sky is represented by 1s. The output is an integer representing the location of a tallest building. For example, for the input 5×6 matrix below, a tallest building has height 5 and is in location 3 (assuming we are 0-indexing). Thus the output is 3.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

a) Find and implement (in C++) an algorithm that finds a tallest building in time $O(m \log n)$.

b) Find and implement (in C++) an algorithm that finds a tallest building in time $O(m + n)$.

Note: Add as a comment block in your code a brief justification of the runtime of each algorithm.