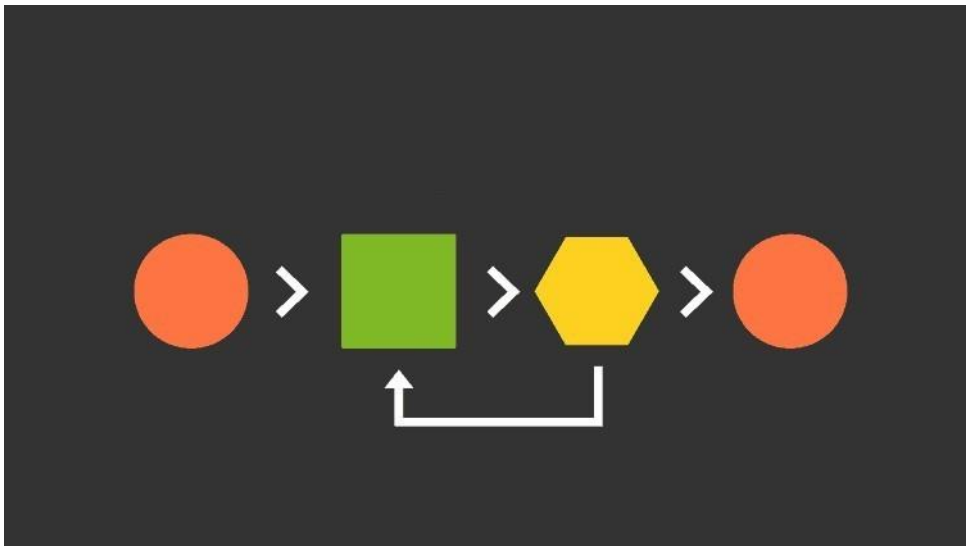

Suma da subsecuencia máxima

Lucía Pérez Rego, Gonzalo Feijóo Fernández, Eliseo Bao Souto

Algoritmos, 2º

Grupo 3.1

10/10/19

Introdución

Imos realizar un programa que calcula a **suma da subsecuencia máxima** dun vector implementando **dous algoritmos** diferentes. O primeiro deles é **cuadrático**, mentres que o segundo é **lineal**. Ademais, creamos dous tests, de xeito que un comprobe a implementación dos algoritmos con valores xa definidos e o outro con valores aleatorios.

A continuación mediremos os **tempos de execución** aplicando os algoritmos a uns vectores de n valores aleatorios, incrementando n en cada medida. Por último, realizamos unha comprobación empírica para as **cota subestimada, a exacta e a sobre-estimada** na que utilizaremos os datos anteriormente obtidos.

Datos

Para a análise dos tempos de execución dos algoritmos temos que definir **varias cotas**: a subestimada, a exacta e a sobre-estimada. Ao ter implementado dous algoritmos, estes empregan cotas diferentes polo que teremos que definilas para cada un deles.

A cota subestimada é aquela función que se aproxima por **debaixo** da función do algoritmo e que polo tanto **tende a infinito**, ó contrario do que pasa coa sobre-estimada, posto que esta se aproxima por enriba e tende a cero. Por último, a cota exacta é aquela que aproxima o seu valor ao do algoritmo.

Algoritmo 1: Podemos afirmar que este algoritmo é de tipo **cuadrático**, xa que a cota exacta segue esta forma.

Empregamos as cotas: *f(n) = n^1.8*, *f(n) = n^2* e *f(n) = n^2.2*.

Algoritmo 2: Nesta ocasión, vemos que o algoritmo segue a forma **lineal** da súa cota exacta.

Empregamos as cotas: f(n) = n^0.8, f(n) = n e f(n) = n^1.2.

Para o primeiro algoritmo a **constante** resultado da operación t(n)/f(n) para a función f(n) da cota exacta ten un valor de **0.001**, sendo este valor de **0.003** para o segundo algoritmo. En ambos casos se producen valores anómalos, isto é, valores que se distancian de maneira moi significativa dos valores anteriormente acotados. Isto é debido a que a máquina non utiliza todos os recursos nas primeiras execucións e polo tanto intentamos non considerar esas medicións.

As medidas dos tempo realízanse en **microsegundos**.

n	T(n)	$T(n)/n^{1,8}$	$T(n)/n^{2}$	$T(n)/n^{2,2}$
500	**270.996000**	0.003757	0.001084	0.000313
1000	1060.000000	0.004220	0.001060	0.000266
2000	4196.000000	0.004797	0.001049	0.000229
4000	16661.000000	0.005470	0.001041	0.000198
8000	68003.000000	0.006412	0.001063	0.000176
16000	269264.000000	0.007291	0.001052	0.000152
32000	1076505.000000	0.008370	0.001051	0.000132

n	T(n)	$T(n)/n^{0,8}$	$T(n)/n^{1}$	$T(n)/n^{1,2}$
500	**1.801000**	0.012484	0.003602	0.001039
1000	**3.201000**	0.012743	0.003201	0.000804
2000	**5.879000**	0.013442	0.002939	0.000643
4000	**11.013000**	0.014463	0.002753	0.000524
8000	**25.061000**	0.018903	0.003133	0.000524
16000	**41.431000**	0.017949	0.002589	0.000374
32000	**81.629000**	0.020311	0.002551	0.000320

Os tempos en negriña foron medidos en 1000 iteracións posto que eran menores de 500 microseg.

Máquina

- MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
- Procesador 2,3 GHz Intel Core i5
- Memoria 8 GB 2133 MHz LPDDR3
-

Conclusións

Na primeira parte da práctica observamos que os dous algoritmos son **equivalentes a efectos de resultados obtidos**, xa que devolven os mesmos valores en ambos tests.

Na segunda parte da práctica, referida ao cálculo das funcións, puidemos observar que o primeiro algoritmo é cuadrático, mentres que o segundo é lineal, o que significa que o segundo é moito máis **rápido** que o primeiro realizando a mesma tarefa.

Consecuencia disto último, vemos que para o primeiro algoritmo os **tempos de execución** medran de maneira exponencial cando o número de elementos dos vectores aumenta, mentres que co segundo algoritmo este aumento dos tempos segue unha **progresión lineal**.