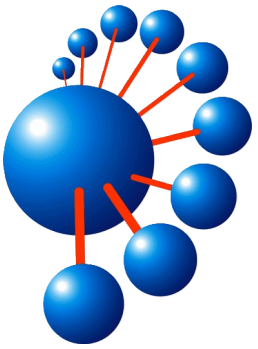


Tarea 3
Análisis de Algoritmos

Profesora: Andrea Rodríguez.
Integrantes: Diego Varas.
Jeremías Torres.

Junio 21, 2017

Introducción

Dados varios segmentos de línea en el eje X se pide encontrar la cantidad mínima de ellos que cubran completamente un segmento [0,M], para ello se tiene que los segmentos ingresados sean coordenadas [Li, Ri], donde Li < Ri y Ri <= M.

Desarrollo

Comenzaremos a cubrir la región [0, M] de izquierda a derecha. Sea s el extremo izquierdo de la región que necesitamos cubrir. Al principio, s = 0. Encontraremos el grupo de segmentos cuya coordenada izquierda es ≤ s, ordenándolos a partir de la coordenada (Ri) y escogiendo el segmento(digamos k) cuya coordenada derecha es la más grande del grupo. Actualizaremos s = Rk y repetiremos el proceso hasta que s ≥ M.

```
int s=0; // el extremo izquierdo de la región que necesita ser cubierta
while (s < M) {
	int max=0;
	int j= -1; // índice del candidato actual
	for (int i=0; i< S.size();i++) { // Para cada segmento i en S
		if (S[i].Li <= s && S[i].Ri >= max) {
			max = S[i].Ri;
			j=i;
		}
	}
	O.push_back(S[j]); //Ponga el segmento j en el conjunto O;
	s = S[j].Ri; // Cambie el valor del extremo izquierdo
	S.erase(S.begin()+j); // Elimine el segmento j de S;
}
```

Demostración:

Sea O el conjunto devuelto por el método greedy.

Supongamos O no es la solución óptima, entonces existe conjunto S de segmentos que cubre [0, M], tal que |S| < |O|. Luego sea k = |S| y j = |O|, entonces k < j.

Entonces O1, O2, …. ,Oj los segmentos de O en orden que son elegidos por el algoritmo Greedy. Entonces RO1 < RO2 < … < ROj-1 < ROj. Del algoritmo Greedy tenemos que ROj-1 < M.

Vamos a ordenar los segmentos en S basado en sus coordenadas correctas. Están en el orden S1, S2, … Sk. Entonces RS1 < RS2 < RS3 < … < RSk.

Según el método de Greedy tenemos RS1 <= RO2; RS2 <= RO2; … ; RSk <= ROk; como k < j, entonces, RSk <= ROk <= ROj-1 < M, eso significa que S1, S2, …, Sk no puede cubrir la región [0, M].

Así S no es la solución. La suposición no es correcta. Así O es la solución óptima.

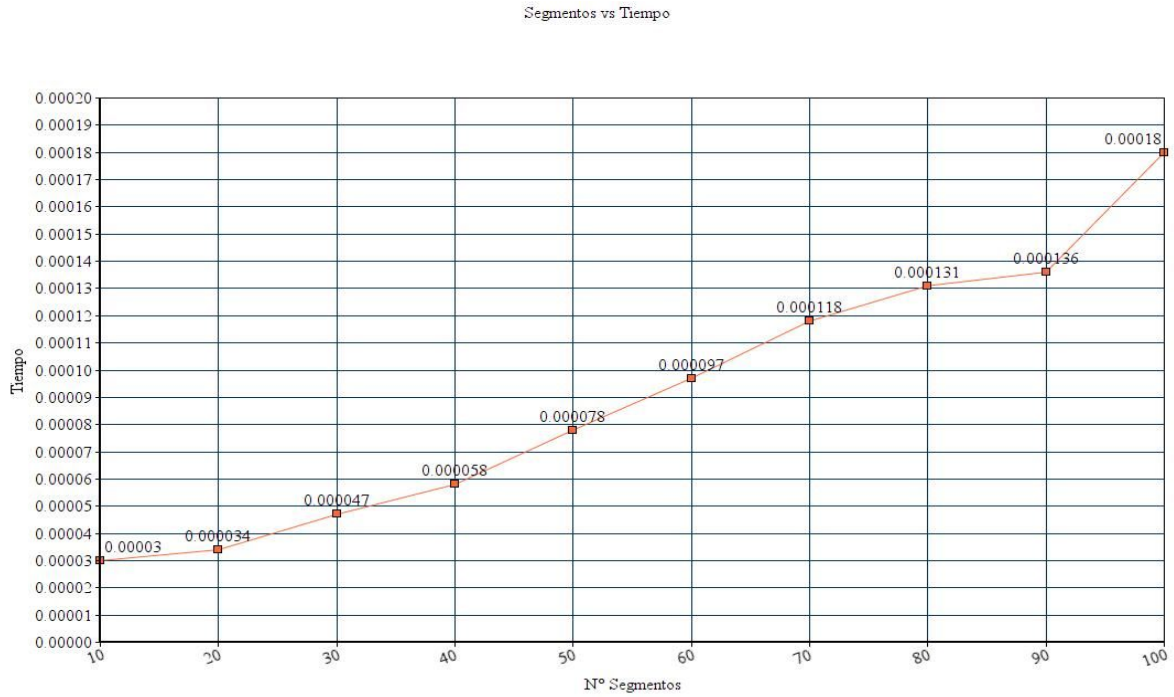
Costo Asintótico:

El costo asintótico del algoritmo usado para resolver este problema es de O(n^2), donde n es el número de segmentos. El ciclo for se recorre de la siguiente manera:

- La primera iteración ejecutará n veces el ciclo for para encontrar el segmento que cubre más parte del segmento principal [0, M], luego se ser encontrado este se eliminará para luego volver a iterar ahora con n-1 segmentos y así sucesivamente.

- En el peor caso iterará hasta que llegue al último segmento, por lo que nos dará $\sum_{i=1}^{n} i$ lo que será $n(n+1)/2$.
- Por lo tanto el coste asintótico del algoritmo es $O(n^2)$.

Evaluación Experimental:



El algoritmo se ejecutó en un Notebook con procesador Intel® Core™ i3-5005U CPU @ 2.00GHz × 4, con 8gb de Ram DDR3L 1600mhz.

Los valores para las pruebas fueron con un M fijo de 100, y un número variable de Segmentos que fueron desde 10 a 100, desordenados.

Los datos entregados se asemejan a una curva n^2, por lo que el costo experimental tiende a O(n^2).

Conclusión

El costo asintótico experimental y teórico se asemejan bastante, lo cual significa que ejercimos de manera acertada el análisis teórico.

Logramos aplicar un algoritmo bastante eficiente para lo que se pedía, ya que se necesitaba encontrar el número mínimo de segmentos para cubrir un segmento dado, lo que implicaba encontrar siempre el segmento “mayor”, por lo que se debía iterar todos los segmentos dados. Esto implica que se encuentra la solución óptima en cada iteración del algoritmo lo cual deducía en un algoritmo greedy, siendo este esquema algorítmico el que menos dificultades presenta a la hora de diseñar y comprobar su funcionamiento.