Proyecto 1

Análisis de Algoritmos

Primer Semestre 2020, Prof. Cecilia Hernández

Fecha Inicio: Miércoles 6 de Mayo 2020.

Fecha Entrega: Martes 19 de Mayo 2020 (23:59 hrs).

1. [0.5 puntos] Ordene de menor a mayor orden asintótico las siguientes funciones.
   a) $3^{2^{1000000}}$
   b) $n\sqrt[3]{n}$
   c) $3^{0,1n}$
   d) n^2
   e) $log(n)^{2\log(n)}$
2. [0.5 puntos] Resuelva las siguientes recurrencias
   a) $T(n) = 4T(n/4) + 5n$
   b) $T(n) = 4T(n/5) + 5n$
   c) $T(n) = 5T(n/4) + 4n$
   d) $T(n) = 4T(\sqrt{n}) + \log^5(n)$
3. [0.5 puntos] Determine si las siguientes afirmaciones son verdaderas o falsas. Justifique su respuesta.
   a) $2^{n/2}$ es $\Theta(2^n)$
   b) $n^{3/2}$ es $O(n\log^2(n))$
   c) Si $f(n) = O(g(n))$ entonces $log(f(n)) = O(log(g(n)))$
   d) $f(n) = 1{,}0000001^n$ es $O(n^2)$
4. [0.5 puntos] Construya los árboles recursivos para las siguientes recurrencias y úselo con el método de substitución para demonstrar la solución de la recurrencia.
   a) $T(n) = T(n/4) + T(n/2) + n^2$
   b) $T(n) = 3T(n/3) + n\log(n)$
5. [0.5 puntos] Resuelva las siguientes recurrencias usando el método de substitución.
   a) $T(n) = 4T(n/2) + 100n$
   b) $T(n) = 4T(\sqrt{n}) + \log^5(n)$

6. [1 punto] Proporcione un análisis asintótico de peor caso en notación $O()$ para el tiempo de ejecución de los siguientes fragmentos de programa.

(a)

```
int F(int x, int y){
   int m = 1;

   if(y == 0){
      return 1;
   }
   while(y > 0){
      if(y % 2 ){
         m *= x;
      }
      x *= x;
      y /= 2;
   }
   return m;
}
```

(b)

```
int recurse(int n) {
   for (i = 0; i < n*n; i += 2) {
      procesar(i); // O(n)
   }
   if (n <= 0)
      return 1;
   else
      return recurse(n-3);
}
```

7. [2.5 puntos] Asuma que existen n personas que entran y salen de una sala. Cada persona x entra en tiempo x_i y sale en tiempo z_i. Si una persona i entra cuando la sala está vacía, la persona i prende la luz, pero si ya hay alguien dentro la luz ya está encendida. Por otro lado, si cuando la persona i sale de la sala no hay nadie mas dentro, la persona i apaga la luz. Puede suponer que $z_i > x_i$ para todo i. Luego, dado los valores $(x_1, z_1), (x_2, z_2), ...(x_n, z_n)$, se desea contar el número de veces que se enciende la luz. Puede asumir, que la persona que sale de la sala tiene prioridad por sobre la persona que entra. Osea si una persona quiere entrar al mismo tiempo en que otra persona sale, la persona que sale lo hace primero, esto es, la persona que sale apaga la luz, luego la persona que entra la prende solo si no hay nadie dentro. Si dos personas salen al mismo tiempo, solo una de ellas apaga la luz. Si dos personas entran el mismo tiempo, solo una de ellas prende la luz si no hay nadie en la sala.

Ejemplos:
$(8, 10), (4, 6), (6, 8), (3, 7), (9, 12)$
2

$(9, 10), (2, 5), (4, 6), (6, 8), (3, 5), (10, 12)$
4

a) Escriba un pseudo código para un algoritmo $O(n^2)$ que resuelva el problema.

b) Diseñe un algoritmo que resuelva el problema en $O(n \log(n))$ y escriba su pseudo código.

c) Demuestre correctitud de sus algoritmo y realice los análisis de tiempo de ejecución.

d) Implemente sus algoritmos usando C++ definiendo una función para cada uno.

e) Realice análisis experimental para lo cual se pide que construya un gráfico que muestre como varían los tiempos de ejecución en nanosegundos variando el tamaño de la entrada (n).

8. Medición de tiempos.

```
#include <chrono>

using namespace std;

auto start = chrono::high_resolution_clock::now();
auto finish = chrono::high_resolution_clock::now();
auto d = chrono::duration_cast<chrono::nanoseconds> (finish - start).count();
cout <<"total time "<< duration << " [ns]" << " \n";

// nota, si estima necesario puede usar milliseconds
// en lugar de nanoseconds
```