1era Lista de Problemas de Análisis y Diseño de Algoritmos

(Otoño-Invierno 2022)

Cristhian Alejandro Ávila-Sánchez

0. Considere una máquina $M = (\Sigma, Q, \delta, q_0, q_f)$, con alfabeto es $\Sigma = \{0,1\}$, conjunto de estados $Q = \{q_0, q_1, q_2, q_3, q_f\}$, estado inicial q_0, estado final q_f y regla de transición:

δ	0	1
q_0	$0Rq_1$	$-$
q_1	$0Rq_2$	$1Rq_1$
q_2	$0Rq_f$	$1Lq_3$
q_3	$1Rq_4$	$-$
q_4	$0Lq_5$	$1Rq_4$
q_5	$-$	$0Rq_f$
q_f	$-$	$-$

a) ¿La máquina siempre llega a un estado final para alguna entrada $\omega = 01^*01^*0$?
b) Si se detiene, escribir en pseudocódigo el algoritmo respectivo.
c) ¿Cuántas operaciones le toma al autómata realizar la concatenación, $\omega \to \omega'$ (i.e. $\omega = 0\omega_1 0\omega_2 0$, $\omega' = 0\omega_1\omega_2 0$, $\omega_1 = 1^*, \omega_2 = 1^*$).
d) ¿Cuántas unidades de memoria utiliza?
e) Implemente el algoritmo en un sistema/fenómeno natural no convencional (p. ej. granos de arena, copos de nieve, hormigas, mariposas, abejas, corcholatas, frijolitos, etc.)

1. Describa las *Ecuaciones de Maxwell* de electromagnetismo como una maquinaria de ruedas y engranes y explique el efecto físico de cada una de las 4 ecuaciones.

2. Reconfigure la compuerta conservativa de *Fredkin-Toffoli* para obtener las funciones lógicas correspondientes a las compuertas AND, OR y NOT. Utilizando estas compuertas, construya un circuito para obtener la función XOR.

3. Demostrar que el algoritmo de formación de parejas de *Gale-Shapley* entrega una solución perfecta y estable (esto es, no hay elementos m o w sin pareja y no hay inestabilidades en donde haya parejas $\langle m, w\rangle$ y $\langle m', w'\rangle$ donde m y w' se prefieren mutuamente sobre m' y w.

4. Contar el número exacto de instrucciones que se ejecutan, así como la complejidad temporal en el mejor y peor de los casos, para cada uno de los siguientes ciclos:

```
for (k=N-1; k>=0; k--)
   arr[k]= k;

for (k=1; k<N; k*=2)
   arr[k]+= k;
for (k=0; k<N; k+=2)
   if (k==N/2)
     k=1;
```

```
for (q=0; q<N; q+=7)
   {
      c= q;
      while (q<N)
           q= 5*q+3;
      q= 2*c;
   }
q= 11;
```

5. Considere el siguiente ciclo, correspondiente al *Problema del Granizo* / *Conjetura de Collatz*:

```
while (x>1)
{
   if (x%2==0)
     x= x/2;
   else
     x= 3*x+1;
}
```

¿Cuántas operaciones realiza el ciclo cuando a) $x = 9$, b) $x = 1619$, c) $x = 2^N$?

6. Calcule la complejidad temporal, en el peor de los casos, del siguiente segmento de código:

```
int c=0, k=0, q=0, N=0;
int arr[N];

for (q=1, k=0; k<N; q++)
   {
      for (c=0; c<q && k<N; c++, k++)
         arr[k]= c;

      if (k<N)
        {
            arr[k]= -1;
            k++;
        }
   }
```

7. Calcule la cantidad exacta de pasos que se ejecutan al correr el siguiente código. Asuma que los tiempos de ejecución de las funciones u(int N) y v(int N) son $T_u(N) = 7N^5$ y $T_v(N) = 3Nlog_6N$. Determine la complejidad temporal en el mejor y peor de los casos.

```
for (k=0, q=1; k<N && q<N; k+=3, q*=2)
   {
      w= u(N) * v(N);
      for (x=N, y=0; x>N/3 || y<N/3; x--, y++)
         w= u(N) + v(N);
   }
```

8. Observe a la siguiente función recursiva:

```
void funcionRecursiva(int N, int profundidad)
{
      int *arr= NULL;

      if (N==profundidad)
        return;
```

```
        arr= (int *) malloc(N*sizeof(int));

        while (x<N)
        {
              funcionRecursiva(N+1, profundidad);
              x++;
        }

        free(arr);
}
```

a) Dibujar el árbol de llamadas recursivas.
b) ¿Cuál es la complejidad temporal correspondiente a la generación de todo el árbol de llamadas?
c) ¿Cuál es la complejidad espacial correspondiente a la generación de todo el árbol de llamadas?
d) ¿Cuántas bifurcaciones recursivas tiene cada invocación?
e) ¿Cuántas hojas tiene el árbol en su nivel más profundo?

9. Escriba un programa que corresponda a la función temporal $T(N) = N^6 + 3N^5 + 9N^4 + 6N\log N$.

10. Determinar la ecuación de recurrencia y la complejidad temporal de los algoritmos de "*Las Torres de Hanoi"* y *"Quicksort"*.

11. Escriba un algoritmo para resolver el problema de *"Las 8 Reinas"* y grafique el árbol de *backtracking*.

12. Dibuje el árbol de invocaciones recursivas y escriba la ecuación de recurrencia del siguiente código. Calcule su complejidad temporal en el peor de los casos. (Nota: asuma que la funcionLineal tiene una complejidad $O(N)$).

```
void funcionRecursiva(int N)
{
      if (N<=0)
        return;

      funcionRecursiva(N/7);
      funcionRecursiva(N/5);

      funcionLineal();

      funcionRecursiva(N/3);
      funcionRecursiva(N/2);
}
```

13. Sean $T_1(m) = 3T_1(m/7) + cm$ y $T_2(m) = 2T_2(m/5) + cm^2$ dos ecuaciones de recurrencia. Obtenga el orden de complejidad temporal de ambas ecuaciones y bosqueje sus árboles recursivos.

14. Un algoritmo común y silvestre le toma, a alto nivel, $f(N)$ pasos ejecutarse. a) Calcule el tiempo de ejecución del algoritmo, a bajo nivel, si se implementa en un programa que corre en un procesador con una velocidad de procesamiento V y cuya arquitectura emplea C ciclos para ejecutar cada instrucción. El procesador permite aumentar/disminuir su velocidad de procesamiento de V_0 a V_x en incrementos/decrementos de v ciclos. Tomando en cuenta estas características, b) calcule la aceleración/deceleración del procesador y c) el tiempo de ejecución del algoritmo conforme va alcanzando la velocidad V_x.

15. Proponga un algoritmo que solucione el problema del *Productor-Consumidor* entre hilos utilizando semáforos. Calcule la complejidad temporal en el peor de los casos de las operaciones de semáforos $down()$ y $up()$ y la complejidad de su algoritmo.

16. Determine el tiempo de ejecución de un algoritmo paralelo cuyo computo es distribuido en *M* hilos, con complejidades $f_0(N)$, $f_1(N)$, …, $f_{M-1}(N)$, cada uno corriendo libremente en un núcleo distinto con velocidades de procesamiento V_0, $V_1, \dots, V_{M-1}$. a) Calcule la complejidad del algoritmo, así como su tiempo de ejecución. ¿Cómo cambia su respuesta si b) hay comunicación y sincronización entre hilos, c) las velocidades de los procesadores se ajustan dinámicamente, acelerando/desacelerando a V_0', V_1', …, V_{M-1}' y d) hay más hilos que núcleos?

17. Planificación estratificada: Calcule la complejidad temporal de un algoritmo de planificación de una cantidad *H* de hilos *Round Robin* sobre un núcleo de procesamiento. Considere que tiene un multiprocesador de *N* núcleos. Calcule la complejidad de planificar $J \geq H$ hilos, en grupos de *W* hilos, sobre los *N* núcleos. Finalmente considere que tiene *M* multiprocesadores y una rejilla de *B* bloques de *J* hilos cada uno. Calcule la complejidad de planificar B bloques sobre los M multiprocesadores. Finalmente calcule la complejidad de planificar un kernel de K hilos sobre esta arquitectura paralela.

18. Obtenga el tiempo de ejecución de un algoritmo que se implementa sobre un arquitectura distribuida y paralela sobre N procesadores. Cada procesador cuenta con M núcleos de procesamiento y cada núcleo tiene una velocidad dinámica (i.e acelerar/desacelerar). V. El algoritmo tiene H hilos. Considere tiempos de planificación y ejecución de los hilos en cada procesador, así como los tiempos de comunicación y sincronización entre hilos.

19. Considere un grafo $G = (V, E)$ ($|V| = N$, $|E| = M$). ¿Cuántos subgráfos contiene el grafo G? Demostrar que el número de árboles dentro del grafo es N^{N-2} (*Fórmula de Cayley*).

20. Considerar un árbol A, formado a partir de un grafo G. Bosquejar la construcción del árbol por profundidad (p. ej. utilizando pilas) y por anchura (p. ej. utilizando colas).

21. Sea M una mochila con un volumen V y con una capacidad para almacenar un peso máximo W. Contabilice el número de posibles combinaciones para incluir un conjunto de N objetos, con diferentes volúmenes v_k, pesos w_k y utilidad u_k, tal que no rebasen la capacidad de la mochila y maximicen su utilidad y proponga una forma "eficiente" para resolver este problema *NP-Completo*.

22. Considere un problema *NP-Difícil* cuya verificación de la solución se realiza en un tiempo no polinomial $O(2^N)$. Proponga una estrategia para explorar el espacio de soluciones de forma "eficiente".

23. Considere una Máquina de Turing con N estados y M transiciones que procesa una cadena de L símbolos. Proponga un análisis para poder contabilizar la cantidad de transiciones y la longitud de la cadena hasta que la máquina se detenga (i.e. que llegue a un estado de aceptación). Si la palabra de entrada no pertenece al lenguaje de la máquina, ¿cuántas transiciones le toma?

24. Plantee un isomorfismo entre los *Teoremas de Incompletitud e Inconsistencia de Gödel* y el *Problema de Detención de la Máquina de Turing*. (Tip: Considere los argumentos utilizados por Gödel y Turing en las demostraciones para resolver el problema de encontrar soluciones enteras a las *ecuaciones Diofantinas*).

25. Un *problema no computable* (p. ej. *El Problema de Detención*, *Domino's de Wang*) es aquel en donde no se puede decidir si su máquina de Turing asociada se detiene (si se detuviese entregaría una respuesta ya sea afirmativa ó positiva al problema). Considere un *conjunto de Domino's de Wang* y establezca un isomorfismo con un circuito lógico combinatorio. Describa la relación que existe entre El Problema de Detención y los Domino's de Wang, en términos de indecidibilidad.