Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Algoritmos y Complejidad

Trabajo Práctico 4

Algoritmos *Dividir y Conquistar*

Primer cuatrimestre de 2019

1. Resolver las siguientes recurrencias e indicar el orden de $t(n)$.

   a) $t(n) = 3t(n-1) - 2t(n-2)$, sujeta a $t(0) = 0$, $t(1) = 1$.

   b) $t(n) = 4t(n-1) - 4t(n-2)$, sujeta a $t(0) = 1$, $t(1) = 4$.

   c) $t(n) = t(n-2)$.

   d) $t(n) = 4t(n-1) + 2^n$, sujeta a $t(0) = 0$, $t(1) = 2$.

   e) $t(n) = t(n-1) + n$.

   En los incisos (a), (b) y (d) determinar las constantes en la solución general.

2. Considere la recurrencia $t(n) = 4t(n-1) - 2^n$. Determinar el orden asintótico exacto de la función cuando:

   - $t(0) = 1$
   - $t(0) = 2$

   ¿Son del mismo orden? ¿Por qué?

3. Cuando se obtiene un orden asintótico de tiempo condicional, suele ser útil el siguiente resultado

   Regla de Suavidad [BB96, Sección 3.4]
   Sea $f : \mathbb{N} \longrightarrow \mathbb{R}^{\geq 0}$ una función suave y $t : \mathbb{N} \longrightarrow \mathbb{R}^{\geq 0}$ una función eventualmente no decreciente. Si $t(n) \in \Theta(f(n)|$ n es potencia de b) entonces $t(n) \in \Theta(f(n))$.

   a) Definir cuando una función es:

   - *eventualmente no decreciente*
   - *b-suave*
   - *suave*

   b) Sea $f : \mathbb{N} \longrightarrow \mathbb{R}^{\geq 0}$ una función b-suave con constantes c y n_0 tales que $f(bn) \leq cf(n)$ para todo $n \geq n_0$. Probar que para cualquier natural positivo i, $f(b^i n) \leq c^i f(n)$ para todo $n \geq n_0$.

   c) Utilizando los resultados del inciso anterior, probar que si una funcion f es *b-suave* para algún entero $b \geq 2$, entonces f es suave.

   d) Demostrar la *Regla de suavidad*.

4. Demostrar que n^k es suave para todo $k \in \mathbb{N}$.

5. Mostrar que las funciones $n^{\log n}$, 2^n, y $n!$ no son suaves. (Ayuda: probar que no son b-suaves para b=2).

6. Efectuar los cambios de variable necesarios y resolver las siguientes recurrencias, indicando *el orden condicional exacto* en el que está $t(n)$. Verificar si es posible aplicar la *Regla de Suavidad*.

   a) $t(n) = 4t(n/2) + n$.
   b) $t(n) = 4t(n/2) + n^2$.
   c) $t(n) = 2t(n/2) + \lg n$.
   d) $t(n) = 2t(n/2) + n \lg n$.
   e) $t(n) = 2t(\sqrt{n}) + \lg n$ (para n de la forma $2^{(2^k)}$).

7. *Teorema Maestro* [CLRS09, Sección 4.5]

   a) Indicar qué recurrencias del ejercicio 6 pueden resolverse utilizando el método del *Teorema Maestro*. Resolver dichas recurrencias utilizando ese método.
   b) Resuelva las siguientes recurrencias utilizando el Teorema Maestro.
      - $t(n) = 3t(n/2) + n^2$
      - $t(n) = 16t(n/4) + n$.
      - $t(n) = 3t(n/3) + n/2$.

8. Existe una generalización del método de la ecuación característica que permite resolver recurrencias de la forma

$$t(n) = a_1 t(n-1) + a_1 t(n-2) + \ldots + a_k t(n-k) + b_1^n p_1(n) + b_2^n p_2(n) + \ldots$$

   donde los b_i son constantes y los p_i son polinomios en n de grado s_i. La ecuación característica para estas recurrencias se construye de la siguiente manera:

$$(x^k - a_1 x^{k-1} - \ldots - a_k)(x - b_1)^{s_1+1}(x - b_2)^{s_2+1} \ldots = 0$$

   Utilizando la generalización del método de la ecuación característica, resolver las recurrencias que se enuncian a continuación. Determinar el valor de las constantes en la solución general.

   a) $t(n) = 2t(n-1) + n + 2^n$, sujeta a $t(0) = 0$.
   b) $t(n) = 4t(n-1) - 4t(n-2) + 3^n + n^2$.
   c) $t(n) = 4t(n-1) - 2^n$, sujeta a $t(0) = 1$.
   d) $t(n) = 3t(n-1) - 2t(n-2) + 3(2^{n-2})$, sujeta a $t(0) = 1$ y $t(1) = 2$.

9. Para la implementación recursiva de la operación de *búsqueda* en un Árbol Binario de Búsqueda, enunciar y resolver la recurrencia generada por su tiempo de ejecución en el peor caso.

10. Considere el problema de encontrar el mínimo y el máximo elemento de un arreglo de n elementos. Dar un (único) algoritmo Dividir y Conquistar que resuelva este problema y analice el tiempo de ejecución.

11. Plantear y resolver las recurrencias del tiempo de ejecución de QuickSort para:

    - Peor caso.
    - Mejor caso.

12. Considerar la variante del algoritmo de mergesort que en lugar de dividir el arreglo en dos mitades lo separa en tres partes de tamaños $\lfloor n/3 \rfloor$, $\lfloor (n+1)/3 \rfloor$ y $\lfloor (n+2)/3 \rfloor$, que son ordenadas recursivamente y luego mezcladas para obtener la solución final. Escribir el pseudocódigo y realizar un análisis del orden del tiempo de ejecución de este algoritmo.

13. Adaptar el algoritmo de MergeSort para que reciba un arreglo de números $A[1 \dots N]$ y retorne un nuevo arreglo con los elementos de A ordenados y sin duplicados. Por ejemplo, para $A = [1, 7, 12, 3, 22, 12, 7, 12]$ la salida debería ser $[1, 3, 7, 12, 22]$.

14. *Multiplicacion de enteros largos* [BB96, Sección 7.1]

    Sean u y v enteros de exactamente m y n dígitos respectivamente, con $m \leq n$. El algoritmo clásico de multiplicación obtiene $u \times v$ con un tiempo de ejecución en $O(nm)$. El algoritmo de dividir y conquistar desarrollado en teoría considera que ambos números son del mismo tamaño, aún cuando m es mucho mas pequeño que n; esto provoca que el orden del tiempo ejecución del algoritmo sea $O(n^{\log 3})$.

    Reformular este algoritmo de modo que se aproveche la diferencia de tamaño para lograr un tiempo de ejecución en $O(nm^{\log(3/2)})$.

15. Dado un arreglo de enteros no necesariamente positivos $A[1 \dots N]$, se desea hallar el subarreglo $A[i \dots j]$ cuya suma sea máxima. Es decir, se desea hallar un par de índices $1 \leq i \leq j \leq N$ tal que la suma $A[i] + A[i+1] + \dots + A[j]$ sea la máxima posible.

    Por ejemplo, para el arreglo $A = [3, -4, 5, -2, -2, 6, -3, 5, -3, 2]$, el subarreglo de suma máxima es $[5, -2, -2, 6, -3, 5]$ cuya suma es 9.

    a) Escribir el pseudocódigo de un algoritmo que resuelva el problema. El algoritmo deberá resolver una instancia de tamaño N a partir de 2 instancias de tamaño $\frac{N}{2}$ y el orden del tiempo de ejecución de la etapa de combinación deberá ser $O(N)$.

    b) Realizar el análisis del tiempo de ejecución a partir de una recurrencia y el Teorema Maestro.

16. Dado un arreglo ordenado $A[1 \dots N]$ de enteros **distintos**, algunos de los cuales pueden ser negativos. Dar un algoritmo que encuentre un índice i tal que $A[i] = i$ ó detecte que no existe tal índice. El tiempo de ejecución debe estar en el orden de $O(\log n)$ en el peor caso.

17. *Par de puntos más cercanos* [CLRS09, Seccion 33.5]

    a) ¿Cuál es la máxima cantidad de puntos que entran en un cuadrado de $d \times d$ si la distancia entre cada par de puntos es al menos d?

    b) En la etapa de combinación del algoritmo que resuelve este problema, se recorren todos los puntos dentro de *Franja* y se chequea la distancia con otros 7 puntos. Justificar por qué son suficientes 7 puntos.

    c) Determine el orden del tiempo de ejecución de la etapa de combinación y del algoritmo completo cuando:

    - Se ordenan los puntos por coordenada y en cada etapa de combinación.
    - Se recibe como parámetro la lista de puntos ordenados por coordenada y.

18. *Exponenciación modular* [BB96, Sección 7.7]

    a) Sea $N(n)$ la función que indica la cantidad de multiplicaciones realizadas por el algoritmo de *exponenciación modular* para calcular a^n. Defina $N(n)$ como una recurrencia.

    b) Analizar cuántas multiplicaciones realiza el algoritmo de *exponenciación modular* para calcular a^{15} y a^{16}.

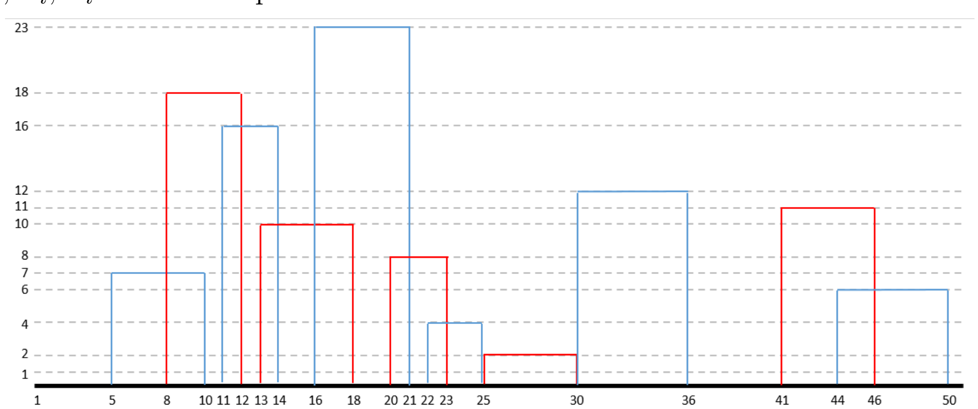
    c) Mostrar que $N(n)$ NO es eventualmente no decreciente.

    d) Mostrar que $N(n)$ está en el orden $\Theta(\log n)$.

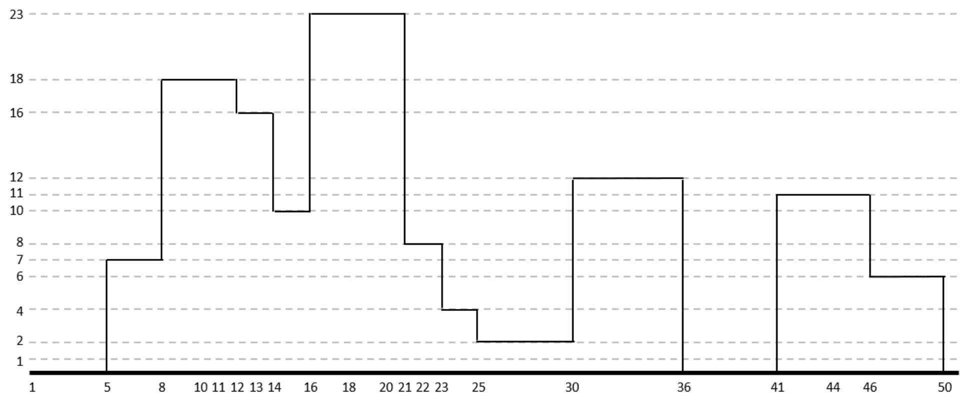
19. Sea T un arreglo de n elementos. Un elemento x se dice mayoritario si el número de elementos de T iguales a x es mayor que $n/2$. Escribir un algoritmo para encontrar el elemento mayoritario de un arreglo (si es que existe), cuyo tiempo de ejecución sea $O(n \log n)$.

20. *Dibujando el horizonte*

    Se desea desarrollar un algoritmo para asistir en la tarea de dibujar en 2 dimensiones el horizonte de una ciudad dados los datos de altura y ubicación de sus edificios. En esta ciudad todos los edificios tienen contornos rectangulares y están construidos sobre un terreno alineado y nivelado. Cada edificio se representa como una tupla de 3 valores (I_i, H_i, D_i) donde I_i y D_i son los extremos izquierdo y derecho respectivamente del edificio i y H_i es la altura de dicho edificio. Todos los valores I_i, H_i, D_i son enteros positivos.



    Por ejemplo dados los edificios $(5, 7, 10)$ $(8, 18, 12)$ $(11, 16, 14)$ $(13, 10, 18)$ $(16, 23, 21)$ $(20, 8, 23)$ $(22, 4, 25)$ $(25, 2, 30)$ $(30, 12, 36)$ $(41, 11, 46)$ $(44, 6, 50)$, estos se ubican en la ciudad como se ve en la figura 1. El horizonte a obtener para esta ciudad es el que se muestra en la figura 2. La representación utilizada para un horizonte es una secuencia de valores $(v_1, v_2, v_3, ..., v_{n-1}, v_n)$ en la que los v_i tales que i es impar representan lineas verticales del horizonte (extremos de edificios) y los v_i tales que i es par representan lineas horizontales (alturas de los edificios).



Para el ejemplo de horizonte de la figura 2 la secuencia del horizonte es la siguiente:

$(5, \mathbf{7}, 8, \mathbf{18}, 12, \mathbf{16}, 14, \mathbf{10}, 16, \mathbf{23}, 21, \mathbf{8}, 23, \mathbf{4}, 25, \mathbf{2}, 30, \mathbf{12}, 36, \mathbf{0}, 41, \mathbf{11}, 46, \mathbf{6}, 50, \mathbf{0})$

La secuencia del horizonte puede ser vista como el camino que se realiza para hacer el dibujo sin levantar el lapiz comenzando en el mínimo punto de las coordenadas x y moviendose vertical y horizontalmente por el contorno hasta el final. De esta manera en cada posición de las coordenadas x en las que no haya un edificio, incluyendo al final del horizonte, el valor que representa esa situación en la secuencia es el 0.

a) Desarrolle un algoritmo que dada la lista de tuplas de los edificios construya la secuencia del horizonte como se describe en este enunciado utilizando la técnica Dividir y Conquistar. Diseñe para esto un algoritmo que a partir de dos horizontes de una misma ciudad H_1 y H_2 construya el horizonte resultante para la ciudad completa.

Por ejemplo: la mezcla de los horizontes $(1, \mathbf{4}, 6, \mathbf{8}, 10, \mathbf{2}, 13)$ y $(8, \mathbf{5}, 11, \mathbf{9}, 12)$ es el horizonte $(1, \mathbf{4}, 6, \mathbf{8}, 10, \mathbf{5}, 11, \mathbf{9}, 12, \mathbf{2}, 13)$.

Identifique en su resolución: cuáles son los subproblemas, cómo los obtiene a partir del problema original y cómo obtiene finalmente la solución general (combinación de soluciones a subproblemas).

Si hace uso de primitivas *muestre su implementación en pseudocódigo o explique detalladamente qué realiza* cada una.

b) Analice cuál es el orden del tiempo de ejecución de su algoritmo. Muestre la recurrencia y explique cómo la resuelve.

Referencias

[BB96] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics*. Prentice Hall, 1996.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction To Algorithms*. The MIT Press, 3rd edition, 2009.