Complejidad y Técnicas de Diseño de Algoritmos

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL RESISTENCIA

GIUA DE TRABAJOS PRACTICOS

2023

Equipo Docente:
Profesor Asociado: Dr. Acuña Cesar Javier.
JTP: Ing. Tortosa Nicolás.

PRACTICO 1: COMPLEJIDAD ALGORÍTMICA

1. El análisis de la eficiencia de un algoritmo supone determinar la cantidad de recursos informáticos consumidos por el mismo (es decir, cantidad de memoria y cantidad de tiempo de cómputo requeridos). Para llevar adelante este análisis hay diferentes metodologías. Enuncie las ventajas y desventajas de cada una.
2. En el análisis de eficiencia o complejidad algorítmica se considera muy importante la manera en que la función de complejidad va aumentando con el tamaño del problema, lo que se conoce como crecimiento asintótico. ¿Cuál es el comportamiento asintótico de las siguientes funciones de complejidad algorítmica?
    a. $(n + 1)^3$
    b. $3 + 2^n$
    c. $7n - 2$
    d. Log n
3. Sea a una constante real, $0<a<1$. Usar las relaciones $\subset$ y = para ordenar los órdenes de complejidad de las siguientes funciones:

$(n/2)\log n$	$n^3+8n^2+\log n$	n^2	n^3+5
$\log n$	$n!+n$	4^n	$n^{(1/2)}$

4. Teniendo en cuento todos los ordenes de complejidad que conoce, analice y busque un problema que responda a dicho orden de complejidad.
5. Si tengo que elegir entre dos algoritmos con su correspondiente crecimiento asintótico, cual implementaría?
    a. $O(n)$ ó $O(2^n)$
    b. $O(\log n)$ ó $O(n^2)$
    c. $O(n \log n)$ ó $O(n)$
6. Dados los siguientes algoritmos, calcular su tiempo de ejecución (T(n)) en el Peor Caso, y decir qué complejidad o crecimiento asintótico tiene (O).

Siendo:

```
Constante n
Vector: arreglo [1 .. n] de enteros
```

a)

```
Procedimiento Algoritmo1 (vector a) Ambiente
    Entero j, i, temp; Inicio
    Para i:= 1 a n-1 hacer
            Para j:= n a i+1, -1 hacer
                    Si a[j-1]> a[j] entonces temp:=
                            a[j-1];
                            a[j-1]:= a[j]; a[j]:=
                            temp;
                    fin_si;
            Fin_para;
    Fin_para;
Fin
```

b)

```
Función Algoritmo2 (vector a, entero c): entero Ambiente
     Entero inf, sup, i;
Inicio
     inf:=1; sup:=n;
     Mientras (sup>=inf) hacer i:= (inf +
               sup) / 2;
               Si a[i]=c entonces
                         Retorna i;
               Sino
                         Si c < a[i] entonces
                                   sup := i – 1;
                         sino
                                   inf:= i + 1;
                         fin si
               fin si
   fin mientras
   Retorna 0;
FIN
```

c)

```
   Función Algoritmo3 (entero n, m): entero; Ambiente
               Entero temp;
   Inicio
   Mientras m > 0 hacer
          temp:= m;
             m:= n MOD m;
             n:= temp;
   Fin_mientras;
   Retorna n;
   Fin
```

d)

```
Procedimiento Algoritmo4 (entero n) Ambiente
   Entero i, j, k, s;
Inicio
   S:= 0;
   Para i:= 1 a n-1 hacer
      Para j:= i+1 a n hacer
               Para k:= 1 a j hacer
                         S:= s+2;
               Fin_para;
          Fin_para;
   Fin_para;
Fin
```

PRACTICO 2: RECURSIVIDAD y ORDENACIÓN

1) Escriba un algoritmo que realice una búsqueda binaria recursiva en un vector de N posiciones.

2) Utilizando la técnica de divide y vencerás, realice un algoritmo recursivo para resolver el problema de las torres de Hanói.

   Torres de hanoi de 3 postes (A,B,C): El problema de las torres de Hanoi consiste en pasar N discos, uno por uno (o uno a la vez), del poste A, al poste B con la condición de no poner un disco más grande sobre otro más pequeño, se puede usar el poste C como auxiliar.

   La solución debe mostrar por pantalla, el orden de los movimientos a efectuar.

3) Sea v un vector de componentes Integer positivas que se ajustan al perfil de una curva

   cóncava, es decir, que existe una única posición k en el vector tal que:

   - Los elementos a la izquierda de k están ordenados descendentemente
   - Los elementos a la derecha de k están ordenados ascendentemente

   Diseñar el método recursivo que más eficientemente determine dicha posición k.

4) Modificar los algoritmos Quicksort y Mezcla de forma que sustituyan las llamadas recursivas por llamadas al procedimiento Selección cuando el tamaño del vector a ordenar sea menor que una cota dada M.

PRACTICO 3: DIVIDE y VENCERÁS

1) Construir una función recursiva que, dado un número entero, devuelva como resultado el factorial del mismo.

2) Dados dos números: a y b. Calcule a^b, usando sólo multiplicaciones sucesivas.

3) Dados dos números enteros n y m, construir una función recursiva que calcule el mínimo común múltiplo entre ambos.

4) Dos amigos matan el tiempo de espera en la cola del cine jugando a un juego muy sencillo: uno de ellos piensa un número natural positivo y el otro debe adivinarlo preguntando solamente si es menor o igual que otros números. Diseñar un algoritmo eficiente para adivinar el número.

5) Dados un vector V [1..n] y un número natural k entre 1 y n-1, diseñar un algoritmo eficiente que transponga los k primeros elementos de V con los elementos en las n-k últimas posiciones, sin hacer uso de un vector auxiliar. Por ejemplo, si V es el siguiente vector con 10 elementos:

a	b	c	d	e	f	g	h	i	j

y k = 3, el resultado deseado es

d	e	f	g	h	i	J	a	b	c

6) Dados dos vectores de enteros X[1..n] e Y [1..n] con n >= 1 ordenados en forma creciente, escribir un algoritmo para hallar la mediana del vector formado por el total de los 2n elementos. La mediana de un grupo de números es el número central de ese grupo de números ordenados por tamaño.

7) Diseñar un algoritmo de coste en O(n) que dado un conjunto S con n números, y un entero positivo k <= n, determine los k números de S más cercanos a la mediana de S.

8) Dado un árbol, la distancia entre dos nodos es la longitud del único camino que los conecta, y el diámetro del árbol es la distancia máxima sobre todos los pares de nodos. Desarrollar un algoritmo con complejidad O(n) (con respecto al número de nodos del árbol) para hallar el diámetro de un árbol binario dado. Si la cantidad de términos es par, la mediana es el promedio de los dos números centrales.

9) Juan Perdedor es aficionado al juego y todas las noches va a jugar una partida de póker. Como es un jugador responsable, para controlar sus pérdidas, anota lo que gana y lo que pierde cada noche como un número entero de euros. Por ejemplo, la siguiente tabla muestra los resultados cosechados por Juan Perdedor en el último mes:

				290	-70	
140	210	300	-470	10	70	-390
230	-200	-360	-410	270	-340	70
480	350	-460	-160	320	180	50
330	270	280	-220	10	-200	-420

Se observa que empezó bien el mes con una ganancia de $290 pero terminó con una pérdida de $420. El beneficio total obtenido a lo largo de todo el mes es $-500. Analizando esta información Juan Perderdor se da cuenta de que si hubiera empezado a jugar el día 16 y terminado el día 26, habría maximizado sus ganancias, obteniendo $1050, una diferencia de $1550 con respecto a su situación actual.

Dado un vector de ganancias/pérdidas de longitud n se desea encontrar el sub-vector sobre el cual se obtiene el beneficio total máximo.

a) Desarrollar un algoritmo iterativo de $O(n^2)$

b) Desarrollar un algoritmo de tipo divide y vencerás de O(n log n).

c) Desarrollar un algoritmo de tipo divide y vencerás de O(n).

d) Desarrollar un algoritmo iterativo de O(n)

PRACTICO 4: Algoritmos Ávidos o Voraces

1) Resolver el siguiente problema utilizando un algoritmo voraz: En un tablero de ajedrez (de tamaño n x n) partimos de una casilla inicial (x,y). Tenemos una ficha de un caballo, que puede realizar los mismos movimientos que en el ajedrez. El objetivo es, partiendo de la posición inicial, visitar todas las casillas del tablero, sin repetir ninguna.

2) Resolver el siguiente problema utilizando un algoritmo voraz: En un tablero de ajedrez (de tamaño n x n) definir si existen casillas (x,y) que siendo iniciales, logren el propósito de que un caballo recorra todas y cada una de las casillas sin repetir ninguna, sin ser necesario que el punto de partida y de final sean el mismo.

3) El problema consiste en, teniendo unos objetos que se desean meter en una mochila, siendo que la mochila solo puede llevar un peso máximo y si queremos meter todos los objetos se sobrepasa el peso máximo de la mochila, seleccionar los objetos que maximicen la ganancia del vendedor. Para ello cada objeto tiene un peso y un valor material asociado.

4) El conocido genio de la finanzas “Pepe el Broker” recibe el encargo de invertir en la bolsa de valores una cantidad de M millones de euros procedentes de un fondo de inversión. Haciendo uso de sus conocimientos bursátiles, nuestro magnate elabora una lista de los n valores más interesantes. Para cada valor v_i, se conocen sus características más importantes: C_i es la cantidad máxima que la CMV acepta para una sola inversión en v_i, B_i es el beneficio que, si todo sale bien, cabe esperar de cada euro invertido en v_i. Pero como es bien sabido, la Bolsa no es una ciencia exacta, de donde se cuantifica en p_i la probabilidad de que efectivamente se consiga la ganancia esperada. Se pide superar los conocimientos de Pepe el Broker en cuanto a finanzas, y elaborar un algoritmo devorador que maximize los beneficios esperados de una inversión. Nota: Si en el valor i se invierte x_i, el beneficio es $x_i*B_i*p_i$

5- A través de la técnica voraz, y siguiendo la técnica de Prim, diseñar un algoritmo que halle el árbol de recubrimiento mínimo de un grafo conexo, no dirigido y valorado. El algoritmo de Prim comienza por un vértice y escoge en cada etapa el arco de menor peso que verifique que uno de sus vértices se encuentre en el conjunto de vértices ya seleccionados y el otro no. Al incluir un nuevo arco a la solución, se añaden sus dos vértices al conjunto de vértices seleccionados.

6- A través de la técnica voraz, y siguiendo la técnica de Kruskal, diseñar un algoritmo que halle el árbol de recubrimiento mínimo de un grafo conexo, no dirigido y valorado. En el de Kruskal se ordenan primero los arcos por orden creciente de peso, y en cada etapa se decide qué hacer con cada uno de ellos. Si el arco no forma un ciclo con los ya seleccionados (para poder formar parte de la solución), se incluye en ella; si no, se descarta.

7- Establecer una red eléctrica entre ciertas ciudades de forma que el coste de la red sea mínimo. Dado: G=(V,E,p) un grafo no dirigido conexo y ponderado, con V: conjunto de ciudades, E: enlaces factibles, y p: coste de los enlaces (no negativos). La red eléctrica deseada será un subconjunto de arcos T de E, sin ciclos que conecte todos los vértices y cuyo peso total $p(T)=\sum_{(u,v)\in T} p(u,v)$ sea mínimo.

8- Un empleado de una Agencia de Turismo tiene encargada la tarea de definir las ruta más corta, la distancia recorrida y la cantidad de combustible necesaria para cada viaje, para cada uno de los destinos turísticos que la empresa ofrece, considerando siempre como punto de partida la ciudad donde se encuentra localizada la agencia, que es City Aburrida. Escribir un algoritmo que, teniendo como entrada una lista con todos los destinos posibles, y un grafo con todas ras rutas (donde los vértices son las distintas ciudades y los arcos las rutas que las unen), modifique el algoritmo de Dijkstra para resolver dicho problema.

PRACTICO 5: VUELTA ATRÁS

1) **Grafica el árbol combinatorio de backtracking** del problema de encontrar un subconjunto de {12, 23, 1, 8, 33, 7, 22} que sume exactamente 50.

2) **Problema del Salto del Caballo**
   Grafica el árbol de backtracking del problema acotado y hallando solo la 1°.
   Resolver el siguiente problema utilizando un algoritmo con vuelta atrás: En un tablero de ajedrez (de tamaño n x n) partimos de una casilla inicial (x,y). Tenemos una ficha de un caballo, que puede realizar los mismos movimientos que en el ajedrez. El objetivo es, partiendo de la posición inicial, visitar todas las casillas del tablero, sin repetir ninguna.

3) **Problema del Viajante**
   Un viajante de comercio debe cumplir una ruta con varias ciudades todos los días. Conocidas las distancias entre todas las ciudades, hallar el camino más corto que pase por todas las ciudades de la ruta del viajante.

4) **Problema de la Mochila**
   El problema consiste en, teniendo unos objetos que se desean meter en una mochila, siendo que la mochila solo puede llevar un peso máximo y si queremos meter todos los objetos se sobrepasa el peso máximo de la mochila, seleccionar los objetos que maximicen la ganancia del vendedor. Para ello cada objeto tiene un peso y un valor material asociado.

5) **Problema de las N Reinas**
   Grafica el árbol binario de backtracking del problema para un tablero de 4 x 4.
   El problema consiste en colocar n reinas en un tablero de ajedrez de dimensión n x n, sin que se den jaque (dos reinas se dan jaque si comparten fila, columna o diagonal).

6) **Problema del laberinto**
   Nos encontramos en una entrada de un laberinto y debemos intentar atravesarlo encontrando el camino que nos lleve hasta su salida.

7) **Problema de coloreo de grafos**
   Grafica el árbol de permutaciones de backtracking del problema, suponiendo un grafo de .
   Sea G=(N, A) un grafo no dirigido. Deseamos colorear los nodos de G de tal manera que no haya dos nodos adyacentes del mismo color. El problema pregunta además ¿cuál es el mínimo número de colores que se necesita?. Este número mínimo se denomina número cromático del grafo.

8) **Asignación de tareas**
   Se dispone de un número de agentes “n” asi como de un número de tareas “n” que deben de resolver los agentes. Se ha de tener en cuenta que cada agente sólo puede resolver una única tarea. El objetivo consiste en que el coste de la asignación de agentes a tareas sea el menor posible.

9) **Juego: Ta-Te-Ti**
   Grafica el árbol de backtracking del problema.
   Implementar un juego de TA-TE-TI donde una persona juegue contra la computadora, que “piensa” sus jugadas. El programa debe permitir que el usuario seleccione la celda donde ubicará su ficha, indicando su coordenada, y luego el movimiento de la computadora. El juego termina cuando el tablero está lleno, ó algún jugador ha hecho TA-TE-TÍ.

10) Juego: Solo 1

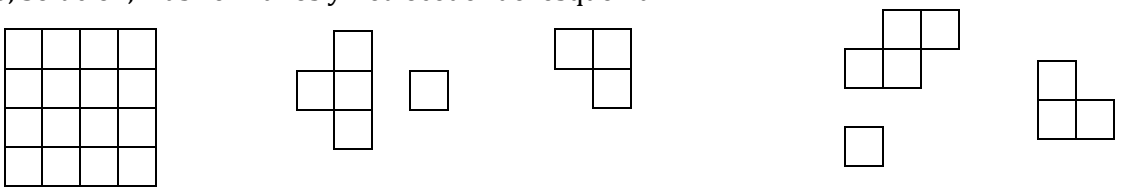
En el juego del Continental se colocan 32 piezas en un tablero de 33 casillas, tal y como se indica en la _gura:

```
    o o o
    o o o
o o o o o o o
o o o   o o o
o o o o o o o
    o o o
    o o o
```

Solo se permiten movimientos en vertical y horizontal.
Una pieza solo puede moverse saltando sobre otra y situándose en una casilla libre. La pieza sobre la que se salta se retira del tablero.
Se consigue terminar con éxito cuando queda una sola pieza en la posición central del tablero.
Diseña un algoritmo de backtracking que encuentre una serie de movimientos para llegar al final del juego.

11) Rompecabezas

Se quiere hacer un programa por backtracking que resuelva un rompecabezas consistente en rellenar completamente una plantilla cuadriculada con unas ciertas piezas dadas (ver el dibujo de abajo). Las piezas se pueden rotar y cada pieza sólo se puede usar una vez. Explicar cómo vendrían dados los datos del problema, cómo se podría representar una solución, cómo sería el árbol de soluciones, cuál sería la condición de final, y cómo serían los procedimientos Generar, Criterio, Solución, MasHermanos y Retroceder del esquema.



PRACTICO 6: Programación Dinámica

1) Diseñar el algoritmo de Fibonacci utilizando programación dinámica.

2) Resolver el problema del cambio utilizando programación dinámica, defina usted los parámetros necesarios como tipo de monedas y valor, siempre teniendo en cuenta que debe existir la moneda con valor 1.

3) Dada una secuencia X={x1 x2 ... xm}, decimos que Z={z1 z2 ... zk} es una subsecuencia de X (siendo $k \leq m$) si existe una secuencia creciente {i1 i2 ... ik} de índices de X tales que para todo j = 1, 2, ..., k tenemos xij = zj. Dadas dos secuencias X e Y, decimos que Z es una subsecuencia común de X e Y si es subsecuencia de X y subsecuencia de Y. Deseamos determinar la subsecuencia de longitud máxima común a dos secuencias.

4) Sean u y v dos cadenas de caracteres. Se desea transformar u en v con el mínimo número de operaciones básicas del tipo siguiente: eliminar un carácter, añadir un carácter, y cambiar un carácter.

   Por ejemplo, podemos pasar de abbac a abcbc en tres pasos:

   abbac → abac (eliminamos b en la posición 3)
   → ababc (añadimos b en la posición 4)
   → abcbc (cambiamos a en la posición 3 por c)

   Sin embargo, esta tranformación no es óptima. Lo que queremos en este caso es diseñar un algoritmo que calcule el número mínimo de operaciones, de esos tres tipos, necesarias para transformar u en v y cuáles son esas operaciones, estudiando su complejidad en función de las longitudes de u y v.

5) Dados n elementos e1, e2, ..., en con pesos p1, p2, ..., pn y beneficios b1, b2, ..., bn, y dada una mochila capaz de albergar hasta un máximo de peso M (capacidad de la mochila), queremos encontrar las proporciones de los n elementos x1, x2, ..., xn ($0 \leq xi \leq 1$) que tenemos que introducir en la mochila de forma que la suma de los beneficios de los elementos escogidos sea máxima.