Tarea y Examen 3
Algoritmos y Estructuras de Datos I

Fecha de entrega: Jueves 2 de Junio hora de clase

Indicaciones:

- La tarea y examen se realizará de forma individual. La calificación para cada uno (tarea y examen) será acorde a los problemas resueltos. El puntaje y a que rubro pertenece el problema esta marcado.
- Como entregables se entregará un archivo para el ejercicio 9, éste archivo deberá contener el código fuente de la implementación. Además se deberá entregar un documento impreso en el cual se de la solución a los demás problemas.
- El archivo con el código, se enviará por correo. El asunto del correo será [TareaExamen 03] (ojo poner los corchetes). En el contenido del correo se deberán escribir el nombre del alumno.
- Por ningún motivo se recibirán tareas de forma extemporánea a la fecha límite.

Ejercicios:

1.- (Tarea - 1 punto)
Investigue y haga un breve resumen de lo que son los problemas P, NP y $NP - completos$. ¿Que significa $N = NP$ y cual es su relevancia en el área de las ciencias computacionales? ¿En qué capitulo de los Simpson se hace referencia a este resultado?

2.- (Examen - 2 puntos)
Las siguientes son funciones de complejidad resultado del análisis de diversos algoritmos. Ordene las funciones de menor a mayor dependiendo de su orden de complejidad. Argumente su respuesta.

$$f_1(n) = 10^n$$
$$f_2(n) = n^{1/3}$$
$$f_3(n) = n^n$$
$$f_4(n) = ln(n)$$
$$f_5(n) = 2\sqrt{ln(n)}$$

3.- (Tarea - 2 puntos)
Sean $f(n)$ y $g(n)$ dos funciones tales que no toman valores negativos y suponga que $f(n) = O(g(n))$. Demuestre que $g(n) = \Omega(f(n))$.

4.- (Examen - 2 puntos)
Dadas las siguientes funciones de complejidad, indica cuales son ciertas y cuales son falsas. Argumenta cada una de tus respuestas en caso de ser verdadera y de un contra ejemplo en caso de ser falso.

$$\begin{aligned} i)\ n^2 = O\left(n^3\right) \qquad & ii)\ n^3 = O\left(n^2\right) \\ iii)\ 2^{n+1} = O\left(2^n\right) \qquad & iv)\ 2^{n+1} = O\left(2^n\right) \\ v)\ n^3 = \Omega\left(n^2\right) \qquad & vi)\ ln(n) = \Omega\left(n^{1/2}\right) \end{aligned}$$

5.- (Tarea - 3 puntos)
Sea $G = (V, E)$ una gráfica no dirigida cuyas aristas $e \in E$ tienen un costo $c_e \geq 0$. Sea T el árbol de peso mínimo de G. Ahora asuma que una nueva arista con un costo c es agregada a la gráfica tal que ésta conecta los vértices $v, w \in V$. Entonces:
a) Proponga un algoritmo que se ejecute en $O(|E|)$ que pruebe sí T sigue siendo o no el árbol de peso mínimo para G. ¿Puede hacerse un algoritmo que sea del orden de $O(|V|)$? Analiza en tu algoritmo las estructuras de datos usadas representar al árbol T y la gráfica G ya que estas pueden interferir en el orden de complejidad de tu algoritmo propuesto.
b) Suponga que T ya no es el árbol de peso mínimo de G después de agregar la nueva arista. Proponga un algoritmo de orden lineal sobre las aristas $O(|E|)$ que actualice a T para volver a ser el árbol de peso mínimo de G.

Notas:
i) El árbol de peso mínimo resulta de aplicar el algoritmo de Dijkstra en una gráfica.
ii) Una gráfica no dirigida puede verse como una gráfica dirigida en la cual cada arista conecta a un par de vértices de ida y de regreso.

6.- (Examen - 2 puntos)
Dado el algoritmo "Merge Sort" visto en clase, muestre la secuencia de pasos completa que ejecutaría el algoritmo al ordenar el siguiente arreglo de caracteres `A = [e s t o e s f a c i l]`

7.- (Tarea - 3 puntos)
Usando la técnica de Divide y Vencerás la partición y el reparto equilibrado de los subproblemas es el punto crucial en el diseño de estos algoritmos. En el problema de la busqueda binaria si nos planteamos lo siguiente: supongamos que en vez de dividir el vector de elementos en dos mitades del mismo tamaño, las dividimos en dos partes de tamaños 1/3 y 2/3. ¿Conseguiremos de esta

forma un algoritmo mas eficiente que el original?
Para dar su respuesta escriba el código del algoritmo propuesto y obtenga la función de complejidad y en base a esto argumente su respuesta.

8.- (Examen - 3 puntos)
Las monedas usadas en un lejano país tienen las denominaciones de $v_1, v_2, ..., v_n$. Suponiendo que $v_1 = 1$, que $v_j = 2v_i$ con $j = i + 1$ y que disponemos de un número ilimitado de monedas de cada denominación.
a) ¿Existe un algoritmo que pudiera dar cualquier cantidad de dinero usando el mínimo de monedas requerido?
b) Si la moneda de valor unitario dejara de existir, ¿el argumento mencionado en a) seguiría siendo valido?
c) Analice el tiempo de ejecución del algoritmo y encuentre el orden de complejidad del mismo.
Nota: para a) de el algoritmo y para todos los incisos argumente su respuesta.

9.- (Tarea y Examen - 2 puntos)
Implemente las siguientes funciones de forma recursiva. Y describa brevemente el algoritmo propuesto.

i) Una función que imprima el contenido de una lista
`def print_list(list):`

ii) Una función que diga si una cadena es un palíndromo
`def is_palindrome(string):`

iii) Una función que genere e imprima el triángulo de Sierpinski de tamaño size
`def sierpinski(size):`

iv) Una función que genere e imprima el fractal de Koch de tamaño size
`def koch(size):`

Nota: todas las funciones deberán ser implementadas en el lenguaje de programación pyhton.