Estructuras Computacionales | 2024-2 | Ejercicios de práctica

Los enunciados aquí presentes se deben resolver implementando programas en Python.

1. Implementar la búsqueda binaria para un vector o lista de N elementos. La búsqueda binaria se basa en el principio de la división en dos partes del conjunto de elementos en los que se va a buscar. El procedimiento es más o menos así:

Se inicia ordenando el vector. Después se identifica el elemento de la mitad del vector; ese será nuestro punto de división. Se generan así dos particiones: una desde el inicio hasta el punto de la mitad y la otra desde la mitad hasta el final del conjunto.

Se revisa si el valor a buscar es menor o mayor que el valor del elemento en el punto de división. Si el elemento es menor, las posición de inicio sigue siendo la misma, pero la del final se reemplaza por la de la mitad. Si el valor es mayor, la posición de inicio se reemplaza por la de la mitad y la del final se deja como estaba. De esta manera se reduce el conjunto de elementos en los que hay que buscar, recortándolo a la mitad cada vez, de manera sucesiva. Si en algún punto el valor de la mitad es igual al buscado se detiene el proceso; Si nunca se encuentra un valor igual al de búsqueda, el proceso continúa hasta la posición de final sea menor o igual a la de inicio, condición a la que se debe llegar pues el conjunto se reduce a la mitad en cada paso.

Nota: Este algoritmo es bastante conocido y de el se encuentra bastante información en recursos dirigidos al aprendizaje de la programación.

2. Construir un programa que permita cifrar y descifrar una frase usando el cifrado conocido coloquialmente como el cifrado de César. Este cifrado consiste en reemplazar cada letra de un texto por la letra que está tres posiciones adelante en el alfabeto, por ejemplo cambiar todas las “A” por “D”; este valor de tres, para tres posiciones del alfabeto, lo llamaremos valor de salto, y puede variar. Este programa debe hacer algo más: usar el valor de salto de 3 para las palabras en posiciones pares y el valor de salto de 4 para las que estén en posiciones impares. El programa naturalmente tomará los espacios en blanco como separador de palabras y los ignorará para el proceso de reemplazo, al igual que los signos de puntuación.

3. Construir una calculadora de funciones trigonométricas que permita seleccionar una función a través de un menú, por ejemplo:

1. seno.
2. coseno.
3. tangente.
4. salir.

Y que al darle un valor de x en grados, obtenga el valor de la función para ese valor de x, pero usando la formula de serie infinita de las función seleccionada. Para el cálculo se

utilizan n términos de la serie, obteniendo así un valor aproximado de la función. Las series que representan al seno y coseno son:

$$\sin(x)=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1}$$

$$\cos(x)=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!}x^{2n}$$

recordar que hay que cambiar el valor de infinito en el límite de la serie por n.

Para el caso de la tangente no hay que implementarla, pero el programa debe mostrar un mensaje indicando que con esa opción se podría hallar, pero que aún no ha sido implementada.

Hacer también una interfaz gráfica para realizar los mismos cálculos.

4. Un número es un *cuadrado perfecto* si su raíz cuadrada es un número exacto (sin decimales). Por ejemplo, el 4 es un cuadrado perfecto (2^2), al igual que lo son el 36 (6^2) y el 3.500.641 (1871^2).

Todos los números que *no* son cuadrados perfectos pueden multiplicarse por otros para conseguir serlo. Por ejemplo, el número 8 no es un cuadrado perfecto, pero al multiplicarlo por 2 se obtiene el 16, que sí lo es.

Entradas del programa: La entrada comienza con un número que indica cuántos casos de prueba tendrán que procesarse. Cada caso de prueba consiste en un número mayor que 0 y menor que 2^{31}.

Salidas: Para cada caso de prueba, el programa escribirá por la salida estándar, en una línea independiente, el número más pequeño que al ser multiplicado por el número del caso de prueba da como resultado un cuadrado perfecto.

5. Construir un programa que reciba las componentes en ***x*** y ***y*** de un vector y calcule una proyección del mismo sobre un par de vectores unitarios al azar. El programa de permitir recibir más de un vector, pero uno a la vez. Para cada caso graficar el punto inicial y los puntos que representan las proyecciones.

6. (Ejercicio Especial) Construir un programa que tome una cadena de caracteres e imprima todas la posibles permutaciones de los mismos. Antes de construirlas, el programa deberá limpiar posibles caracteres repetidos de la cadena. Se deberá obtener un número N de elementos que se toman, y se debe tener en cuenta el número M de posibles valores, es decir, el número de caracteres diferentes en la cadena dada. Por ejemplo:

cadena de entrada: abcfga

elementos que se tienen en cuenta: abcfg
permutaciones:
abcfg
abcgf
abgfc
...

El programa deberá imprimir todas las posibles permutaciones, que se entienden como todos los posibles ordenamientos del conjunto de caracteres que resulta de la limpieza de la cadena dada. Cada permutación se debe convertir en una cadena de texto a la hora de presentar los resultados.