**Modelo matemático para: Algoritmo Voraz, descomponer billetes**

Para descomponer un monto de manera voraz, se puede utilizar el siguiente modelo matemático:

Definir el monto inicial: M

Definir una lista de denominaciones o valores disponibles para descomponer el monto: [D1, D2, D3, …, Dn]

Crear una lista vacía para almacenar las denominaciones utilizadas: denominaciones utilizadas = []

Ordenar la lista de denominaciones en orden descendente: [Dn, Dn-1, …, D3, D2, D1]

Iniciar un bucle hasta que el monto descomponga completamente en denominaciones:

a. Inicializar una variable contador en 0: contador = 0

b. Recorrer la lista de denominaciones:

i. Mientras el monto sea mayor o igual que la denominación y se cumpla que (contador \* denominación) + denominación <= M:

1. Incrementar el contador en 1: contador += 1

ii. Si el contador es mayor que 0:

1. Restar al monto la cantidad (contador \* denominación)

2. Añadir el par (denominación, contador) a la lista de denominaciones utilizadas

3. Reiniciar el contador a 0

Retornar la lista de denominaciones utilizadas.

Este modelo matemático se puede implementar en diversos lenguajes de programación y adaptar según las necesidades del problema.

**Modelo matemático para: búsqueda lineal:**

La búsqueda lineal es un algoritmo de búsqueda que recorre todos los elementos de una lista en orden, uno por uno, hasta encontrar el elemento buscado o llegar al final de la lista.

Un modelo matemático para la búsqueda lineal podría ser el siguiente:

Dada una lista de elementos [x1, x2, x3,…, xn] y un elemento a buscar y.

Inicializar el índice i en 0, que representa la posición del elemento actual en la lista.

Repetir los pasos 4 y 5 hasta que se encuentre el elemento buscado o se llegue al final de la lista.

Si el elemento en la posición i es igual al elemento buscado y, se ha encontrado y se devuelve la posición i en la lista.

Incrementar el índice i en 1 para pasar al siguiente elemento en la lista.

Si se ha recorrido toda la lista y no se ha encontrado el elemento buscado, se devuelve un valor especial que indica que el elemento no está en la lista.

Este modelo matemático describe de manera general el proceso de búsqueda lineal y puede ser utilizado para analizar su complejidad y tiempo de ejecución.

**Modelo matemático para: Ordenamiento por burbuja:**

El bubble sort es un algoritmo de ordenamiento simple que funciona comparando y intercambiando repetidamente los elementos adyacentes si están en el orden incorrecto. Aquí está el modelo matemático para el bubble sort:

Dado un arreglo de elementos A de tamaño n, el bubble sort se puede representar matemáticamente de la siguiente manera:

Se establece un contador i desde 0 hasta n-1 (i = 0 a n-1) para realizar las iteraciones del algoritmo.

Se establece un segundo contador j desde 0 hasta n-i-1 (j = 0 a n-i-1) para comparar los elementos adyacentes y realizar los intercambios necesarios.

Si A[j] > A[j+1], se intercambian los elementos A[j] y A[j+1].

Repetir los pasos 2 y 3 para j hasta que termine la iteración de j.

Repetir los pasos 1 a 4 para i hasta que termine la iteración de i.

Este modelo matemático describe cómo el algoritmo bubble sort compara y ordena los elementos en el arreglo hasta que estén en el orden correcto.

**Modelo matemático para atm**

Un modelo matemático para un cajero automático (ATM) podría incluir las siguientes variables:

Cantidad de dinero disponible en el ATM en cada momento.

Monto máximo de retiro permitido por transacción.

Cantidad de billetes de cada denominación en el ATM.

Número de transacciones realizadas en un tiempo determinado.

Monto promedio de retiro por transacción.

Cantidad de billetes dispensados por transacción.

Las ecuaciones o funciones matemáticas podrían ser las siguientes:

Actualización de la cantidad de dinero disponible en el ATM después de cada transacción: Dinerodisponibleactualizado = Dinerodisponibleanterior - Monto\_retirado

Cálculo del monto máximo de retiro permitido: Montomáximoretiro = Dinerodisponible \* Porcentajemáximo\_retiro

Actualización de la cantidad de billetes de cada denominación después de cada transacción: Billetesdisponiblesactualizados = Billetesdisponiblesanteriores - Billetes\_dispensados

Cálculo del monto promedio de retiro por transacción: Montopromedioretiro = Sumademontosretirados / Númerode\_transacciones

Cálculo de la cantidad de billetes dispensados por transacción: Billetesdispensados = Montoretirado / Denominación\_billete

Estas ecuaciones y variables podrían utilizarse en un modelo matemático para simular el funcionamiento de un ATM y analizar aspectos como el impacto de la disponibilidad de dinero, el número de transacciones realizadas, la denominación de billetes, entre otros.

**Modelo matemático para saldos del cliente en ATM**

Un posible modelo matemático para representar los saldos del cliente en un cajero automático (ATM) podría ser el siguiente:

S = S\_0 + D - R

Donde:

S es el saldo actual del cliente.

S\_0 es el saldo inicial del cliente.

D representa el total de depósitos realizados por el cliente.

R representa el total de retiros realizados por el cliente.

Este modelo asume que el saldo del cliente se actualiza de acuerdo a los depósitos y retiros realizados. El saldo inicial (S\_0) se suma al total de depósitos (D) y se resta el total de retiros (R) para obtener el saldo actual (S).

**Modelo matemático ordenar programación dinámica**

La programación dinámica se utiliza para resolver problemas de optimización en los que es posible dividir el problema en subproblemas más pequeños y luego combinar las soluciones de estos subproblemas para obtener la solución óptima del problema original. A continuación se muestra un modelo matemático básico para un problema de ordenamiento:

Dados n elementos {x1, x2, …, xn} que se desea ordenar en orden ascendente, el objetivo es encontrar una permutación {x'1, x'2, …, x'n} tal que x'1 ≤ x'2 ≤ … ≤ x'n.

Para modelar este problema con programación dinámica, se puede definir una función de coste C(i) que represente el coste mínimo para ordenar los primeros i elementos de la lista. La función de coste se define de la siguiente manera:

C(i) = min{C(j) + cost(i, j)} para todo j ≤ i-1

donde cost(i, j) representa el coste de ordenar los elementos desde j+1 hasta i. En este caso, el coste se puede definir como el número de inversiones necesarias para ordenar estos elementos.

Dado que la función de coste depende de los subproblemas más pequeños (C(j) para j < i), es posible utilizar la técnica de memoización para almacenar los valores ya calculados y evitar recalcularlos en futuros cálculos.

El algoritmo recursivo para resolver este problema utilizando programación dinámica sería:

Crea un arreglo de tamaño n para almacenar los valores de la función de coste.

Inicializa C(0) = 0, ya que no hay elementos para ordenar.

Para i de 1 a n, calcula C(i) utilizando la fórmula recursiva mencionada anteriormente.

Devuelve C(n) como la solución final.

Este modelo matemático y algoritmo básico se pueden adaptar para abordar diferentes problemas de ordenamiento utilizando programación dinámica. Es importante tener en cuenta que el modelo matemático y el algoritmo específico pueden variar dependiendo de las características y restricciones del problema en particular.

**Justifica**

|  |  |
| --- | --- |
| Implementa | Describe |
| Clases | Estructura la app/ |
| Método de ordenamiento por burbuja | Tiene complejidad n x m |
| Método de búsqueda |  |
| Método voraz |  |
| Método para actualizar saldos en ATM | Hay diferentes métodos para actualizar saldos en un cajero automático (ATM, por sus siglas en inglés), para el proyecto se aplica el método de: **actualización en tiempo real**: Con este método, cada vez que se realiza una transacción en el cajero automático, el saldo de la cuenta correspondiente se actualiza en tiempo real. Esto implica una conexión constante con la institución financiera y un proceso de actualización rápida. Es el método más preciso y actualizado, pero también puede requerir recursos significativos y una infraestructura técnica estable. |