Soluciones a los problemas: Clase 7 - Búsqueda binaria

Problemas: Clase 7 - Algoritmos voraces (greedy) 1

Encontrar el mínimo de una lista

```
# Leer un numero entero que indica la cantidad de elementos
n = int(input())

# Leer la lista de numeros separados por espacio
lista = list(map(int, input().split()))

# Tomar el primer elemento como el menor inicialmente
menor = lista[0]

# Recorrer el resto de la lista para encontrar el menor valor
for ele in lista[1:]:
    # Actualizar el menor si se encuentra un valor mas pequeno
    menor = min(menor, ele)

# Imprimir el valor menor encontrado
print(menor)
```

Comer los caramelos más dulces

```
# Leer dos numeros enteros separados por espacio
n, m = list(map(int, input().split()))

# Leer la lista de numeros
li = list(map(int, input().split()))

# Ordenar la lista de menor a mayor
li.sort()

# Calcular la suma de los m valores mas grandes
print(sum(li[-m:]))
```

Reducir arreglo con costo mínimo

```
# Leer un numero entero que indica la cantidad de elementos
n = int(input())

# Leer la lista de numeros separados por espacio
li = list(map(int, input().split()))
```

```
# Calcular (n - 1) multiplicado por el valor minimo de la lista
print((n - 1) * min(li))
```

El candado numérico

```
# Leer el primer numero como lista de digitos
a = list(map(int, list(input())))

# Leer el segundo numero como lista de digitos
b = list(map(int, list(input())))

# Inicializar la variable para acumular el resultado
sol = 0

# Recorrer todos los digitos
for i in range(len(a)):
    # Calcular el numero minimo de movimientos para igualar los digitos
    # considerando que el dial va del 0 al 9 (puede girar hacia adelante o atras)
    sol += min(
        abs(a[i] - b[i]),       # distancia directa
        9 - a[i] + b[i] + 1,    # girar hacia adelante pasando por 9
        9 - b[i] + a[i] + 1     # girar hacia atras pasando por 0
    )

# Imprimir la suma total de movimientos minimos
print(sol)
```

Seleccionar elementos con suma mayor que el resto

```
# Leer la cantidad de elementos del arreglo
n = int(input())

# Leer la lista de numeros no negativos
li = list(map(int, input().split()))

# Calcular la suma total de todos los elementos
total = sum(li)

# Inicializar contadores
can = 0   # cantidad de elementos seleccionados
su = 0    # suma acumulada de los elementos seleccionados

# Ordenar la lista de menor a mayor para tomar los elementos mas grandes primero
li.sort()

# Recorrer la lista desde el elemento mas grande hacia el mas pequeño
for i in range(n - 1, -1, -1):
    su += li[i]   # sumar el elemento actual a la suma acumulada
    can += 1      # contar este elemento como seleccionado
```

```
    # Detenerse cuando la suma seleccionada sea mayor que la suma del resto
    if su > total - su:
        break

# Imprimir la cantidad minima de elementos necesarios para superar la mitad
print(can)
```

Niños y cookies

```
# Leer el numero de ninos
n = int(input())
# Leer la lista de niveles de hambre
la = list(map(int, input().split()))

# Leer el numero de cookies
m = int(input())
# Leer la lista de tamanos de cookies
lb = list(map(int, input().split()))

# Ordenar las listas para usar el enfoque voraz
la.sort()
lb.sort()

# Contadores
sol = 0  # numero de ninos satisfechos
p = 0    # indice del nino actual

# Recorrer todas las cookies
for i in range(m):
    # Si la cookie actual satisface al nino actual
    if p < n and lb[i] >= la[p]:
        sol += 1  # aumentar el numero de ninos satisfechos
        p += 1    # pasar al siguiente nino

# Imprimir el resultado final
print(sol)
```

Cambio de monedas mínimo

```
def cambio_minimo(monedas, N):
    # Ordenar las monedas de mayor a menor para usar la moneda mas grande posible
primero
    monedas.sort(reverse=True)

    resultado = []      # Lista para guardar las monedas seleccionadas
    restante = N       # Cantidad restante que necesitamos formar

    # Recorrer cada tipo de moneda
    for coin in monedas:
```

```
        # Mientras la moneda pueda usarse sin exceder el resto
        while coin <= restante:
            resultado.append(coin)    # Agregar la moneda a la solucion
            restante -= coin          # Reducir el monto restante

        # Si ya se formo el valor exacto, salir del bucle
        if restante == 0:
            break

    return resultado

# Leer el numero de tipos de monedas
m = int(input())
# Leer la lista de monedas disponibles
monedas = list(map(int, input().split()))
# Leer el valor total a formar
N = int(input())

# Calcular la combinacion minima de monedas
resultado = cambio_minimo(monedas, N)

# Imprimir las monedas seleccionadas separadas por espacio
print(" ".join(map(str, resultado)))
```