

# MERGE SORT

DIVIDE Y  
VENCERAS

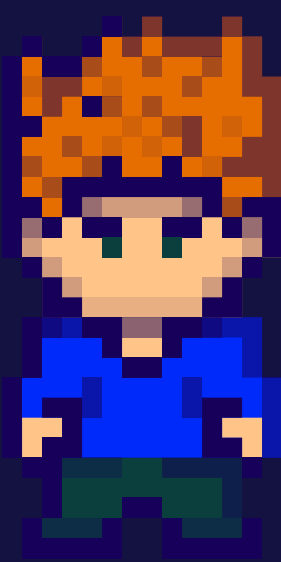
START



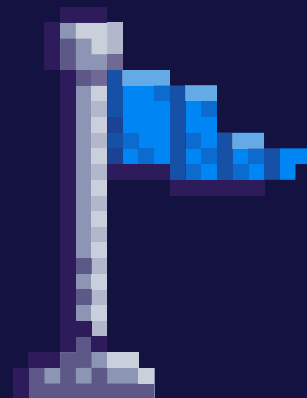
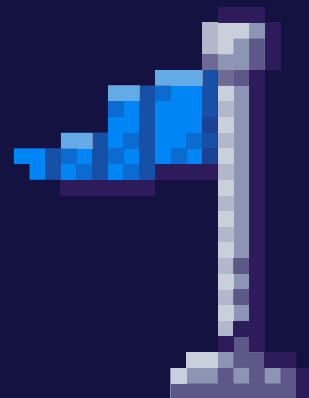
JHONATAN



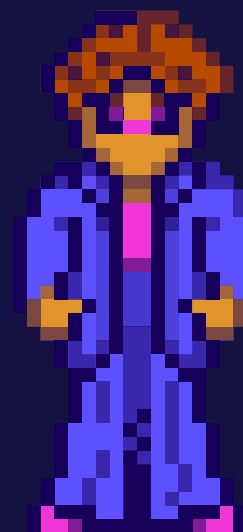
DANTE



VALENTIN



JUAN



JIMMY



FLORENCIO



# ¿QUE ES *MERGESORT*?



**MergeSort es un método para ordenar una lista de elementos como números o palabras de manera eficiente. La idea principal de este algoritmo es dividir la lista en partes más pequeñas, ordenarlas y luego unirlos de nuevo en el orden correcto**



## 1. Dividir:

**Toma la lista grande y la divides en mitades hasta que cada parte tenga un solo elemento. Un solo elemento está, por supuesto, ordenado.**



## 2.Ordenar:

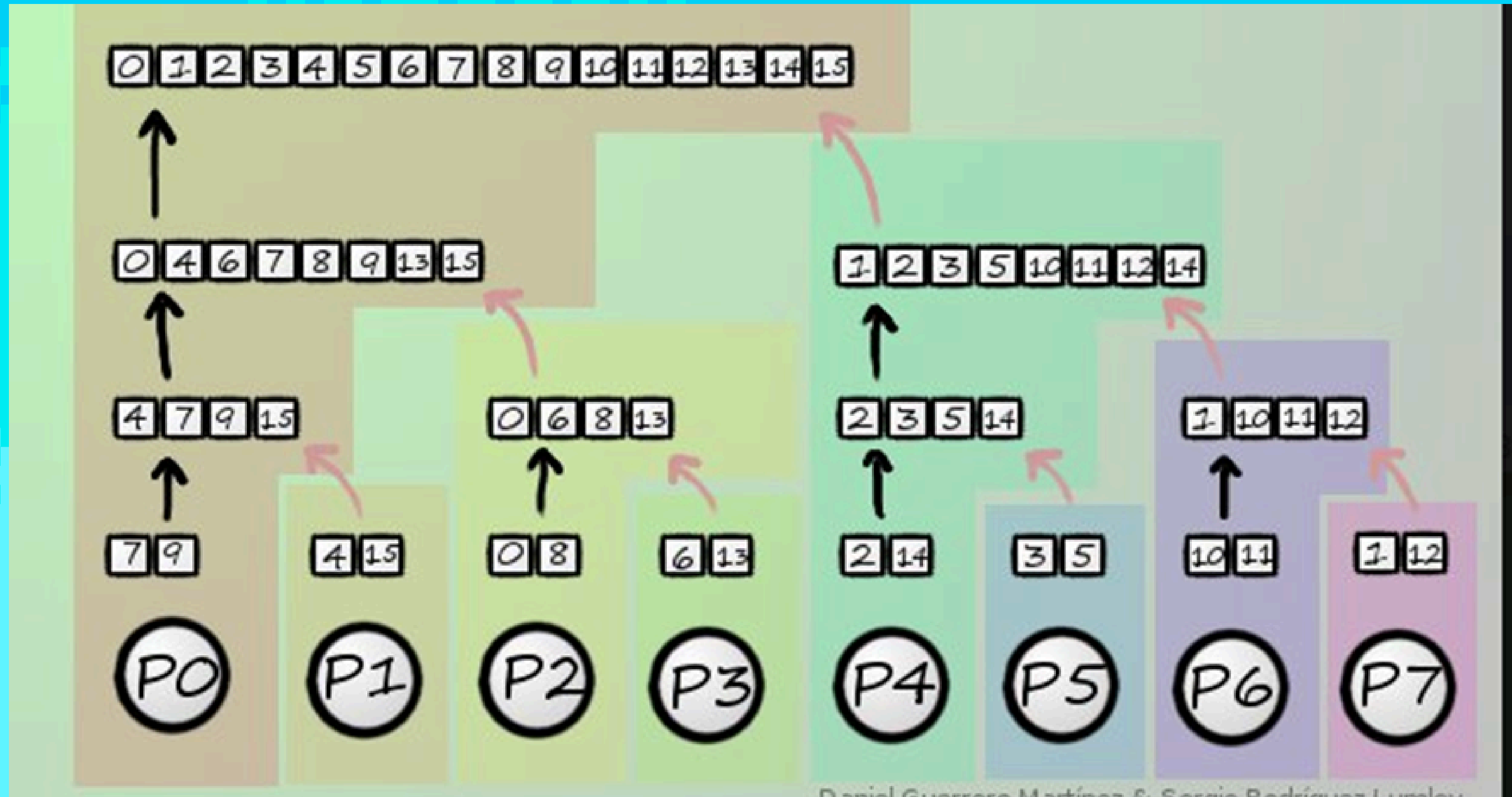
**Luego, vas tomando esas pequeñas partes y las ordenas. Esto se hace de manera simultánea, es decir, varios procesos pueden estar ordenando diferentes partes al mismo tiempo. Esto se llama "paralelización".**



### 3.Mezclar:

**Finalmente, una vez que todas las pequeñas partes están ordenadas, las vas combinando de dos en dos. Durante esta mezcla, te aseguras de que todo quede en el orden correcto.**



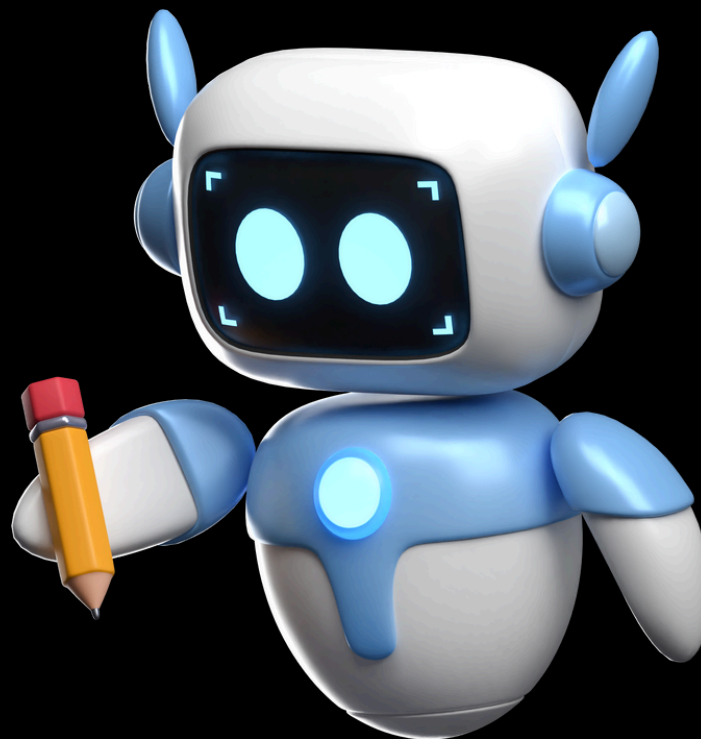


# codigo de MERGE SORT

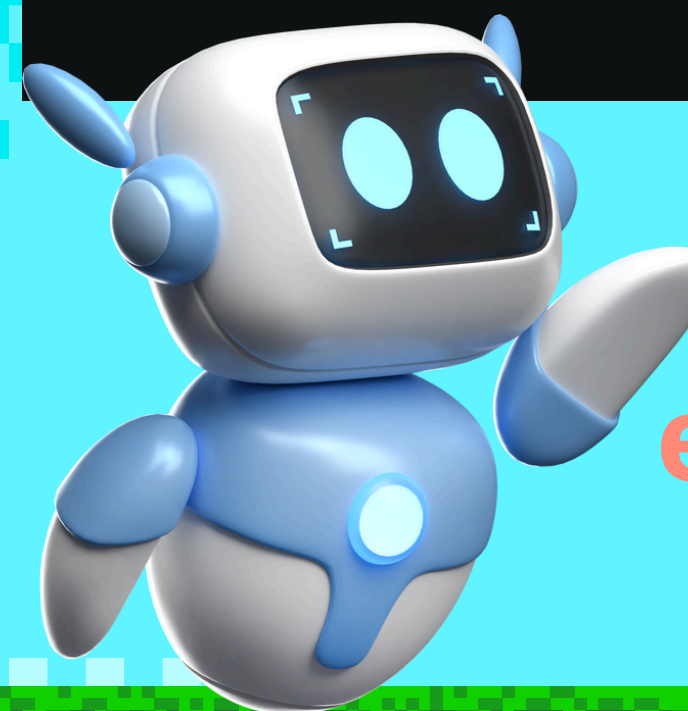
```
def merge_sort(arreglo):
    longitud = len(arreglo)
    mitad = longitud//2
    if longitud <= 1:
        return arreglo
    mitad_izquierda = arreglo[:mitad]
    mitad_derecha = arreglo[mitad:]
    mitad_izquierda = merge_sort(mitad_izquierda)
    mitad_derecha = merge_sort(mitad_derecha)
    return merge(mitad_izquierda, mitad_derecha)

def merge(izquierda, derecha):
    print(f"Recibo {izquierda} y {derecha}")
    arreglo_ordenado = []
    indice_de_izquierda = 0
    indice_de_derecha = 0
    indice_arreglo_ordenado = 0
    while indice_de_izquierda < len(izquierda) and indice_de_derecha < len(derecha):
        valor_izquierda = izquierda[indice_de_izquierda]
        valor_derecha = derecha[indice_de_derecha]
        if valor_izquierda <= valor_derecha:
            arreglo_ordenado.append(valor_izquierda)
            indice_de_izquierda += 1
        else:
            arreglo_ordenado.append(valor_derecha)
            indice_de_derecha += 1

        indice_arreglo_ordenado += 1
    while indice_de_izquierda < len(izquierda):
        arreglo_ordenado.append(izquierda[indice_de_izquierda])
        indice_de_izquierda += 1
    while indice_de_derecha < len(derecha):
        arreglo_ordenado.append(derecha[indice_de_derecha])
        indice_de_derecha += 1
    print(f"Los ordeno y combino. Resultado: {arreglo_ordenado}.")
```



```
Los ordeno y combino. Resultado: [4, 8].
Recibo [5] y [4, 8]
Los ordeno y combino. Resultado: [4, 5, 8].
Recibo [6] y [9]
Los ordeno y combino. Resultado: [6, 9].
Recibo [2] y [6, 9]
Los ordeno y combino. Resultado: [2, 6, 9].
Recibo [4, 5, 8] y [2, 6, 9]
Los ordeno y combino. Resultado: [2, 4, 5, 6, 8, 9].
arreglo ordenado: [2, 4, 5, 6, 8, 9]
PS C:\Users\ASUS>
```

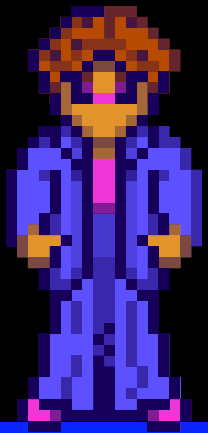


ejecutando el  
MERGE SORT



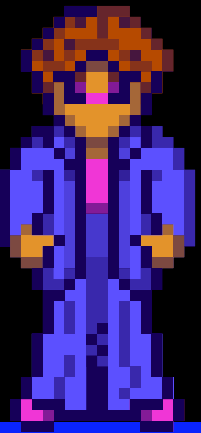


- **EFICIENCIA EN LAS LISTAS GRANDES:**

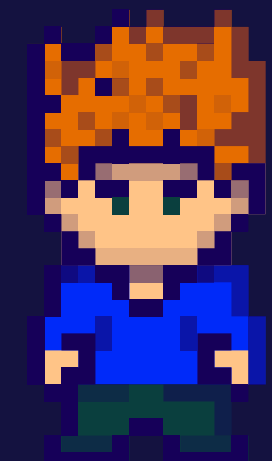


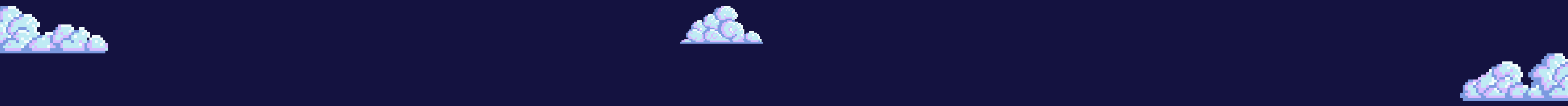
**#:La parte de mezclar (juntar las listas ordenadas) es más sencilla y rápida en comparación con ordenar cada parte. Al dividir y ordenar partes pequeñas, MergeSort puede manejar listas grandes de manera eficiente.**

**#:La paralelización permite que se aprovechen múltiples procesadores, haciendo que el proceso sea aún más rápido.**

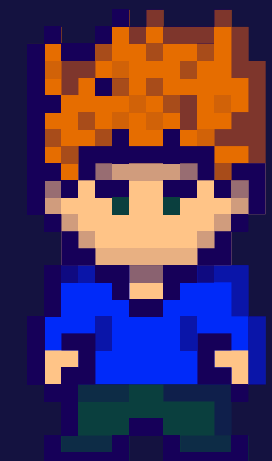


CONCLUSIÓN:



Three pixel art clouds are positioned at the top of the slide: one on the left, one in the center, and one on the right. They are rendered in shades of light blue and white against a dark blue background.

**En conclusion MergeSort es un algoritmo que ordena listas dividiéndolas, ordenando esas divisiones al mismo tiempo y luego combinándolas de forma ordenada. Esto lo hace muy eficiente, especialmente con listas grandes.**



GRACIAS