```
from flask import Flask, request, jsonify
from ibge import busca, get_nome, bubble_sort, merge_sort, quick_sort
from time import time

app = Flask(_name_)

@app.route("/busca_cidades")
def busca_cidades():
    try:
        sort = request.args.get("sort")
        response = busca()
        lista = get_nome(response)

        if sort == 'bubble':
            inicio = time()
            bubble = bubble_sort(lista)
            fim = time()
            tempo = fim - inicio
            return jsonify({"lista_ordenada": bubble['lista'], "numero_comparacoes":
bubble['contador'], "tempo_de_execucao": tempo})

        elif sort == 'merge':
            inicio = time()
            merge = merge_sort(0, len(lista), lista)
            fim = time()
            tempo = fim - inicio
            return jsonify({"lista_ordenada": merge['lista'], "numero_comparacoes":
merge['contador'], "tempo_de_execucao": tempo})

        elif sort == 'quick':
            inicio = time()
            quick = quick_sort(lista)
            fim = time()
            tempo = fim - inicio
            return jsonify({"lista_ordenada": quick['lista'], "numero_comparacoes": quick['contador'],
"tempo_de_execucao": tempo})

        else:
            return jsonify({'message': 'Metodo de ordenacao nao encontrado!'})
    except Exception as e:
        return f"Falha na rota /busca_cidades: {e}"


app.run(debug=True)
```

```
import requests
from unidecode import unidecode
def busca():
    url = f"https://servicodados.ibge.gov.br/api/v1/localidades/municipios"
    resposta = requests.get(url)
    return resposta.json()

def get_nome(obj):
    if len(obj) > 0:
        conteudo = obj
        lista = []
        for i in conteudo:
            lista.append(unidecode(i['nome']))

        return lista
    else:
        return []

# ------ BUBBLE SORT ------ #

def bubble_sort(L):
    contador = 0
    j = len(L)-1
    while j>0:
        for i in range(0,j):
            if L[i]>L[i+1]:
                L[i], L[i+1] = L[i+1],L[i]
                contador +=1
        j = j-1
    print(contador)
    return {'lista':L, 'contador': contador}

# ------ MERGE SORT ------ #

def intercala(inicio, meio, fim, lista):
    w_lista = []
    i = inicio
    j = meio
    contador = 0
    while (i < meio and j < fim):
        contador+=1
```

```
        if (lista[i] < lista[j]):
            w_lista.append(lista[i])
            i+=1
        else:
            w_lista.append(lista[j])
            j+=1

    while j < fim:
        w_lista.append(lista[j])
        j+=1

    while i < meio:
        w_lista.append(lista[i])
        i+=1

    for k in range(inicio, fim):
        lista[k] = w_lista[k-inicio]

    return contador


def merge_sort(inicio, fim, lista):
    contador = 0
    if inicio < fim - 1:
        meio = (inicio + fim) // 2
        merge_sort(inicio, meio - 1, lista)
        merge_sort(meio, fim, lista)
        contador += intercala(inicio, meio, fim, lista)
    return {'lista':lista, 'contador': contador}

# ------ QUICK SORT ------ #

def quick_sort(lista):
    total_comparacoes = quick_sort_ordenado(lista, 0, len(lista) - 1)
    return {'lista':lista, 'contador': total_comparacoes[0]}

def quick_sort_ordenado(lista, esq, dir):
    comparacoes = 0
    if esq < dir:
        indice, comp_particao = particao(lista, esq, dir)
        comparacoes += comp_particao
        comp_esq, lista = quick_sort_ordenado(lista, esq, indice - 1)
        comparacoes += comp_esq
        comp_dir, lista = quick_sort_ordenado(lista, indice + 1, dir)
```

```
        comparacoes += comp_dir
    else:
        comparacoes = 0
    return comparacoes, lista

def particao(lista, esq, dir):
    indice_pivo = (esq + dir) // 2
    pivo = lista[indice_pivo]
    comparacoes = 0
    # Particionamento
    i = esq
    j = dir
    while i <= j:
        while i <= dir and lista[i] <= pivo:
            i += 1
            comparacoes += 1
        while j >= esq and lista[j] > pivo:
            j -= 1
            comparacoes += 1
        if i < j:
            lista[i], lista[j] = lista[j], lista[i]
    # Posicionando o pivo no local correto
    lista[indice_pivo], lista[j] = lista[j], lista[indice_pivo]
    return j, comparacoes
```