# پروژه پنجم :‌

دانشجو سیدمحمدمهدی مصطفوی غمام

### **تحلیل کلی پروژه**

1. **ساختار پروژه:**
   * پروژه شامل یک سرور و سه کلاینت است.
   * کلاینت‌ها آرایه‌ای از اعداد را به سرور ارسال می‌کنند.
   * سرور باید آرایه‌های دریافتی را با استفاده از الگوریتم‌های مرتب‌سازی مختلف مرتب کند.
   * سپس نتیجه مرتب‌شده را به کلاینت‌ها بازگرداند.
2. **الگوریتم‌های مرتب‌سازی:**
   * **Stalin Sort:** الگوریتمی که عناصر نامرتب را حذف می‌کند.
   * **Bogo Sort:** الگوریتمی که به‌صورت تصادفی آرایه را مرتب می‌کند تا زمانی که به حالت مرتب برسد.
   * **Bubble Sort:** الگوریتمی ساده که با مقایسه جفت‌به‌جفت عناصر و جابجایی آن‌ها، آرایه را مرتب می‌کند.
3. **قابلیت‌های اضافی:**
   * سرور باید بتواند زمان اجرای هر الگوریتم را اندازه‌گیری کند و سریع‌ترین الگوریتم را به کاربر اعلام کند.
   * مدیریت خطاها و ثبت لاگ‌ها باید در برنامه پیاده‌سازی شود.

#### **1. پیاده‌سازی برنامه سوکت با سه کلاینت و یک سرور**

برای این بخش باید برنامه‌ای نوشته شود که:

* کلاینت‌ها آرایه‌های ورودی را به سرور ارسال کنند.
* سرور آرایه‌ها را دریافت کرده و آن‌ها را با الگوریتم‌های مختلف مرتب کند.
* سرور نتایج را به کلاینت‌ها بازگرداند.

**نمونه کد ساده برای سرور و کلاینت:**

**کد سرور:**

import socket

import threading

import time

import random

# الگوریتم‌های مرتب‌سازی

def stalin\_sort(arr):

if not arr:

return []

sorted\_list = [arr[0]]

for i in range(1, len(arr)):

if arr[i] >= sorted\_list[-1]:

sorted\_list.append(arr[i])

return sorted\_list

def is\_sorted(arr):

return all(arr[i] <= arr[i + 1] for i in range(len(arr) - 1))

def bogo\_sort(arr):

while not is\_sorted(arr):

random.shuffle(arr)

return arr

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

for j in range(0, n - i - 1):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

return arr

# تابع برای مدیریت کلاینت‌ها

def handle\_client(client\_socket, client\_address):

print(f"اتصال جدید: {client\_address}")

try:

# دریافت آرایه از کلاینت

data = client\_socket.recv(1024).decode()

array = list(map(int, data.split(',')))

print(f"آرایه دریافتی از {client\_address}: {array}")

# مرتب‌سازی با هر الگوریتم

results = {}

for sort\_name, sort\_func in [("Stalin Sort", stalin\_sort), ("Bogo Sort", bogo\_sort), ("Bubble Sort", bubble\_sort)]:

start\_time = time.time()

sorted\_array = sort\_func(array.copy())

elapsed\_time = time.time() - start\_time

results[sort\_name] = (sorted\_array, elapsed\_time)

# انتخاب سریع‌ترین الگوریتم

fastest\_sort = min(results.items(), key=lambda x: x[1][1])

fastest\_name, (fastest\_array, fastest\_time) = fastest\_sort

# ارسال نتیجه به کلاینت

response = f"Sorted Array: {fastest\_array}\nFastest Algorithm: {fastest\_name} ({fastest\_time:.6f} seconds)"

client\_socket.send(response.encode())

except Exception as e:

client\_socket.send(f"Error: {str(e)}".encode())

finally:

client\_socket.close()

# تنظیمات سرور

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(("0.0.0.0", 12345))

server.listen(5)

print("سرور در حال اجرا است...")

while True:

client\_socket, client\_address = server.accept()

client\_thread = threading.Thread(target=handle\_client, args=(client\_socket, client\_address))

client\_thread.start()

**کد کلاینت:**

# اتصال به سرور

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(("127.0.0.1", 12345))

# ارسال آرایه به سرور

array = input("آرایه خود را وارد کنید (اعداد را با کاما جدا کنید): ")

client.send(array.encode())

# دریافت نتیجه از سرور

response = client.recv(1024).decode()

print("پاسخ سرور:")

print(response)

# بستن اتصال

client.close()

#### **2. اندازه‌گیری زمان اجرای الگوریتم‌ها**

در کد سرور، زمان اجرای هر الگوریتم با استفاده از time.time() اندازه‌گیری شده و سریع‌ترین الگوریتم مشخص می‌شود:

start\_time = time.time()

sorted\_array = sort\_func(array.copy())

elapsed\_time = time.time() - start\_time

#### **3. مدیریت خطاها و ثبت لاگ‌ها**

برای مدیریت خطاها:

* از بلوک try-except در کد سرور استفاده شده است.
* در صورت بروز خطا، پیام خطا به کلاینت ارسال می‌شود.

برای ثبت لاگ‌ها:

* می‌توان از ماژول logging در پایتون استفاده کرد. به‌عنوان مثال:

import logging

logging.basicConfig(filename="server.log", level=logging.INFO, format="%(asctime)s - %(message)s")

# ثبت لاگ

logging.info(f"Array received from {client\_address}: {array}")

#### **4. توضیح الگوریتم‌ها**

* **Stalin Sort:** عناصر نامرتب را حذف می‌کند. پیچیدگی زمانی: O(n)O(n).
* **Bogo Sort:** به‌صورت تصادفی مرتب می‌کند. پیچیدگی زمانی: بسیار بالا (غیرعملی).
* **Bubble Sort:** جفت‌به‌جفت مقایسه و جابجا می‌کند. پیچیدگی زمانی: O(n2)O(n^2).

### **نتیجه‌گیری**

این پروژه شامل مفاهیم برنامه‌نویسی سوکت، الگوریتم‌های مرتب‌سازی، مدیریت خطاها و بهینه‌سازی عملکرد است. کدهای ارائه‌شده نمونه‌ای ساده برای پیاده‌سازی این مفاهیم هستند و می‌توان آن‌ها را با توجه به نیازهای پروژه گسترش داد.