"""

Assignment-A3

Problem Statement: Implement Greedy search algorithm for any of the following application:

I. Selection Sort

IV. Job Scheduling Problem

"""

# BEGINNING OF SELECTION SORT

numbers = [] # Empty list to store numbers

# Function to take input for numbers

def input\_numbers():

total = int(input("\nHow many numbers you wish to enter?\nTotal numbers:\t"))

for i in range(total):

val = float(input(f"Enter number {i+1}:\t"))

numbers.append(val)

print("\nNumbers you've entered are:\t", numbers)

# Function for selection sort

def selection\_sort():

for i in range(len(numbers)):

min\_index = i

for j in range(i+1, len(numbers)):

if numbers[j] < numbers[min\_index]:

min\_index = j

numbers[i], numbers[min\_index] = numbers[min\_index], numbers[i] # Swapping

print("Numbers sorted in ascending order using selection sort:\t", numbers)

# END OF SELECTION SORT

# BEGINNING OF JOB SCHEDULING

def job\_scheduling():

jobs = []

total = int(input("Total jobs to add:\t"))

# Take input for jobs

print("\n", "-"\*10, "JOBS", "-"\*10, "\n")

for i in range(total):

print(f"JOB {i+1} ->")

job\_id = int(input(f"ID for job {i+1}:\t\t"))

deadline = int(input(f"Deadline for job {i+1}:\t"))

profit = int(input(f"Profit for job {i+1}:\t"))

jobs.append((job\_id, deadline, profit)) # Index 0 for job\_id; Index 1 for deadline; Index 2 for profit

print(f"\nAdded {total} jobs.")

print("-"\*27, "\n")

# Initialize

jobs.sort(key=lambda x: x[2], reverse=True) # Sort jobs by profit; Using index 2 to access profit

max\_deadline = max(job[1] for job in jobs) # Highest deadline; Using index 1 to access deadline

slots = [0] \* (max\_deadline + 1)

total\_profit = 0

# Scheduling jobs using greedy strategy

for job in jobs:

for i in range(job[1], 0, -1):

if slots[i] == 0:

slots[i] = job[0]

total\_profit += job[2]

break

# Print scheduled jobs

print("Scheduled Jobs:", end=" ")

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

if slots[i] != 0:

print(slots[i], end=" ")

print(f"\nTotal Profit: {total\_profit}")

# END OF JOB SCHEDULING

# Main function for menu

def main():

while True:

print("\n\n", "-"\*10, "MAIN MENU", "-"\*10)

print("1. Selection Sort")

print("2. Job Scheduling")

print("3. Exit")

choice = int(input("Choose an option (1-3):\t"))

print("-"\*32)

if (choice == 1):

input\_numbers()

selection\_sort()

elif (choice == 2):

job\_scheduling()

elif (choice == 3):

print("\n## END OF CODE\n")

break

else:

print("\nPlease choose a valid option (1-3)")

main()

Here's a full **explanation** of your code and a set of **viva questions with answers** based on it:

**🔍 Explanation of the Code (Assignment-A3)**

**📌 Problem Statement:**

Implement Greedy search algorithm for:

1. **Selection Sort**
2. **Job Scheduling Problem**

**🧩 1. Selection Sort**

**➤ Concept:**

Selection sort is a greedy algorithm where we repeatedly **pick the smallest element** and place it in the correct position.

**➤ Code Breakdown:**

* **input\_numbers()**:
  + Takes user input to store numbers in a list.
  + Dynamically stores float numbers based on user count.
* **selection\_sort()**:
  + Implements selection sort:
    - Loop through the list.
    - For each position, find the **minimum element** from the remaining unsorted part.
    - **Swap** it with the current element.
  + This is greedy because it always **chooses the best minimum option at that step.**

**➤ Output:**

Sorted list in **ascending order**.

**🧩 2. Job Scheduling Problem**

**➤ Concept:**

Given jobs with **deadlines** and **profits**, schedule jobs to **maximize total profit** such that each job takes 1 unit time and is completed before its deadline.

**➤ Greedy Strategy:**

* Sort jobs by **descending profit**.
* Try to place each job in the **latest available slot** before its deadline.

**➤ Code Breakdown:**

* **Input**:
  + Takes job ID, deadline, and profit for each job.
  + Stores in a list jobs = [(job\_id, deadline, profit)].
* **Greedy Scheduling**:
  + Sorts jobs by profit.
  + Tries to assign each job to the latest possible available time slot before its deadline.
  + Tracks total profit and scheduled jobs.

**➤ Output:**

Scheduled job IDs and **maximum profit**.

**🖥️ Main Menu:**

Gives user a menu to choose between:

* Selection Sort
* Job Scheduling
* Exit

**🎤 Viva Questions with Answers**

**🔹 General**

1. **Q:** What is a Greedy algorithm?  
   **A:** A greedy algorithm makes the **locally optimal choice** at each step, hoping it leads to a global optimum.
2. **Q:** Which two problems are solved in your code using greedy approach?  
   **A:** Selection Sort and Job Scheduling Problem.

**🔹 Selection Sort**

1. **Q:** Why is Selection Sort considered greedy?  
   **A:** Because it selects the **minimum element** in each iteration greedily and places it in the correct position.
2. **Q:** What is the time complexity of selection sort?  
   **A:** **O(n²)** in all cases (best, average, worst).
3. **Q:** Can selection sort be optimized further like bubble sort?  
   **A:** No, because it always scans the unsorted portion of the list to find the minimum.

**🔹 Job Scheduling**

1. **Q:** What is the objective of the Job Scheduling Problem?  
   **A:** To schedule jobs such that the **total profit is maximized**, and no job is done after its deadline.
2. **Q:** Why do you sort the jobs by profit in descending order?  
   **A:** To ensure the **most profitable jobs are scheduled first**, maximizing total profit (greedy choice).
3. **Q:** What is the time complexity of your job scheduling implementation?  
   **A:** O(n²) in worst case due to nested loops, but sorting takes O(n log n).
4. **Q:** Can multiple jobs be scheduled at the same time?  
   **A:** No, each time slot can hold **only one job**.
5. **Q:** What happens if a job’s deadline is already filled?  
   **A:** The algorithm checks earlier slots. If none are free, that job is **skipped**.