**1.**

- Code:

*# Đọc dữ liệu từ file hoặc stdin*def read\_input():  
 N = int(input().split()[1]) *# Points N* distances = {}  
 input\_line = input().split()  
 while input\_line[0] != 'DISTANCES':  
 input\_line = input().split()  
 num\_distances = int(input\_line[1])  
  
 for \_ in range(num\_distances):  
 i, j, d = map(int, input().split())  
 if i not in distances:  
 distances[i] = {}  
 distances[i][j] = d  
  
 *# Đọc thông tin rơ mooc* trailer\_info = input().split()  
 while trailer\_info[0] != 'TRAILER':  
 trailer\_info = input().split()  
 trailer\_location = int(trailer\_info[1])  
 trailer\_attach\_time = int(trailer\_info[2])  
  
 *# Đọc thông tin đầu kéo* truck\_info = input().split()  
 while truck\_info[0] != 'TRUCK':  
 truck\_info = input().split()  
 m = int(truck\_info[1])  
 trucks = {}  
 for \_ in range(m):  
 truck\_id, location = map(int, input().split())  
 trucks[truck\_id] = {  
 'location': location,  
 'capacity': 2, *# Sức chứa tính theo đơn vị container 20ft* 'route': [],  
 'load': 0,  
 'time': 0,  
 }  
  
 *# Đọc yêu cầu vận chuyển* requests = {}  
 line = input()  
 while line.strip() != '#':  
 if line.startswith('REQ'):  
 parts = line.strip().split()  
 req\_id = int(parts[1])  
 size = int(parts[2])  
 p1 = int(parts[3])  
 pickup\_action = parts[4]  
 pickup\_duration = int(parts[5])  
 p2 = int(parts[6])  
 drop\_action = parts[7]  
 drop\_duration = int(parts[8])  
 requests[req\_id] = {  
 'size': size,  
 'pickup\_point': p1,  
 'pickup\_action': pickup\_action,  
 'pickup\_duration': pickup\_duration,  
 'drop\_point': p2,  
 'drop\_action': drop\_action,  
 'drop\_duration': drop\_duration,  
 'assigned': False,  
 }  
 line = input()  
  
 return N, distances, trailer\_location, trailer\_attach\_time, trucks, requests  
  
def assign\_requests(trucks, requests, distances, trailer\_location):  
 unassigned\_requests = list(requests.keys())  
  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 current\_location = truck['location']  
 *# Đầu tiên, đầu kéo đi lấy rơ mooc* truck['route'].append({  
 'point': trailer\_location,  
 'action': 'PICKUP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[current\_location][trailer\_location] + trailer\_attach\_time  
 current\_location = trailer\_location  
 *# Khởi tạo tải trọng* truck['load'] = 0  
  
 while unassigned\_requests:  
 *# Tìm yêu cầu gần nhất* nearest\_req = None  
 min\_distance = float('inf')  
 for req\_id in unassigned\_requests:  
 req = requests[req\_id]  
 if req['size'] == 20 and truck['load'] + 1 <= 2:  
 size\_unit = 1  
 elif req['size'] == 40 and truck['load'] + 2 <= 2:  
 size\_unit = 2  
 else:  
 continue *# Không đủ sức chứa* distance = distances[current\_location][req['pickup\_point']]  
 if distance < min\_distance:  
 min\_distance = distance  
 nearest\_req = req\_id  
  
 if nearest\_req is None:  
 break *# Không còn yêu cầu phù hợp  
  
 # Gán yêu cầu cho đầu kéo* req = requests[nearest\_req]  
 truck['route'].append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': nearest\_req  
 })  
 truck['time'] += distances[current\_location][req['pickup\_point']] + req['pickup\_duration']  
 current\_location = req['pickup\_point']  
 truck['load'] += 1 if req['size'] == 20 else 2  
  
 *# Thực hiện giao hàng* truck['route'].append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': nearest\_req  
 })  
 truck['time'] += distances[current\_location][req['drop\_point']] + req['drop\_duration']  
 current\_location = req['drop\_point']  
 truck['load'] -= 1 if req['size'] == 20 else 2  
  
 unassigned\_requests.remove(nearest\_req)  
 requests[nearest\_req]['assigned'] = True  
  
 *# Trả rơ mooc về bãi nếu cần* truck['route'].append({  
 'point': trailer\_location,  
 'action': 'DROP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[current\_location][trailer\_location] + trailer\_attach\_time  
 current\_location = trailer\_location  
  
 *# Quay về bãi của đầu kéo* truck['route'].append({  
 'point': truck['location'],  
 'action': 'STOP',  
 'request\_id': None  
 })  
 truck['time'] += distances[current\_location][truck['location']]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 N, distances, trailer\_location, trailer\_attach\_time, trucks, requests = read\_input()  
 assign\_requests(trucks, requests, distances, trailer\_location)  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 print(f'TRUCK {truck\_id}')  
 print(truck['time'])  
 for step in truck['route']:  
 print(f'{step["point"]} {step["action"]}')  
 print()

- Input: Input mẫu trên Hustack

- Output:

TRUCK 1

11100

3 PICKUP\_TRAILER

1 PICKUP\_CONTAINER

5 DROP\_CONTAINER

5 PICKUP\_CONTAINER\_TRAILER

7 DROP\_CONTAINER

7 PICKUP\_CONTAINER\_TRAILER

9 DROP\_CONTAINER

8 PICKUP\_CONTAINER

6 DROP\_CONTAINER\_TRAILER

6 PICKUP\_CONTAINER

2 DROP\_CONTAINER

3 DROP\_TRAILER

4 STOP

TRUCK 2

1800

3 PICKUP\_TRAILER

3 DROP\_TRAILER

4 STOP

TRUCK 3

1800

3 PICKUP\_TRAILER

3 DROP\_TRAILER

4 STOP

Process finished with exit code 0

**2.**

- Code:

*# Đọc dữ liệu từ file hoặc stdin*def read\_input():  
 N = int(input().split()[1]) *# Points N* distances = {}  
 input\_line = input().split()  
 while input\_line[0] != 'DISTANCES':  
 input\_line = input().split()  
 num\_distances = int(input\_line[1])  
  
 for \_ in range(num\_distances):  
 i, j, d = map(int, input().split())  
 if i not in distances:  
 distances[i] = {}  
 distances[i][j] = d  
  
 *# Đọc thông tin rơ mooc* trailer\_info = input().split()  
 while trailer\_info[0] != 'TRAILER':  
 trailer\_info = input().split()  
 trailer\_location = int(trailer\_info[1])  
 trailer\_attach\_time = int(trailer\_info[2])  
  
 *# Đọc thông tin đầu kéo* truck\_info = input().split()  
 while truck\_info[0] != 'TRUCK':  
 truck\_info = input().split()  
 m = int(truck\_info[1])  
 trucks = {}  
 for \_ in range(m):  
 truck\_id, location = map(int, input().split())  
 trucks[truck\_id] = {  
 'location': location,  
 'capacity': 2, *# Sức chứa tính theo đơn vị container 20ft* 'route': [],  
 'load': 0,  
 'time': 0,  
 }  
  
 *# Đọc yêu cầu vận chuyển* requests = {}  
 line = input()  
 while line.strip() != '#':  
 if line.startswith('REQ'):  
 parts = line.strip().split()  
 req\_id = int(parts[1])  
 size = int(parts[2])  
 p1 = int(parts[3])  
 pickup\_action = parts[4]  
 pickup\_duration = int(parts[5])  
 p2 = int(parts[6])  
 drop\_action = parts[7]  
 drop\_duration = int(parts[8])  
 requests[req\_id] = {  
 'size': size,  
 'pickup\_point': p1,  
 'pickup\_action': pickup\_action,  
 'pickup\_duration': pickup\_duration,  
 'drop\_point': p2,  
 'drop\_action': drop\_action,  
 'drop\_duration': drop\_duration,  
 'assigned': False,  
 }  
 line = input()  
  
 return N, distances, trailer\_location, trailer\_attach\_time, trucks, requests  
  
def assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time):  
 unassigned\_requests = list(requests.keys())  
  
 *# Khởi tạo trạng thái cho các đầu kéo* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 *# Đầu tiên, đầu kéo đi lấy rơ mooc* truck['route'] = []  
 truck['time'] = 0  
 truck['load'] = 0  
 truck['current\_location'] = truck['location']  
 truck['route'].append({  
 'point': trailer\_location,  
 'action': 'PICKUP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 truck['current\_location'] = trailer\_location  
  
 *# Sử dụng chiến lược round-robin để gán yêu cầu cho các đầu kéo* truck\_ids = list(trucks.keys())  
 num\_trucks = len(truck\_ids)  
 request\_index = 0  
  
 while unassigned\_requests:  
 req\_id = unassigned\_requests.pop(0)  
 req = requests[req\_id]  
  
 *# Tìm đầu kéo phù hợp nhất để phục vụ yêu cầu này* best\_truck\_id = None  
 min\_additional\_time = float('inf')  
  
 for truck\_id in truck\_ids:  
 truck = trucks[truck\_id]  
 *# Kiểm tra sức chứa* required\_capacity = 1 if req['size'] == 20 else 2  
 if truck['load'] + required\_capacity > 2:  
 continue *# Không đủ sức chứa  
  
 # Tính toán thời gian tăng thêm nếu đầu kéo này phục vụ yêu cầu* time\_to\_pickup = distances[truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 time\_to\_drop = distances[req['pickup\_point']][req['drop\_point']] + req['drop\_duration']  
 total\_additional\_time = time\_to\_pickup + time\_to\_drop  
  
 if total\_additional\_time < min\_additional\_time:  
 min\_additional\_time = total\_additional\_time  
 best\_truck\_id = truck\_id  
  
 if best\_truck\_id is None:  
 *# Không có đầu kéo nào có thể phục vụ yêu cầu này (do sức chứa)* continue  
  
 *# Gán yêu cầu cho đầu kéo tốt nhất* truck = trucks[best\_truck\_id]  
 *# Đi đến điểm lấy container* truck['route'].append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': req\_id  
 })  
 truck['time'] += distances[truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 truck['current\_location'] = req['pickup\_point']  
 truck['load'] += required\_capacity  
  
 *# Đi đến điểm hạ container* truck['route'].append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': req\_id  
 })  
 truck['time'] += distances[truck['current\_location']][req['drop\_point']] + req['drop\_duration']  
 truck['current\_location'] = req['drop\_point']  
 truck['load'] -= required\_capacity *# Giả sử container được dỡ xuống* requests[req\_id]['assigned'] = True  
  
 *# Sau khi gán xong tất cả yêu cầu, đầu kéo trả rơ mooc về bãi và quay về bãi đỗ của mình* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 *# Trả rơ mooc về bãi nếu cần* if truck['current\_location'] != trailer\_location:  
 truck['route'].append({  
 'point': trailer\_location,  
 'action': 'DROP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 truck['current\_location'] = trailer\_location  
 else:  
 *# Nếu đã ở bãi rơ mooc, chỉ cần tháo rơ mooc* truck['route'].append({  
 'point': trailer\_location,  
 'action': 'DROP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += trailer\_attach\_time  
  
 *# Quay về bãi của đầu kéo* truck['route'].append({  
 'point': truck['location'],  
 'action': 'STOP',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][truck['location']]  
 truck['current\_location'] = truck['location']  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 N, distances, trailer\_location, trailer\_attach\_time, trucks, requests = read\_input()  
 assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time)  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 print(f'{truck\_id} {truck["time"]}')  
 for step in truck['route']:  
 print(f'{step["point"]} {step["action"]}')

- Input: Mẫu trên Hustack

- Output:

1 12300

3 PICKUP\_TRAILER

1 PICKUP\_CONTAINER

5 DROP\_CONTAINER

5 PICKUP\_CONTAINER\_TRAILER

7 DROP\_CONTAINER

8 PICKUP\_CONTAINER

6 DROP\_CONTAINER\_TRAILER

7 PICKUP\_CONTAINER\_TRAILER

9 DROP\_CONTAINER

6 PICKUP\_CONTAINER

2 DROP\_CONTAINER

3 DROP\_TRAILER

4 STOP

2 1800

3 PICKUP\_TRAILER

3 DROP\_TRAILER

4 STOP

3 1800

3 PICKUP\_TRAILER

3 DROP\_TRAILER

4 STOP

Process finished with exit code 0

**3.**

- Code:

*# Đọc dữ liệu từ file hoặc stdin*def read\_input():  
 N = int(input().split()[1]) *# Points N* distances = {}  
 input\_line = input().split()  
 while input\_line[0] != 'DISTANCES':  
 input\_line = input().split()  
 num\_distances = int(input\_line[1])  
  
 for \_ in range(num\_distances):  
 i, j, d = map(int, input().split())  
 if i not in distances:  
 distances[i] = {}  
 distances[i][j] = d  
  
 *# Đọc thông tin rơ mooc* trailer\_info = input().split()  
 while trailer\_info[0] != 'TRAILER':  
 trailer\_info = input().split()  
 trailer\_location = int(trailer\_info[1])  
 trailer\_attach\_time = int(trailer\_info[2])  
  
 *# Đọc thông tin đầu kéo* truck\_info = input().split()  
 while truck\_info[0] != 'TRUCK':  
 truck\_info = input().split()  
 m = int(truck\_info[1])  
 trucks = {}  
 for \_ in range(m):  
 truck\_id, location = map(int, input().split())  
 trucks[truck\_id] = {  
 'location': location,  
 'capacity': 2, *# Sức chứa tính theo đơn vị container 20ft* 'route': [],  
 'load': 0,  
 'time': 0,  
 }  
  
 *# Đọc yêu cầu vận chuyển* requests = {}  
 line = input()  
 while line.strip() != '#':  
 if line.startswith('REQ'):  
 parts = line.strip().split()  
 req\_id = int(parts[1])  
 size = int(parts[2])  
 p1 = int(parts[3])  
 pickup\_action = parts[4]  
 pickup\_duration = int(parts[5])  
 p2 = int(parts[6])  
 drop\_action = parts[7]  
 drop\_duration = int(parts[8])  
 requests[req\_id] = {  
 'size': size,  
 'pickup\_point': p1,  
 'pickup\_action': pickup\_action,  
 'pickup\_duration': pickup\_duration,  
 'drop\_point': p2,  
 'drop\_action': drop\_action,  
 'drop\_duration': drop\_duration,  
 'assigned': False,  
 }  
 line = input()  
  
 return N, distances, trailer\_location, trailer\_attach\_time, trucks, requests  
  
def assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time):  
 unassigned\_requests = list(requests.keys())  
  
 *# Khởi tạo trạng thái cho các đầu kéo* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 truck['route'] = []  
 truck['time'] = 0  
 truck['load'] = 0  
 truck['current\_location'] = truck['location']  
 truck['has\_trailer'] = False  
  
 *# Gán yêu cầu* while unassigned\_requests:  
 req\_id = unassigned\_requests.pop(0)  
 req = requests[req\_id]  
  
 *# Tìm đầu kéo có thời gian hoàn thành dự kiến nhỏ nhất* best\_truck\_id = None  
 min\_completion\_time = float('inf')  
  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 *# Kiểm tra sức chứa* required\_capacity = 1 if req['size'] == 20 else 2  
 if truck['load'] + required\_capacity > 2:  
 continue *# Không đủ sức chứa  
  
 # Tính thời gian hoàn thành dự kiến nếu gán yêu cầu này* temp\_truck = truck.copy()  
 temp\_truck['route'] = truck['route'][:]  
 temp\_truck['load'] = truck['load']  
 temp\_truck['current\_location'] = truck['current\_location']  
 temp\_truck['time'] = truck['time']  
 temp\_truck['has\_trailer'] = truck['has\_trailer']  
  
 *# Nếu chưa có rơ mooc, đi lấy* if not temp\_truck['has\_trailer']:  
 temp\_truck['route'].append({  
 'point': trailer\_location,  
 'action': 'PICKUP\_TRAILER',  
 'request\_id': None  
 })  
 temp\_truck['time'] += distances[temp\_truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 temp\_truck['current\_location'] = trailer\_location  
 temp\_truck['has\_trailer'] = True  
  
 *# Đi đến điểm lấy container* temp\_truck['time'] += distances[temp\_truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 temp\_truck['current\_location'] = req['pickup\_point']  
 temp\_truck['load'] += required\_capacity  
  
 *# Đi đến điểm hạ container* temp\_truck['time'] += distances[temp\_truck['current\_location']][req['drop\_point']] + req['drop\_duration']  
 temp\_truck['current\_location'] = req['drop\_point']  
 temp\_truck['load'] -= required\_capacity  
  
 *# Thời gian hoàn thành dự kiến* completion\_time = temp\_truck['time']  
  
 if completion\_time < min\_completion\_time:  
 min\_completion\_time = completion\_time  
 best\_truck\_id = truck\_id  
  
 if best\_truck\_id is not None:  
 *# Gán yêu cầu cho đầu kéo tốt nhất* truck = trucks[best\_truck\_id]  
 *# Nếu chưa có rơ mooc, đi lấy* if not truck['has\_trailer']:  
 truck['route'].append({  
 'point': trailer\_location,  
 'action': 'PICKUP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 truck['current\_location'] = trailer\_location  
 truck['has\_trailer'] = True  
  
 *# Đi đến điểm lấy container* truck['route'].append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': req\_id  
 })  
 truck['time'] += distances[truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 truck['current\_location'] = req['pickup\_point']  
 truck['load'] += required\_capacity  
  
 *# Đi đến điểm hạ container* truck['route'].append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': req\_id  
 })  
 truck['time'] += distances[truck['current\_location']][req['drop\_point']] + req['drop\_duration']  
 truck['current\_location'] = req['drop\_point']  
 truck['load'] -= required\_capacity  
  
 requests[req\_id]['assigned'] = True  
  
 *# Hoàn thành lộ trình cho các đầu kéo* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 if truck['has\_trailer']:  
 *# Trả rơ mooc* truck['route'].append({  
 'point': trailer\_location,  
 'action': 'DROP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 truck['current\_location'] = trailer\_location  
 truck['has\_trailer'] = False  
  
 *# Quay về bãi* truck['route'].append({  
 'point': truck['location'],  
 'action': 'STOP',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][truck['location']]  
 truck['current\_location'] = truck['location']  
  
def calculate\_objective(trucks, distances):  
 F1 = 0 *# Makespan* F2 = 0 *# Tổng thời gian di chuyển* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 F1 = max(F1, truck['time'])  
 *# Tính tổng thời gian di chuyển* total\_distance = 0  
 route = truck['route']  
 prev\_point = truck['location']  
 for step in route:  
 point = step['point']  
 total\_distance += distances[prev\_point][point]  
 prev\_point = point  
 F2 += total\_distance  
 return F1, F2  
  
def output\_result(trucks):  
 print(f'ROUTES {len(trucks)}')  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 print(f'TRUCK {truck\_id}')  
 for step in truck['route']:  
 point = step['point']  
 action = step['action']  
 if step['request\_id'] is not None:  
 request\_id = step['request\_id']  
 print(f'{point} {action} {request\_id}')  
 else:  
 print(f'{point} {action}')  
 print('#')  
  
def calculate\_truck\_time(truck, distances):  
 total\_time = 0  
 prev\_point = truck['location']  
 for step in truck['route']:  
 point = step['point']  
 action = step['action']  
 total\_time += distances[prev\_point][point]  
 if action == 'PICKUP\_TRAILER' or action == 'DROP\_TRAILER':  
 total\_time += trailer\_attach\_time  
 elif action in ['PICKUP\_CONTAINER', 'DROP\_CONTAINER', 'PICKUP\_CONTAINER\_TRAILER', 'DROP\_CONTAINER\_TRAILER']:  
 request\_id = step['request\_id']  
 req = requests[request\_id]  
 if action == req['pickup\_action']:  
 total\_time += req['pickup\_duration']  
 elif action == req['drop\_action']:  
 total\_time += req['drop\_duration']  
 prev\_point = point  
 return total\_time  
  
def is\_feasible(truck, req, action, capacities):  
 *# Kiểm tra sức chứa* required\_capacity = 1 if req['size'] == 20 else 2  
 if action == 'add':  
 if truck['load'] + required\_capacity > capacities:  
 return False  
 elif action == 'remove':  
 if truck['load'] - required\_capacity < 0:  
 return False  
 return True  
  
def update\_truck\_route(truck, new\_route):  
 truck['route'] = new\_route  
 *# Cập nhật lại thời gian và tải trọng* truck['time'] = calculate\_truck\_time(truck, distances)  
 *# Tải trọng được cập nhật trong quá trình tính toán lộ trình*def swap\_requests(trucks, requests, distances):  
 improved = True  
 while improved:  
 improved = False  
 for truck\_id1 in trucks:  
 for truck\_id2 in trucks:  
 if truck\_id1 >= truck\_id2:  
 continue  
 truck1 = trucks[truck\_id1]  
 truck2 = trucks[truck\_id2]  
 *# Lấy danh sách các yêu cầu của mỗi đầu kéo* reqs\_truck1 = [step['request\_id'] for step in truck1['route'] if step['request\_id'] is not None]  
 reqs\_truck2 = [step['request\_id'] for step in truck2['route'] if step['request\_id'] is not None]  
 *# Thử hoán đổi từng cặp yêu cầu* for req\_id1 in reqs\_truck1:  
 for req\_id2 in reqs\_truck2:  
 req1 = requests[req\_id1]  
 req2 = requests[req\_id2]  
 *# Kiểm tra tính khả thi của việc hoán đổi* if (is\_feasible(truck1, req2, 'add', 2) and  
 is\_feasible(truck1, req1, 'remove', 2) and  
 is\_feasible(truck2, req1, 'add', 2) and  
 is\_feasible(truck2, req2, 'remove', 2)):  
 *# Tạo bản sao của lộ trình để thử nghiệm* new\_route1 = replace\_request\_in\_route(truck1['route'], req\_id1, req2)  
 new\_route2 = replace\_request\_in\_route(truck2['route'], req\_id2, req1)  
 *# Tính toán thời gian mới* temp\_truck1 = truck1.copy()  
 temp\_truck1['route'] = new\_route1  
 time1 = calculate\_truck\_time(temp\_truck1, distances)  
 temp\_truck2 = truck2.copy()  
 temp\_truck2['route'] = new\_route2  
 time2 = calculate\_truck\_time(temp\_truck2, distances)  
 *# Kiểm tra xem makespan có giảm không* old\_makespan = max(truck1['time'], truck2['time'])  
 new\_makespan = max(time1, time2)  
 if new\_makespan < old\_makespan:  
 *# Chấp nhận hoán đổi* update\_truck\_route(truck1, new\_route1)  
 update\_truck\_route(truck2, new\_route2)  
 improved = True  
 break *# Có cải thiện, tiếp tục vòng lặp chính* if improved:  
 break  
 if improved:  
 break  
 if improved:  
 break  
  
def replace\_request\_in\_route(route, old\_req\_id, new\_req):  
 new\_route = []  
 for step in route:  
 if step['request\_id'] == old\_req\_id:  
 *# Thay thế bằng yêu cầu mới* if step['action'] == requests[old\_req\_id]['pickup\_action']:  
 action = new\_req['pickup\_action']  
 duration = new\_req['pickup\_duration']  
 elif step['action'] == requests[old\_req\_id]['drop\_action']:  
 action = new\_req['drop\_action']  
 duration = new\_req['drop\_duration']  
 new\_step = {  
 'point': step['point'], *# Giữ nguyên điểm* 'action': action,  
 'request\_id': new\_req['id']  
 }  
 new\_route.append(new\_step)  
 else:  
 new\_route.append(step)  
 return new\_route  
  
def create\_steps\_for\_request(req\_id, req):  
 steps = []  
 steps.append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': req\_id  
 })  
 steps.append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': req\_id  
 })  
 return steps  
  
def is\_feasible\_route(truck, capacities):  
 load = 0  
 for step in truck['route']:  
 action = step['action']  
 req\_id = step['request\_id']  
 if action in ['PICKUP\_CONTAINER', 'PICKUP\_CONTAINER\_TRAILER']:  
 req = requests[req\_id]  
 required\_capacity = 1 if req['size'] == 20 else 2  
 load += required\_capacity  
 if load > capacities:  
 return False  
 elif action in ['DROP\_CONTAINER', 'DROP\_CONTAINER\_TRAILER']:  
 req = requests[req\_id]  
 required\_capacity = 1 if req['size'] == 20 else 2  
 load -= required\_capacity  
 return True  
  
def insert\_unassigned\_requests(trucks, requests, distances):  
 unassigned\_reqs = [req\_id for req\_id, req in requests.items() if not req['assigned']]  
 for req\_id in unassigned\_reqs:  
 req = requests[req\_id]  
 best\_truck\_id = None  
 best\_insertion = None  
 min\_increase = float('inf')  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 *# Thử chèn yêu cầu vào các vị trí khác nhau trong lộ trình* for i in range(len(truck['route']) + 1):  
 for j in range(i, len(truck['route']) + 1):  
 new\_route = truck['route'][:i] + \  
 create\_steps\_for\_request(req\_id, req) + \  
 truck['route'][i:]  
 temp\_truck = truck.copy()  
 temp\_truck['route'] = new\_route  
 *# Kiểm tra tính khả thi* if is\_feasible\_route(temp\_truck, 2):  
 time\_increase = calculate\_truck\_time(temp\_truck, distances) - truck['time']  
 if time\_increase < min\_increase:  
 min\_increase = time\_increase  
 best\_truck\_id = truck\_id  
 best\_insertion = new\_route  
 if best\_truck\_id is not None:  
 *# Chèn yêu cầu vào lộ trình của đầu kéo tốt nhất* truck = trucks[best\_truck\_id]  
 update\_truck\_route(truck, best\_insertion)  
 requests[req\_id]['assigned'] = True  
  
def improve\_solution(trucks, requests, distances):  
 previous\_makespan = None  
 while True:  
 swap\_requests(trucks, requests, distances)  
 insert\_unassigned\_requests(trucks, requests, distances)  
 *# Tính toán makespan mới* F1, F2 = calculate\_objective(trucks, distances)  
 if previous\_makespan is not None and F1 >= previous\_makespan:  
 break *# Không cải thiện được nữa* previous\_makespan = F1  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 *# Đọc dữ liệu đầu vào* N, distances, trailer\_location, trailer\_attach\_time, trucks, requests = read\_input()  
  
 *# Khởi tạo giải pháp ban đầu* assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time)  
  
 *# Cải thiện giải pháp* improve\_solution(trucks, requests, distances)  
  
 *# Tính toán hàm mục tiêu* F1, F2 = calculate\_objective(trucks, distances)  
 F = 10000 \* F1 + F2  
 print(f'OBJECTIVE {F}')  
  
 *# Xuất kết quả* output\_result(trucks)

- Input: Mẫu trên Hustack

- Output:

OBJECTIVE 60010200

ROUTES 3

TRUCK 1

3 PICKUP\_TRAILER

1 PICKUP\_CONTAINER 1

5 DROP\_CONTAINER 1

3 DROP\_TRAILER

4 STOP

#

TRUCK 2

3 PICKUP\_TRAILER

5 PICKUP\_CONTAINER\_TRAILER 2

7 DROP\_CONTAINER 2

7 PICKUP\_CONTAINER\_TRAILER 4

9 DROP\_CONTAINER 4

3 DROP\_TRAILER

4 STOP

#

TRUCK 3

3 PICKUP\_TRAILER

8 PICKUP\_CONTAINER 3

6 DROP\_CONTAINER\_TRAILER 3

6 PICKUP\_CONTAINER 5

2 DROP\_CONTAINER 5

3 DROP\_TRAILER

4 STOP

#

Process finished with exit code 0

**4.**

- Code:

*# Đọc dữ liệu từ file hoặc stdin*def read\_input():  
 N = int(input().split()[1]) *# Points N* distances = {}  
 input\_line = input().split()  
 while input\_line[0] != 'DISTANCES':  
 input\_line = input().split()  
 num\_distances = int(input\_line[1])  
  
 for \_ in range(num\_distances):  
 i, j, d = map(int, input().split())  
 if i not in distances:  
 distances[i] = {}  
 distances[i][j] = d  
  
 *# Đọc thông tin rơ mooc* trailer\_info = input().split()  
 while trailer\_info[0] != 'TRAILER':  
 trailer\_info = input().split()  
 trailer\_location = int(trailer\_info[1])  
 trailer\_attach\_time = int(trailer\_info[2])  
  
 *# Đọc thông tin đầu kéo* truck\_info = input().split()  
 while truck\_info[0] != 'TRUCK':  
 truck\_info = input().split()  
 m = int(truck\_info[1])  
 trucks = {}  
 for \_ in range(m):  
 truck\_id, location = map(int, input().split())  
 trucks[truck\_id] = {  
 'location': location,  
 'capacity': 2, *# Sức chứa tính theo đơn vị container 20ft* 'route': [],  
 'load': 0,  
 'time': 0,  
 }  
  
 *# Đọc yêu cầu vận chuyển* requests = {}  
 line = input()  
 while line.strip() != '#':  
 if line.startswith('REQ'):  
 parts = line.strip().split()  
 req\_id = int(parts[1])  
 size = int(parts[2])  
 p1 = int(parts[3])  
 pickup\_action = parts[4]  
 pickup\_duration = int(parts[5])  
 p2 = int(parts[6])  
 drop\_action = parts[7]  
 drop\_duration = int(parts[8])  
 requests[req\_id] = {  
 'size': size,  
 'pickup\_point': p1,  
 'pickup\_action': pickup\_action,  
 'pickup\_duration': pickup\_duration,  
 'drop\_point': p2,  
 'drop\_action': drop\_action,  
 'drop\_duration': drop\_duration,  
 'assigned': False,  
 }  
 line = input()  
  
 return N, distances, trailer\_location, trailer\_attach\_time, trucks, requests  
  
def assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time):  
 unassigned\_requests = list(requests.keys())  
  
 *# Khởi tạo trạng thái cho các đầu kéo* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 truck['route'] = []  
 truck['time'] = 0  
 truck['load'] = 0 *# Tải trọng của đầu kéo (không dùng vì container nằm trên rơ mooc)* truck['current\_location'] = truck['location']  
 truck['has\_trailer'] = False  
 truck['trailer\_load'] = 0 *# Tải trọng của rơ mooc  
  
 # Gán yêu cầu* while unassigned\_requests:  
 req\_id = unassigned\_requests.pop(0)  
 req = requests[req\_id]  
  
 *# Tìm đầu kéo có thời gian hoàn thành dự kiến nhỏ nhất* best\_truck\_id = None  
 min\_completion\_time = float('inf')  
  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 temp\_truck = truck.copy()  
 temp\_truck['route'] = truck['route'][:]  
 temp\_truck['current\_location'] = truck['current\_location']  
 temp\_truck['time'] = truck['time']  
 temp\_truck['has\_trailer'] = truck['has\_trailer']  
 temp\_truck['trailer\_load'] = truck['trailer\_load']  
  
 actions = []  
 required\_capacity = 1 if req['size'] == 20 else 2  
  
 *# Nếu yêu cầu là PICKUP\_CONTAINER\_TRAILER* if req['pickup\_action'] == 'PICKUP\_CONTAINER\_TRAILER':  
 if temp\_truck['has\_trailer']:  
 continue *# Không thể thực hiện nếu đã có rơ mooc  
 # Đi đến điểm pickup và gắn rơ mooc có container* actions.append({  
 'point': req['pickup\_point'],  
 'action': 'PICKUP\_CONTAINER\_TRAILER',  
 'request\_id': req\_id  
 })  
 temp\_truck['time'] += distances[temp\_truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 temp\_truck['current\_location'] = req['pickup\_point']  
 temp\_truck['has\_trailer'] = True  
 temp\_truck['trailer\_load'] = required\_capacity  
 else:  
 *# Nếu chưa có rơ mooc, cần đi lấy* if not temp\_truck['has\_trailer']:  
 *# Đi lấy rơ mooc* actions.append({  
 'point': trailer\_location,  
 'action': 'PICKUP\_TRAILER',  
 'request\_id': None  
 })  
 temp\_truck['time'] += distances[temp\_truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 temp\_truck['current\_location'] = trailer\_location  
 temp\_truck['has\_trailer'] = True  
 temp\_truck['trailer\_load'] = 0  
  
 *# Đi đến điểm pickup và nhận container* if temp\_truck['trailer\_load'] + required\_capacity > 2:  
 continue *# Không đủ sức chứa* actions.append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': req\_id  
 })  
 temp\_truck['time'] += distances[temp\_truck['current\_location']][req['pickup\_point']] + req['pickup\_duration']  
 temp\_truck['current\_location'] = req['pickup\_point']  
 temp\_truck['trailer\_load'] += required\_capacity  
  
 *# Đi đến điểm dropoff* actions.append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': req\_id  
 })  
 temp\_truck['time'] += distances[temp\_truck['current\_location']][req['drop\_point']] + req['drop\_duration']  
 temp\_truck['current\_location'] = req['drop\_point']  
  
 *# Cập nhật trạng thái sau dropoff* if req['drop\_action'] == 'DROP\_CONTAINER\_TRAILER':  
 temp\_truck['has\_trailer'] = False  
 temp\_truck['trailer\_load'] = 0  
 else:  
 temp\_truck['trailer\_load'] -= required\_capacity  
  
 *# Thời gian hoàn thành dự kiến* completion\_time = temp\_truck['time']  
  
 if completion\_time < min\_completion\_time:  
 min\_completion\_time = completion\_time  
 best\_truck\_id = truck\_id  
 best\_actions = actions  
  
 if best\_truck\_id is not None:  
 *# Gán yêu cầu cho đầu kéo tốt nhất* truck = trucks[best\_truck\_id]  
 for action in best\_actions:  
 truck['route'].append(action)  
 truck['time'] = min\_completion\_time  
 truck['current\_location'] = temp\_truck['current\_location']  
 truck['has\_trailer'] = temp\_truck['has\_trailer']  
 truck['trailer\_load'] = temp\_truck['trailer\_load']  
 requests[req\_id]['assigned'] = True  
 else:  
 *# Không thể gán yêu cầu này* print(f"Không thể gán yêu cầu {req\_id}")  
 continue  
  
 *# Hoàn thành lộ trình cho các đầu kéo* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 if truck['has\_trailer']:  
 *# Trả rơ mooc* truck['route'].append({  
 'point': trailer\_location,  
 'action': 'DROP\_TRAILER',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][trailer\_location] + trailer\_attach\_time  
 truck['current\_location'] = trailer\_location  
 truck['has\_trailer'] = False  
 truck['trailer\_load'] = 0  
  
 *# Quay về bãi* truck['route'].append({  
 'point': truck['location'],  
 'action': 'STOP',  
 'request\_id': None  
 })  
 truck['time'] += distances[truck['current\_location']][truck['location']]  
 truck['current\_location'] = truck['location']  
  
def calculate\_objective(trucks, distances):  
 F1 = 0 *# Makespan* F2 = 0 *# Tổng thời gian di chuyển* for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 F1 = max(F1, truck['time'])  
 *# Tính tổng thời gian di chuyển* total\_distance = 0  
 route = truck['route']  
 prev\_point = truck['location']  
 for step in route:  
 point = step['point']  
 total\_distance += distances[prev\_point][point]  
 prev\_point = point  
 F2 += total\_distance  
 return F1, F2  
  
def output\_result(trucks):  
 print(f'ROUTES {len(trucks)}')  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 print(f'TRUCK {truck\_id}')  
 for step in truck['route']:  
 point = step['point']  
 action = step['action']  
 if step['request\_id'] is not None:  
 request\_id = step['request\_id']  
 print(f'{point} {action} {request\_id}')  
 else:  
 print(f'{point} {action}')  
 print('#')  
  
def calculate\_truck\_time(truck, distances):  
 total\_time = 0  
 prev\_point = truck['location']  
 for step in truck['route']:  
 point = step['point']  
 action = step['action']  
 total\_time += distances[prev\_point][point]  
 if action == 'PICKUP\_TRAILER' or action == 'DROP\_TRAILER':  
 total\_time += trailer\_attach\_time  
 elif action in ['PICKUP\_CONTAINER', 'DROP\_CONTAINER', 'PICKUP\_CONTAINER\_TRAILER', 'DROP\_CONTAINER\_TRAILER']:  
 request\_id = step['request\_id']  
 req = requests[request\_id]  
 if action == req['pickup\_action']:  
 total\_time += req['pickup\_duration']  
 elif action == req['drop\_action']:  
 total\_time += req['drop\_duration']  
 prev\_point = point  
 return total\_time  
  
def is\_feasible(truck, req, action, capacities):  
 *# Kiểm tra sức chứa* required\_capacity = 1 if req['size'] == 20 else 2  
 if action == 'add':  
 if truck['load'] + required\_capacity > capacities:  
 return False  
 elif action == 'remove':  
 if truck['load'] - required\_capacity < 0:  
 return False  
 return True  
  
def update\_truck\_route(truck, new\_route):  
 truck['route'] = new\_route  
 *# Cập nhật lại thời gian và tải trọng* truck['time'] = calculate\_truck\_time(truck, distances)  
 *# Tải trọng được cập nhật trong quá trình tính toán lộ trình*def swap\_requests(trucks, requests, distances):  
 improved = True  
 while improved:  
 improved = False  
 for truck\_id1 in trucks:  
 for truck\_id2 in trucks:  
 if truck\_id1 >= truck\_id2:  
 continue  
 truck1 = trucks[truck\_id1]  
 truck2 = trucks[truck\_id2]  
 *# Lấy danh sách các yêu cầu của mỗi đầu kéo* reqs\_truck1 = [step['request\_id'] for step in truck1['route'] if step['request\_id'] is not None]  
 reqs\_truck2 = [step['request\_id'] for step in truck2['route'] if step['request\_id'] is not None]  
 *# Thử hoán đổi từng cặp yêu cầu* for req\_id1 in reqs\_truck1:  
 for req\_id2 in reqs\_truck2:  
 req1 = requests[req\_id1]  
 req2 = requests[req\_id2]  
 *# Kiểm tra tính khả thi của việc hoán đổi* if (is\_feasible(truck1, req2, 'add', 2) and  
 is\_feasible(truck1, req1, 'remove', 2) and  
 is\_feasible(truck2, req1, 'add', 2) and  
 is\_feasible(truck2, req2, 'remove', 2)):  
 *# Tạo bản sao của lộ trình để thử nghiệm* new\_route1 = replace\_request\_in\_route(truck1['route'], req\_id1, req2)  
 new\_route2 = replace\_request\_in\_route(truck2['route'], req\_id2, req1)  
 *# Tính toán thời gian mới* temp\_truck1 = truck1.copy()  
 temp\_truck1['route'] = new\_route1  
 time1 = calculate\_truck\_time(temp\_truck1, distances)  
 temp\_truck2 = truck2.copy()  
 temp\_truck2['route'] = new\_route2  
 time2 = calculate\_truck\_time(temp\_truck2, distances)  
 *# Kiểm tra xem makespan có giảm không* old\_makespan = max(truck1['time'], truck2['time'])  
 new\_makespan = max(time1, time2)  
 if new\_makespan < old\_makespan:  
 *# Chấp nhận hoán đổi* update\_truck\_route(truck1, new\_route1)  
 update\_truck\_route(truck2, new\_route2)  
 improved = True  
 break *# Có cải thiện, tiếp tục vòng lặp chính* if improved:  
 break  
 if improved:  
 break  
 if improved:  
 break  
  
def replace\_request\_in\_route(route, old\_req\_id, new\_req):  
 new\_route = []  
 for step in route:  
 if step['request\_id'] == old\_req\_id:  
 *# Thay thế bằng yêu cầu mới* if step['action'] == requests[old\_req\_id]['pickup\_action']:  
 action = new\_req['pickup\_action']  
 duration = new\_req['pickup\_duration']  
 elif step['action'] == requests[old\_req\_id]['drop\_action']:  
 action = new\_req['drop\_action']  
 duration = new\_req['drop\_duration']  
 new\_step = {  
 'point': step['point'], *# Giữ nguyên điểm* 'action': action,  
 'request\_id': new\_req['id']  
 }  
 new\_route.append(new\_step)  
 else:  
 new\_route.append(step)  
 return new\_route  
  
def create\_steps\_for\_request(req\_id, req):  
 steps = []  
 steps.append({  
 'point': req['pickup\_point'],  
 'action': req['pickup\_action'],  
 'request\_id': req\_id  
 })  
 steps.append({  
 'point': req['drop\_point'],  
 'action': req['drop\_action'],  
 'request\_id': req\_id  
 })  
 return steps  
  
def is\_feasible\_route(truck, capacities):  
 load = 0  
 for step in truck['route']:  
 action = step['action']  
 req\_id = step['request\_id']  
 if action in ['PICKUP\_CONTAINER', 'PICKUP\_CONTAINER\_TRAILER']:  
 req = requests[req\_id]  
 required\_capacity = 1 if req['size'] == 20 else 2  
 load += required\_capacity  
 if load > capacities:  
 return False  
 elif action in ['DROP\_CONTAINER', 'DROP\_CONTAINER\_TRAILER']:  
 req = requests[req\_id]  
 required\_capacity = 1 if req['size'] == 20 else 2  
 load -= required\_capacity  
 return True  
  
def insert\_unassigned\_requests(trucks, requests, distances):  
 unassigned\_reqs = [req\_id for req\_id, req in requests.items() if not req['assigned']]  
 for req\_id in unassigned\_reqs:  
 req = requests[req\_id]  
 best\_truck\_id = None  
 best\_insertion = None  
 min\_increase = float('inf')  
 for truck\_id in trucks:  
 truck = trucks[truck\_id]  
 *# Thử chèn yêu cầu vào các vị trí khác nhau trong lộ trình* for i in range(len(truck['route']) + 1):  
 for j in range(i, len(truck['route']) + 1):  
 new\_route = truck['route'][:i] + \  
 create\_steps\_for\_request(req\_id, req) + \  
 truck['route'][i:]  
 temp\_truck = truck.copy()  
 temp\_truck['route'] = new\_route  
 *# Kiểm tra tính khả thi* if is\_feasible\_route(temp\_truck, 2):  
 time\_increase = calculate\_truck\_time(temp\_truck, distances) - truck['time']  
 if time\_increase < min\_increase:  
 min\_increase = time\_increase  
 best\_truck\_id = truck\_id  
 best\_insertion = new\_route  
 if best\_truck\_id is not None:  
 *# Chèn yêu cầu vào lộ trình của đầu kéo tốt nhất* truck = trucks[best\_truck\_id]  
 update\_truck\_route(truck, best\_insertion)  
 requests[req\_id]['assigned'] = True  
  
def improve\_solution(trucks, requests, distances):  
 previous\_makespan = None  
 while True:  
 swap\_requests(trucks, requests, distances)  
 insert\_unassigned\_requests(trucks, requests, distances)  
 *# Tính toán makespan mới* F1, F2 = calculate\_objective(trucks, distances)  
 if previous\_makespan is not None and F1 >= previous\_makespan:  
 break *# Không cải thiện được nữa* previous\_makespan = F1  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 *# Đọc dữ liệu đầu vào* N, distances, trailer\_location, trailer\_attach\_time, trucks, requests = read\_input()  
  
 *# Khởi tạo giải pháp ban đầu* assign\_requests(trucks, requests, distances, trailer\_location, trailer\_attach\_time)  
  
 *# Cải thiện giải pháp* improve\_solution(trucks, requests, distances)  
  
 *# Tính toán hàm mục tiêu* F1, F2 = calculate\_objective(trucks, distances)  
 F = 10000 \* F1 + F2  
 print(f'OBJECTIVE {F}')  
  
 *# Xuất kết quả* output\_result(trucks)

- Input: Mẫu trên Hustack

- Output:

OBJECTIVE 66010800

ROUTES 3

TRUCK 1

3 PICKUP\_TRAILER

1 PICKUP\_CONTAINER 1

5 DROP\_CONTAINER 1

3 DROP\_TRAILER

4 STOP

#

TRUCK 2

5 PICKUP\_CONTAINER\_TRAILER 2

7 DROP\_CONTAINER 2

6 PICKUP\_CONTAINER 5

2 DROP\_CONTAINER 5

3 DROP\_TRAILER

4 STOP

#

TRUCK 3

3 PICKUP\_TRAILER

8 PICKUP\_CONTAINER 3

6 DROP\_CONTAINER\_TRAILER 3

7 PICKUP\_CONTAINER\_TRAILER 4

9 DROP\_CONTAINER 4

3 DROP\_TRAILER

4 STOP

#

Process finished with exit code 0

**5.**

… (Phiên bản hiện tại)

Bảng so sánh output mẫu so với output bản thân (F score càng nhỏ càng tốt):

|  |  |  |
| --- | --- | --- |
| STT Test | F score của output mẫu | F score của output bản thân |
| 1 | 5231539 | 4221192 |
| 2 |  |  |