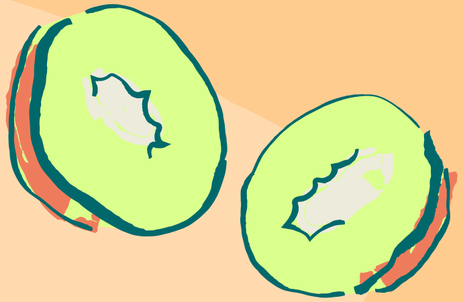
OPTIMAL DELIVERY ROUTE SELECTION

Case Study 1: Solving Real-World Problems using Computational Thinking

Presented by:
BIT & BYTE

The Issue at Hand

Couriers often struggle in determining the optimal delivery route when delivering multiple items across various locations while operating within strict time constraints. The complexity of managing different destinations, varying delivery priorities, adds to the difficulty of ensuring timely and accurate deliveries.



1st Iteration

How much time is needed to complete deliveries to multiple locations?

Decomposition

- List all locations that require deliveries.
- List the distance to each location.
- List the speed of the delivery vehicle.

Pattern Recognition

- Delivery time is dependent on distance and speed.

Abstraction

- Only distance and speed are relevant.
- The number of packages or delivery priority is irrelevant.

2nd Iteration

Question: How can we determine a delivery route to take?

Decomposition

- List all locations that require deliveries.
- List the distance to each location.
- List the priority of each delivery.

Pattern Recognition:

- Some locations have higher priority and should be delivered first.

Abstraction:

- The most relevant factors are distance, speed, and priority.

3rd Iteration

How can we optimize the best delivery route based on speed, priority, and efficiency?

Decomposition

- Identify priority and distance to determine the optimal order.
- Calculate travel time based on speed and distance.
- Choose the best route to minimize total delivery time while prioritizing urgent deliveries.

Pattern Recognition:

- Some deliveries have higher priority and need to be scheduled earlier.
- A shorter distance does not always lead faster delivery route.

Abstraction:

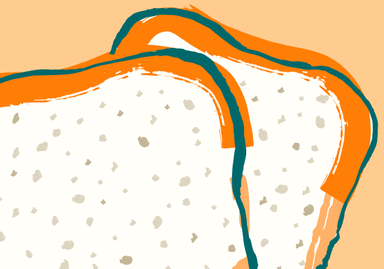
- The key factors are priority, distance, and speed.

Proposed Solution

Our proposed solution a delivery route optimization program that takes location distance, priority levels, and vehicle speed. It prioritizes deliveries with high-priority and distance location based on the time and reduces travel time through an automatic process. The system determines the most efficient delivery sequence, by comparing delivering high-priority or prioritizing locations first. The result yields an optimal route that achieves a balance of speed and efficiency along with the time it would take to deliver all the items.

ALGORITHMS USED

Greedy Algorithm was used
for the program



CODE:

```
def delivery(arr, speed):
    num_delivery = len(arr)

    def dev(done, need, arr, speed):
        needed = sum(s[2] for s in arr)

        #function that sorts the items by its priority level
        # priority level is the amount of hours needed to deliver the item
        def by_priority(arr, speed, needed):
            arr.sort(key=lambda x: x[2])
            best_route = []
            best_time = 0
            dist = 0

            #loops through each item to be delivered
            for i in range(need):
                a = arr[i][1]
                b = arr[i][2]
                total_dist = dist + a
                total_time = total_dist / speed
                best_time += total_time
                #values to be return if the time to deliver all
                #items exceed the time to be delivered

                if best_time > needed or best_time > b:
                    return float('inf'), [], "Error: No possible route" # returns error
                best_route.append(arr[i][0])
                dist = a
            #stating that this method is better
            method1 = "Delivering by prioritizing priority level is better"

            return round(best_time,2), "-->".join(best_route), method1
```

CODE:

```python
def by_distance(arr, speed, needed):
    arr.sort(key=lambda x: x[1])
    best_route = []
    best_time = 0
    dist = 0

    for i in range(need):
        a = arr[i][1]
        b = arr[i][2]
        total_dist = dist + a
        total_time = total_dist / speed
        best_time += total_time
        #values to be return if the time to deliver all
        #items exceed the time to be delivered
        if best_time > needed or best_time > b:
            return float('inf'), [], "Error: No possible route" # returns error
        best_route.append(arr[i][0])
        dist = a
    #stating that this method is better
    method2 = "Delivering by prioritizing location distance is better"

    return round(best_time,2), "-->".join(best_route), method2
```

CODE:

```
    #gives the results to variables named by its representation and
    #its respective method 1 = by_priority 2 = by_distance
    time1, route1, method1 = by_priority(arr, speed, needed)
    time2, route2, method2 = by_distance(arr, speed, needed)

  #compares the time from the two methods and shows which is better
    if time1 < time2:
        return time1, route1, method1
    elif time2 < time1:
        return time2, route2, method2
    #this shows if both the time are equal
    #and checks if it yields same optimal time or both cannot compute
    elif time1 == time2 and route1 == []:
        return time1, "None", ("Both yield same results: " + method1)
    else:
        return time1, "\nor \n".join([(route1),(route2)]), "Both yield same results, both are optimal"
#gives values to results that will be presented
time, route, method = dev(0, num_delivery, arr, speed)
#time = round(time, 2)
return time, route, method
```

CODE:

```
# Testing of the code
# items are in a nested list showing the location name,
# location distance, and priority level respectively
A = [["Location A", 30, 3], ["Location B", 90, 2], ["Location C", 100, 1]]
speed = 50  # in km/hr
time, route, method = delivery(A, speed) #calls the main function to solve
# prints out the results
print(f"Results when delivering the items with average speed of {speed} km/hr:\n")
print(f"Time to deliver all: {time} hours\n")
print(f"Optimal route is:\n{route}\n")
print(method)
```

Q & A

Questions about the study