Algorithms Fall 2026
Greedy Algorithms Solution

October 27, 2025

Zubeyda Shute
Fall 2025 Comps/Algorithms/Chloe & Emma

---

1. Part A: The Company's Greedy Strategy
   (a) Goal: Maximize company profit on this trip.
   - Profit = deliver fee - travel cost
   (b) Task
   1. A greedy algorithm written in Python with proper documentation
   2. A proof for your algorithm in LaTeX.
   (c) Hint: These can be nested further - Look at each nearby destinations you have not visited
   - Pick and move to the destinations gives the largest profit right now
   - Repeat the process
   - Return to the depot (remember going back also costs money)

Ans: Task 1. Python Code

(a) insert python code block here (on GitHub)

Ans: Task 2. Proof of Correctness in LaTeX.

(a) Correctness: For the company's greedy algorithm, this approach works because at each step we're always picking the customer that gives us the most profit right now, which is the delivery fee minus the travel cost to get there. Since the total profit is just adding up all the individual profits from each delivery and subtracting all the travel costs, making the best choice at each step should lead to a pretty good overall profit. It's like if you're trying to make as much money as possible in a day, you'd probably want to do the jobs that pay the most relative to how far you have to travel first. The algorithm makes sure we only visit each customer once and we always end up back at the depot, so it gives us a valid route. While it might not always find the absolute best possible route, for real-world delivery problems this greedy method usually finds a solution that's good enough and it's much simpler than trying to check every single possible route combination.

2. Part B: The Driver's Greedy Strategy
Now, you will modify your greedy approach from the driver's point of view instead of the company's. Each driver earns tips from customers but must also pay for their own fuel and any tolls on the way. Naturally, the driver wants to make as much take-home money as possible, while still driving the shortest or cheapest routes when possible.

(a) Goal: Maximize driver's earnings on this trip..
- Earnings = delivery fee for drivers + estimated fees - travel cost

(b) Task
1. A greedy algorithm written in Python with proper documentation
2. A proof for your algorithm in LaTeX.

(c) Hint: Look at each nearby customer you have not visited. - Pick and move to the destination with the largest earning
- Repeat the process
- Return to the depot (remember going back also costs money)

Ans: Task 1. Python Code

(a) insert python code block here (on GitHub)

Ans: Task 2. Proof of Correctness in LaTeX.

(a) Correctness: For the driver's version, the greedy algorithm changes to focus on what actually matters to the driver, which is the delivery fee plus the tip minus the travel costs. This is different from the company's version because drivers care about tips while the company doesn't. So now when we choose the next customer, we're looking for people who might not have the highest delivery fee but are good tippers, since that's where drivers make most of their money. This makes sense because in real life, drivers would probably want to prioritize customers who are known to tip well, even if the company's delivery fee for that stop is a bit lower. The algorithm still considers travel costs so we don't waste money driving too far, but it better matches how drivers actually think about their earnings. It's still a greedy approach so it has the same advantages of being simple and efficient while giving reasonable results.

3. Part C: Ethical & Real-life Reflections Now that you've seen how a greedy algorithm can make decisions from different perspectives, you might start questioning if a greedy algorithm is always the best? We want to dive deeper in this section for ethical concerns. A delivery algorithm might make routes faster, but what if that means some neighborhoods are always served last? Or what if a driver gets tired because the route keeps choosing short but overwhelmed city streets?
   (a) Goal: Think about responsibility and reflect them in your code.
   (b) Task
   1. Modification to your code in either Part A or Part B (you do not need to do both) 2. Run and compare
   3. Short answers to 3 critical questions (each in 2-3 sentences):

   - Did your ethical rule achieve your goal? (Did it change the order of stops or the total outcome?)

   - Did it improve something for a person (safety, fairness, balance), even if it made profit smaller?

   - How's the change in total profits?

   (c) Hint: Step 1: Choose one real-life concern to focus on. (Eg. Safety, Fairness, Fatigue)
   - Fairness rule: Every second stop must be in a low-tip region if one is available.
   - Fatigue rule: After any long drive, the next stop must be short and nearby.
   Step 2: Implement a new rule into your greedy selection logic. For example:
   Step 3: Run and Compare your modified algorithm against the original. Answer the 3 critical questions.

Ans: Task 1. Modified Python Code to Part A or B

(a) insert python code block here (on GitHub)

Ans: Task 2. Run & Compare Results

(a) insert python code block here or explanation (on GitHub)

Ans: Task 3. Questions:

(a) Yes, the fairness rule alternates between high and low tip areas, changing the order of stops to ensure a more equitable distribution across different neighborhoods with incentives to workers as well.

(b) The fairness rule improves service quality for customers in low-tip areas who likely otherwise always are served last, promoting social fairness even though it slightly reduces overall company earnings.

(c) Total earnings would decrease compared to the purely greedy approach, but this trade-off is acceptable due to the improved fairness and social benefits of serving undeserved communities, incentivizing workers, and creating a long-term customer base with equity.