7.1 Boston to Seattle

You're planning to drive from Boston to Seattle on I-90 this summer, and have a GPS programmed with the locations of gas stations along the way. Assuming that your car can go 400 miles on a tank of gas, determine where you should stop for gas in order to make as few stops as possible.

> Sections (like this one) which marked with a vertical line on the left side are commentary — discussions about the algorithm development process that wouldn't be part of a writeup.

Establish the problem.

- *Specifications.*

  Task: find the gas stations to stop at along the way from Boston to Seattle

  Input: locations of n gas stations

  Output: which gas stations to stop at

  Legal solution: gas stations are not more than 400 miles apart

  Optimization goal: fewest number of stops

- *Examples.*

Identify avenues of attack.

- *Known targets.*
- *Approach.* Series of choices.

  > (This was dictated.)

- *Paradigms and patterns.*

  Paradigm: greedy.

  > (Again, this was dictated.)

  > Identify the flavor, if applicable. Consider how this problem fits into each applicable pattern.

  This is a find a subset problem — we stop at some subset of the full set of gas stations along the route.

  Process input: for each gas station, do we stop there?

  Produce output: repeatedly find the next gas station to stop at

- *The series of choices.*

  Produce output: repeatedly find the next stop, moving from Boston to Seattle

  Process input: for each gas station, do we stop there?

  > Both process input and produce output seem simple enough to express. While this is a subset problem and it in principle doesn't matter what order the stops are reported in, there is a naturally-defined order (from Boston to Seattle). This provides a meaningful idea of "next step" and "next step" lends itself well to having a sense of progress, so it seems like produce output may be a good way to go.

  The series of choices: the next stop, moving from Boston to Seattle

- *Greedy choices and counterexamples.*

> To identify the possible greedy choices, identify what information there is to work with. In this case, we have the location of each gas station and can compute how far it is from another position (such as our current position or the previous stop).

With the goal of the fewest stops, the only plausible choice is to drive as far as possible before stopping again — pick the next stop as the farthest-away gas station (in the direction of Seattle) within 400 miles.

Define the algorithm.

- *Main steps.*

  > The general form for a greedy produce output approach is:
  >
  > ```
  > while not done
  >   choose the next output item
  > ```

  ```
  while not within 400 miles of Seattle
    the next stop is the farthest ahead within 400 miles
  ```

- *Exit condition.*

  Stop when we are within 400 miles of Seattle.

- *Setup.*

  Start in Boston. Assume a full tank of gas. (Necessary so that "farthest ahead within 400 miles" produces a legal choice for the first stop.)

- *Wrapup.*

  Drive the rest of the way to Seattle. (No additional stops.)

- *Special cases.*

  What if there's a gap of more than 400 miles between successive stations in the input? Then there is no legal solution.

  What if there are two gas stations in the same location (the same distance from Boston)? Not an issue — either one is fine to pick as the stop. One additional check is needed — if the farthest ahead gas station within 400 miles is 0 miles away (one of the other stations in the same location), then there's about to be a greater-than-400-mile gap and so there's no solution.

- *Algorithm.*

  Task: Assuming that your car can go 400 miles on a tank of gas, determine where you should stop for gas in order to make as few stops as possible.

  Input: locations of n gas stations

  Output: which gas stations to stop at

  ```
  while not within 400 miles of Seattle
    choose the farthest (towards Seattle) gas station within 400 miles of the
     previous stop
    if there isn't one or if it is at the current position,
     break and report no legal solution
  ```

Show termination and correctness.

- *Termination.*

  - *Measure of progress.*

> The pattern is to base this on the exit condition, which involves the distance remaining to Seattle.

The distance we are from Seattle.

- *Making progress.*

  Each iteration picks the next stop. The only way this doesn't get closer to Seattle is if it is 0 miles away, but in that case "no solution" is reported and the algorithm terminates.

- *The end is reached.*

  Repeatedly reducing the distance to Seattle means it will eventually be ≤ 400 miles.

- *Correctness.*

  - *Loop invariant.*

    > The general pattern for the produce output approach is
    >
    > `After` k `output elements have been produced, ...`
    >
    > The loop invariant needs to address both legality and optimality. For optimality, we try a staying ahead argument rather than a direct assertion of having the best solution so far. However, the optimization criterion in this case is the number of output elements (stops) — staying ahead compares the progress of the algorithm's solution and the optimal solution after the same number of output elements have been produced so we can't say "after the first k output elements, the algorithm hasn't fallen behind in terms of the number of output elements". But what about the measure of progress? That's based on the distance remaining to Seattle, and being at least as close to Seattle as the optimal algorithm after k stops seems like a good thing...

    After the first k stops (in order from Boston to Seattle),

    * no two consecutive stops are more than 400 miles apart, and
    * the algorithm's kth stop is no farther from Seattle than the optimal's kth stop.

  - *Establish the loop invariant.*

    > Show for $k = 0$ to establish that the setup is fine, and for $k = 1$ (after the first iteration) to establish that the greedy choice starts off OK.

    $k = 0$: Before any stops have been made, both the algorithm and the optimal are at the start (in Boston). There are no stops to be more than 400 miles apart, and the algorithm and the optimal are the same distance from Seattle.

    $k = 1$: After the first stop,

    * no stops are more than 400 miles apart because the algorithm only picks stops within 400 miles of the previous stop (in this case, Boston), and
    * the algorithm's first stop is no farther from Seattle than the optimal's first stop because the algorithm picks the farthest stop from Boston within 400 miles — any farther and it wouldn't be a legal solution — so the optimal cannot have gotten any farther from Boston and thus also cannot have gotten any closer to Seattle.

  - *Maintain the loop invariant.*

    Assume that after the first k stops (in order from Boston to Seattle), no stops are more than 400 miles apart and the algorithm's kth stop is no farther from Seattle than the optimal's kth stop.

    Show that after the first $k+1$ stops (in order from Boston to Seattle), no stops are more than 400 miles apart and the algorithm's $k+1$th stop is no farther from Seattle than the optimal's $k+1$th stop.

    Legality. There aren't any stops more than 400 miles apart in the first k stops and the algorithm only picks stops within 400 miles of the previous stop so stop $k+1$ can't be the one with too big of a gap. If it is not possible to pick a next stop within 400 miles (or if the farthest one is 0 miles away), there is no solution.

Optimality. By contradiction: assume that stop $k+1$ is where things go wrong, that is, the optimal's $k+1$th stop is closer to Seattle than the algorithm's $k+1$th stop.

Let A_k and O_k be the location (distance from Boston) of the kth stop in the algorithm's solution and the optimal solution, respectively. The loop invariant means that $A_k \geq O_k$ — the algorithm's kth stop is no farther from Seattle than the optimal's kth stop. Assuming this is where things go wrong means that $A_{k+1} < O_{k+1}$ — the optimal's $(k+1)$th stop is closer to Seattle than the algorithm's. Furthermore, $A_{k+1} > A_k$ because the loop makes progress — each iteration finds a stop closer to Seattle. Putting this all together yields $O_k \leq A_k < A_{k+1} < O_{k+1}$ — but if $O_{k+1} - O_k \leq 400$, then $O_{k+1} - A_k$ must also be ≤ 400. That means O_{k+1} was a legal choice for the algorithm and since $O_{k+1} > A_{k+1}$, the algorithm would have picked it. Thus we have a contradiction, so the assumption that the algorithm falls behind is incorrect and the invariant must hold.

– *Final answer.*

> Show that the main steps + wrapup yields the final answer. Since there aren't any wrapup steps, show that the loop invariant plus the exit condition results in the final answer.

Legality. The loop invariant gives us that after k stops, no stops are more than 400 miles apart. The loop condition gives us that when the loop exits, we are within 400 miles of Seattle so no additional stops are needed. Thus the final solution is valid.

Optimality. The loop invariant gives us that after k stops, the algorithm is no farther from Seattle than the optimal. That the loop has exited means that the algorithm's solution has k stops. But what we need is that k is the fewest number of stops. Let $|A|$ and $|O|$ denote the number of stops in (size of) the algorithm's and optimal solutions, respectively. There are three possibilities:

* $|A| < |O|$
* $|A| = |O|$
* $|A| > |O|$

> $|A| = |O|$ is what we want — the algorithm found a solution with the same number of stops as the optimal solution. Show this by showing that the other two cases are impossible.

$|A| < |O|$ is impossible because the optimal solution by definition has the fewest stops. The algorithm can't have found a legal solution with fewer stops.

If $|A| > |O|$, consider where the algorithm was after $|O|$ stops. The loop invariant gives us that after $|O|$ stops the algorithm must be no farther from Seattle than the optimal. Because stop $|O|$ was the optimal's last, it must at that point be within 400 miles of Seattle. But if the optimal was within 400 miles of Seattle and the algorithm is no farther than the optimal from Seattle, the algorithm must also be within 400 miles of Seattle after $|O|$ stops. So $|A| > |O|$ is not possible.

As a result, $|A| = |O|$ and the algorithm's solution has the fewest number of stops.

Determine efficiency.

- *Implementation.*

  If the gas stations aren't already in order by distance from Boston, sort them. Finding the next stop then just involves scanning forward from the current stop until the last stop ≤ 400 miles away is found.

- *Time and space.*

  Sorting the list of gas stations is $O(n \log n)$. The main loop is $O(n)$ because we only scan forward through the list to find successive gas stations, so the entire list is traversed once.

  The running time of the algorithm is thus $O(n \log n)$ if we have to sort the gas stations and $O(n)$ otherwise.

- *Room for improvement.*

> Seems unlikely to beat $O(n)$ once the list is sorted.