7.1 Boston to Seattle

> 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.*

  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.

  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.

- *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.

  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.*

  > 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). Interpreting "next" in this context provides something more concrete when showing correctness.

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

  Process input: whether to stop at the current gas station, considering the stops in order from Boston to Seattle

  > Both process input and produce output seem simple enough to express. In that case, favor produce output as a starting point because it can be easier for framing the greedy choice.

  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 — the next stop is the farthest-away gas station (in the direction of Seattle) within 400 miles.

Define the algorithm.

- *Main steps.*

> The general form for a produce output approach is:
>
> ```
> repeat
>   make the next choice
> until done
> ```

```
repeat
  choose the farthest (towards Seattle) gas station within 400 miles of the
   previous stop
until we are within 400 miles of Seattle
```

- *Exit condition.*

  Stop when we are within 400 miles of Seattle.

- *Setup.*

  Start in Boston. Assume a full tank of gas.

- *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 — as long as it's not the other gas station is the only one within 400 miles.

- *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

repeat
  choose the farthest (towards Seattle) gas station within 400 miles of the
   previous stop
  if there isn't one or if it is in the same location as the previous stop,
   break and report no legal solution
until we are within 400 miles of Seattle
```

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 a stop closer to Seattle than the previous one, reducing the distance we are from Seattle.

- *The end is reached.*

  If the distance to Seattle keeps getting reduced, it will eventually be ≤ 400 miles.

- *Correctness.*

  - *Loop invariant.*

    > The general pattern is
    >
    > `After` k `iterations, ...`
    >
    > The loop invariant needs to address both legality and optimality. For optimality, try a staying ahead argument. For a subset problem only the set of stop in the solution matters, so consider the optimal solution to be ordered in the same way the algorithm picks stops (from Boston to Seattle) in order to ensure an apples-to-apples comparison with the staying ahead argument.

    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 at least as far from Boston as the optimal's kth stop.

  - *Establish the loop invariant.*

    > Show for $k = 1$ because $k = 0$ is too trivial — it doesn't even involve the greedy choice.

    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 at least as far from Boston as 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.

  - *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 at least as far from Boston as 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 at least as far from Boston as the optimal's $k+1$th stop.

    Legality. The algorithm only picks stops within 400 miles of the previous stop — if this is not possible, 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 farther from Boston than the algorithm's $k+1$th stop.

  - *Final answer.*

    > Show that the setup + main steps + wrapup yields the final answer. Since there aren't any wrapup steps, show that the setup means the loop invariant is true at the beginning ($k = 0$) and that the loop invariant plus the exit condition results in the final answer.

    The setup is that we start in Boston with a full tank of gas. No stops are more than 400 miles apart because we have only the starting position, and the algorithm can't be behind the optimal at this point because both start in the same place (Boston).

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 — the solution is valid.

Optimality. The loop invariant gives us that after k stops, the algorithm is at least as far from Bostom as 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 optimal must be within 400 miles of Seattle since stop $|O|$ was its last. The loop invariant gives us that after $|O|$ stops the algorithm must be at least as far from Boston as the optimal. But if the optimal was within 400 miles of Seattle, that means the algorithm must be too. 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.