

CIS 775

Finding Minimum Stops

Rengakrishnan Subramanian
November 20, 2001

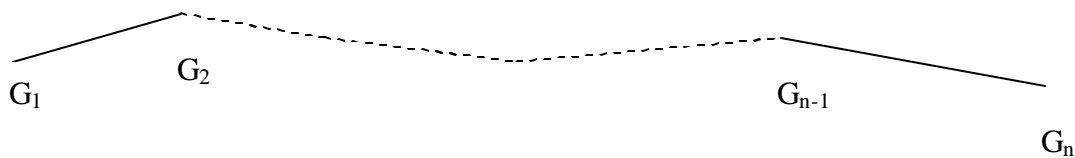
Problem Statement

Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car's gas tank, when full, holds enough gas to travel ' d ' miles, and his map gives the distances between gas stations on his route. The professor wishes to make as few stops as possible along the way.

The problem is to find an efficient method by which Professor Midas can determine at which gas stations he should stop, and prove that the strategy adopted yields an optimal solution. [2]

Problem Description

Input



1. The map of the Interstate from G_1 to G_n , where G_i is a Gas Station, $1 \leq i \leq n$, (including the starting point G_1 - Newark and destination G_n - Reno)
2. The Distance vector $\text{Dist} [1 \dots, n-1]$, where, n = total number of available gas stations and $\text{Dist} [i]$ is the distance from G_i to G_{i+1} . The above diagram is a sample diagram. In the above diagram, the vector will be the set, $\text{Dist} [G_1 - G_2, \dots, G_{n-1} - G_n]$.

Assumptions

1. Each of the stops in the map, including the starting point and ending point are considered as gas stations, which have enough gas to fill up Prof. Midas's gas tank. (Though it is not necessary to consider the end point as Gas station, it is considered for uniformity).

2. Prof. Midas's car can travel ' d ' miles, once it has been filled with full tank of ' g ' gallons.
3. All distances in the Distance Vector are such that the car can surely travel from one gas station to the other. I.e. the distance between gas stations is assumed not to be greater than ' d '. ($\text{Dist}[i] \leq d$).
4. The Prof. fills up his gas tank completely at any gas station he stops.
5. At Newark, the Prof. starts with a full gas tank.

Output

1. A set of stops of the type, $\text{Stops}[G_i, \dots, G_j, \dots]$, where, G_i and G_j are gas stations such that $1 \leq i, j \leq n-1$ and are gas stations where the Prof. stops to fill gas. This set is chosen from the input vector.

Algorithm

The approach followed here is '*greedy*'.

$i \leftarrow 1$

distance_to_travel \leftarrow Dist [i]

while ($G_{i+1} \neq G_n$) {

distance_to_travel \leftarrow Dist [i]

/* Greedy loop */

while ((distance_to_travel \leq d) AND (G_{i+2} exists)) {

destination = G_{i+1}

distance_to_travel \leftarrow distance_to_travel + Dist [i+1]

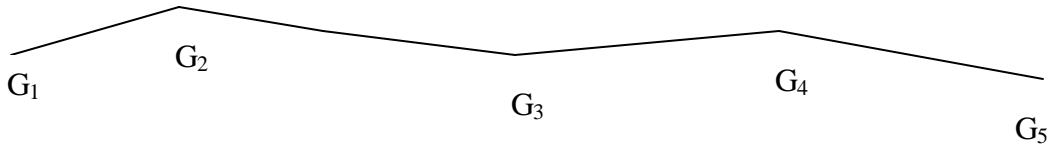
$i \leftarrow i + 1$

}

Put 'destination' in Solution set S {}

}

Example



The optimality of the above greedy solution can be traced back to the optimality of the decision that the algorithm makes at the decision point on the algorithm. Let the above be a sample map given to Prof. Midas.

Let the vector Distance $\{ G_1 - G_2, G_2 - G_3, G_3 - G_4, G_4 - G_5 \} = \{ 10, 20, 25, 20 \}$ be the input

Let the distance 'd' that the car can travel on a full tank be = 30 miles.

Note: $G_i - G_{i+1} \leq d$

The above greedy algorithm proceeds in the following manner:

- The first distance to be checked is $G_1 - G_2$. This distance, as assumed is less than or equal to d .
- The algorithm gets greedy at this stage.
- It checks for the next distance from the input set Distance Vector. (In this case, it is $G_2 - G_3$)`
- The algorithm adds this new distance to the distance it wants to travel.
- If the new distance is still less than or equal to d (in this case, $G_1 - G_2 + G_2 - G_3 = d$), the algorithm computes this distance to be the new distance that it can travel.
- The algorithm now gets more greedy to travel the next distance ($G_3 - G_4$ in this case) and repeats steps 3 to 5 and travels the distance it can travel.
- It proceeds similarly till it finds the destination.

The final solution in this case is $\{ G_1 - G_2 + G_2 - G_3, G_3 - G_4, G_4 - G_5 \}$

Proof of Correctness

Theorem: The greedy algorithm produces an optimal solution

Proof:

Let A be the output set produced by the algorithm. The question is whether there exists any other solution set B, such that B is optimal. If such a solution B exists, it must be possible that "the total number of elements in B is less than or equal to the total number of elements in A".

Consider B. If it includes A [1] (the first element of A), then the greedy algorithm worked at the first step. If B did not have A [1], then let us remove B [1] (the first element of B) and replace it with A [1]. The set B is still of the same size. Let us see if B is still

feasible. A noticeable point here is, according to our greedy algorithm, A [1] is at least as far from Newark as the first element of the optimal solution B, which is B [1]. So, the replacement of B [1] with A [1] is still feasible.

The problem is now reduced to finding optimal solution from A [1] to Reno. If B is the optimal set, then, B – A [1] is also optimal for the sub problem. This is because, if there were a smaller solution set C for traveling from A [1] to Reno, then C+A [1] will be smaller than B. But, C is impossible, because B is the optimal set. Hence, the solution set A exactly matches with the optimal solution B.

Thus, the solution A found by the greedy algorithm is optimal.

Time Complexity Analysis

The algorithm has two while loops in the following way:

```
while (condition) {
    .
    distance_to_travel ← Dist [i]
    while (condition) {
        .
        i ← i+1
    }
    .
}
```

If there are n gas stations in the map, then there are $n-1$ distances in the input vector. The statement to be noticed is $i ← i+1$. During the course of the algorithm, this statement can execute, in the worst case, at most $n-1$ times. The algorithm terminates when the value of i becomes $n-1$. Thus, the inner loop iterates at most $n-1$ times over the course of the algorithm.

The outer loop iterates at most n times.

In conclusion, the algorithm can compute minimum stops with time complexity of **O (n)**.

References

- [1] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics*. Prentice Hall of India, 1998.
- [2] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. *Introduction to Algorithms*. Prentice Hall of India, 1999.