RutaEficiente

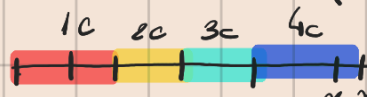
16. Tomás quiere viajar de Buenos Aires a Mar del Plata en su flamante Renault 12. Como está preocupado por la autonomía de su vehículo, se tomó el tiempo de anotar las distintas estaciones de servicio que se encuentran en el camino. Modeló el mismo como un segmento de 0 a M, donde Buenos aires está en el kilómetro 0, Mar del Plata en el M, y las distintas estaciones de servicio están ubicadas en los kilómetros $0 = x_1 \leq x_2 \leq \dots x_n \leq M$.

    Razonablemente, Tomás quiere minimizar la cantidad de paradas para cargar nafta. Él sabe que su auto es capaz de hacer hasta C kilómetros con el tanque lleno, y que al comenzar el viaje este está vacío.

    a) Proponer un algoritmo *greedy* que indique cuál es la cantidad mínima de paradas para cargar nafta que debe hacer Tomás, y que aparte devuelva el conjunto de estaciones en las que hay que detenerse. Probar su correctitud.

    b) Dar una implementación de complejidad temporal $O(n)$ del algoritmo del inciso *a*).

Distancia de Buenos Aires a MDQ



Si esta es la distancia a recorrer y este es C entonces son solo necesarias 4 paradas

C: ⟵

$(x_{n+1} = M)$

Estrategia Greedy: sup que estamos en la pos x_i $1 \leq i \leq n$, sumo C a x_i y la la próxima estación de servicio en la que debere parar sera la que esté en esa posición si hay una o la que esté en la pos x_j con $j = \min\{k / x_i + C - k = x_i' \text{ tq } x_i' \text{ es una estación de servicio}\}$

Algoritmo

1. Dada una pos x_i le sumo C y sumo 1 a la variable donde cuento la # de paradas hechas
2. Recorro en orden las posiciones recordando la última por la que pasé
3. Si $\nexists x_j / x_j = x_i + C$ entonces guardo en mi lista de paradas la última pos en la que estuve. Sino guardo x_j

Demostremos la correctitud del algoritmo.

Sea v^* la solución alcanzada veamos que es correcta

Sup que v^* no es correcta $\rightarrow \exists\ s^* /$ #paradasHechas s^* < #paradasHechas v^*

$\rightarrow \exists\ x_j /\ x_j \in$ paradasHechas v^*

$x_j \notin$ paradasHechas s^*

$\rightarrow$ ABS!

NO probemos x INDUCCION

con pistas para # paradas hasta, el otro es el resultado ✓ de agregar esos valores

Sea $p(i)$: $\forall i / 0 \le i \le n$ mi sol golosa $v^* \le s^*$ una sol candidata

QVQ es correcta

Caso base $i=0$ $v^* = 1 = s^* \rightarrow$ siempre voy a cargar en la 1ra parada.

Caso inductivo $p(i) \rightarrow p(i+1)\ \forall i \in \mathbb{N}$

Si $i \geq n+1$ trivial

Si $i \le n$ sean v_i^* y s_i^* los valores de v^* y s^* a medida que voy recorriendo mis posiciones

Tenemos 2 casos, al llegar a la pos $i+1$ tengo nafta para la prox parada o no tengo y debería haber cargado

① $v_{i+1}^* = v_i^* \overset{HI}{\le} s_i^* \le s_{i+1}^*$ (va a ser igual o menor)

② $v_{i+1}^* = v_i^* + 1 \overset{HI}{\le} s_i^* + 1 \rightarrow$ como en mi sol golosa siempre elijo la parada + lejos posible a la que pueda llegar con esa nafta $\rightarrow$ estoy minimizando # de paradas

$\rightarrow$ ⓐ $s_i^* = v_i^* \rightarrow s_{i+1}^* = s_i^* + 1$ (tuve que haber parado)

$\rightarrow v_i^* + 1 \le s_i^* + 1 \le s_{i+1}^*$

ⓑ $s_i^* > v_i^* \rightarrow v_i^* + 1 \le s_i^* \le s_{i+1}^*$ (como son nat a lo sumo difieren en 1 unidad)

(puedo cargar o no depende)

— Retomando ② $v_i^0 \leq v_{i+1}^* \leq s_{i+1}^* \leq s_{i+1}^0$

```
RE ( vector<int> paradas, int c){
    int cantParadas = 1;
    vector<int> paradasHechas[0] = paradas[0];
    int ultPos = paradas[0]
    int distQuePuedoRecorrer = c
    for (int i = 1; i < paradas.size; i++){
        if paradas[i] = distQuePuedoRecorrer then
            cantParadas++;
            paradasHechas.push_back( paradas[i] );
            distQuePuedoRecorrer = paradas[i] + c;
        else if paradas[i] > distQuePuedoRecorrer then
            cantParadas++;
            paradasHechas.push_back( ultPos );
            distQuePuedoRecorrer = ultPos + c;
        endif
            ultPos = paradas[i];
    }
    return cantParadas, paradasHechas
```

Complejidad espacial $O(n)$ con $n = |paradas|$
temporal $\theta(n)$