Algoritmos golosos - Ejercicios

Ejercicio 1

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 $0 = x_1 \leq x_2 \leq \ldots x_n \leq M$.

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.

- 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.
- Dar una implementación de complejidad temporal $O(n)$ del algoritmo.

Solución

La idea es agarrar siempre la máxima estación alcanzable en C kilómetros, luego repetir lo mismo desde la estación en la que estamos hasta llegar al final. El algoritmo* entonces queda

---

$idx \leftarrow 0$
$estaciones \leftarrow \{E[0]\}$
for $i \in 1 \ldots n$ **do**
    if $E[i] - E[idx] > C$ **then**
        $idx \leftarrow i - 1$
        $estaciones \leftarrow estaciones \cup \{E[i-1]\}$
    end if
end for

---

*Modificado con comentarios durante la clase.

Nos queda probar que es óptimo. Tenemos que ver dos cosas

- Que la solución es válida.
- Que es óptima.

Validez: Nuestra solución sería invalida si en algún momento tuviéramos una estación donde se detiene pero no es alcanzable. Por la elección de nuestro algoritmo esto no es posible.[‡]

Optimalidad: Queremos ver que es óptimo, para esto queremos ver que la cantidad de estaciones de una solución óptima y la golosa son iguales. Supongamos que esto no pasa es decir Sea

$$e_1, \ldots, e_n$$

nuestra solución golosa dónde cada e_i representa los kilómetros hasta dónde se llego y

$$o_1, \ldots, o_m$$

una solución óptima. Y supongamos que $m < n$ es decir la óptima tiene menos paradas.

Demostremos primero lo siguiente

Lema. *Sea $\mathcal{O} = \{o_1, \ldots, o_m\}$ una solución óptima con $|\mathcal{O}| = m$ y $\mathcal{G} = \{g_1, \ldots, g_n\}$ nuestra solución golosa con $|\mathcal{G}| = n$, además $n > m$, entonces para $i \in (1, \ldots, m)$*

$$o_i \leq g_i$$

Demostración: Inducción en las estaciones óptimas.

Caso base $i = 1$, como g_1 era lo más que se podía alcanzar desde el comienzo $o_1 \geq g_1$ sino nuestro algoritmo goloso hubiera elegido o_1 también.

Caso inductivo $i = k + 1$, supongamos que demostramos que para $1 \leq i \leq k$ vale que

$$o_i \leq g_i$$

queremos verlo para $i = k + 1$.

Dado g_{k+1} lo podemos expresar como la estación anterior más una distancia menor o igual que C.

$$o_{k+1} = o_k + D$$

con $0 < D \leq C$, o_k esta en el caso de la HI entonces

$$o_{k+1} = o_k + D \leq g_k + D$$

Veamos que entonces g_{k+1} en realidad está adelante de o_{k+1},

$$o_{k+1} - g_k \leq g_k + D - g_k = D^{\S}$$

[‡]En realidad deberíamos probarlo por inducción, una vez que lo probamos para la k ésima agregar uno nuevo es igual que el caso base y se mantiene válido, pero me parece que no aporta nada. Si no se entendió esto pregunten.

[§]Modificado levemente con comentarios durante la clase

Es decir desde g_k hay una distancia $D \leq C$ a o_{k+1} y como nuestro algoritmo goloso siempre agarra lo máximo posible esto nos dice que al menos alcanza a o_{k+1} (y si hay más lo pasa) por lo tanto

$$o_{k+1} \leq g_{k+1}$$

■

Ahora que probamos el lema veamos que el goloso es en realidad óptimo, habíamos supuesto al principio que $m < n$ es decir que había más estaciones en nuestra solución golosa que en la óptima, esto nos dice que

$$g_m < M - C$$

y aplicando el Lema

$$o_m \leq g_m < M - C$$

Es decir que la solución optima todavía debería hacer al menos una parada más.

Contradicción, vino de suponer que $m < n$ por lo tanto $n = m$. ■

Ejercicio 2

Dados dos vectores $v, w \in \mathbb{R}^n$ queremos reordenar las coordenadas de tal manera de minimizar el producto escalar. Recordemos que este se definía como

$$\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$$

Propuesta de algoritmo: ordenar un vector de menor a mayor y el otro de mayor a menor, luego probar que el producto escalar es se minimiza. El costo total es $O(n \log(n))$. Veamos que es óptimo.

Validez: Acá ni hace falta hacer inducción hicimos un reorden con los elementos originales asi que no se puede invalidar.

Optimalidad: Vamos a hacer un argumento de intercambio modificando la versión optima de a poco en la versión golosa sin perder el valor óptimo.

Nuestra solución golosa $\mathcal{G}$ es de la forma

$$v_1 \leq v_2 \cdots \leq v_n$$

$$w_1 \geq w_2 \cdots \geq w_n$$

Tomemos una solución optima cualquiera:

$$\hat{v}_1, \hat{v}_2 \ldots \hat{v}_n$$

$$\hat{w}_1, \hat{w}_2 \ldots \hat{w}_n$$

Primera observación, podemos reordenar uno de los vectores de la solución golosa sin alterar el valor óptimo (¿Por qué?), por ejemplo $\hat{w}$ con lo que obtenemos una nueva solución óptima, digamos $\mathcal{O}_1$ de la forma

$$\hat{v}_1, \hat{v}_2 \ldots \hat{v}_n$$

$$w_1 \geq w_2 \cdots \geq w_n$$

Si $\hat{v} = v$ no hay nada que hacer asi que asumamos que son distintos y veamos que podemos transformar $\hat{v}$ de manera que sea igual a v sin perder la optimalidad.

Sea el primer índice j donde difieren ($\hat{v}_j \neq v_j$) entonces en $\hat{v}$ hay otro índice $k > j$ tal que

$$\hat{v}_j = v_k \text{ y } \hat{v}_k = v_j \tag{1}$$

y además $\hat{v}_k < \hat{v}_j$ †.

Si restamos los valores invertidos de $\hat{v}$ en el producto interno ($\hat{v}_k w_k + \hat{v}_j w_j$) de la versión optima nueva con los que están en orden ($\hat{v}_j w_k + \hat{v}_k w_j$) tenemos

$$(\hat{v}_k w_k + \hat{v}_j w_j) - (\hat{v}_j w_k + \hat{v}_k w_j)$$

Reordenando

$$\begin{gathered}(\hat{v}_k w_k - \hat{v}_k w_j) + (\hat{v}_j w_j - \hat{v}_j w_k) = \\ \hat{v}_k(w_k - w_j) + \hat{v}_j(w_j - w_k) = \\ -\hat{v}_k(w_j - w_k) + \hat{v}_j(w_j - w_k) = \\ (\hat{v}_j - \hat{v}_k)(w_j - w_k) \geq 0\end{gathered}$$

Esto nos dice que si intercambiamos $\hat{v}_k$ por $\hat{v}_j$ la solución no se incrementa y pone a $\hat{v}_j = v_j$. Obtenemos un $\mathcal{O}_2$ que invirtió los valores k, j y sigue siendo igual de óptimo y ahora hasta el índice v_j se parece a $\mathcal{G}$.

Si repetimos esto para todos los indices, en orden creciente que difieren, como siempre intercambiamos (si es necesario) hacia adelante†† obtenemos una secuencia $\{\mathcal{O}_k\}$ finita que luego de $n+1$ pasos es igual a $\mathcal{G}$ y nunca perdió su optimalidad, por lo tanto $\mathcal{G}$ es óptima.

†Es decir no estos dos no están en orden creciente como la golosa.

††La parte que ya reordenamos se queda igual.