Algoritmos Golosos

Leandro Javier Raffo

Departamento de Computación, FCEyN, UBA

2024

Idea para hoy

- Dar un breve repaso de las técnicas algorítmicas que vimos hasta ahora.
- Presentar que es un algoritmo goloso
- Resolver dos ejercicios de golosos con correctitud incluida.

Cierre técnicas algorítmicas

Repasemos lo que vimos hasta ahora:

- Backtracking
  - ! Búsqueda exhaustiva + Podas
- Programación dinámica
  - ! Búsqueda exhaustiva + solapamiento de subproblemas
- Dividir y conquistar
  - ! Partir subproblemas hasta que son resolubles de una manera más simple
- Algoritmos golosos
  - ! ???

Algoritmos golosos

- Generalmente se usan para problemas de optimización
- La idea es construir una solución al problema haciendo decisiones mediante una heurística que toma la decisión óptima local
- Podemos tener muchas estrategias golosas para un mismo conjunto de datos dependiendo de lo que se quiera optimizar
- No todas nos dan un resultado óptimo global

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

Ejercicio 1

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.

¿Ideas?

Algoritmo

Vamos a elegir algo que parece lo más natural, elegir la máxima estación que alcanzamos con C combustible

---

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

¿Cuál es la complejidad de esto? $O(n)$.

Correctitud

- ¿Por qué correcta esa decisión?
- ¿Podría ser que alguna solución óptima no elija la mayor estación que se puede alcanzar y de repente "pase" a la solución propuesta por nuestro algoritmo visitando menos estaciones?
- Tenemos que demostrar que es correcto, esto es ver que:
  - La solución que obtuvimos es válida.
  - (En general) es óptima.
- Casi siempre la validez es mucho más fácil de demostrar que la optimalidad.

Solución válida - Ejercicio 1

- ¿Cuándo seria inválida nuestra solución?
- 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 en general

En general hay dos formas de ver que un algoritmo goloso es óptimo

- Demostrar que nuestro algoritmo siempre se mantiene "adelante" de cualquier otra solución óptima (Greedy stays ahead).
- Mostrar que podemos agarrar una solución óptima y modificarla sin incrementar su costo y llevarla a la de nuestro algoritmo goloso (Exchange argument).

Optimalidad - idea

Sea

$$e_1, \ldots, e_n$$

nuestra solución golosa y

$$o_1, \ldots, o_m$$

una solución óptima. Para ver que la golosa es óptima tenemos que ver que tienen la misma longitud $n = m$.

- Vamos a dividir la demostración en dos
- Suponer que la golosa no es óptima es decir $m < n$.
- Usar la primer estrategia, mostrar que la solución golosa siempre está "adelante" de la óptima.
- Concluir que llegamos a una contradicción y tenia que ser $n = m$.

Optimalidad - Parte 1

En otras palabras lo que queremos probar es
Lema: Sea $o_i \in \mathcal{O}$ una solución óptima con $|\mathcal{O}| = m$, $g_i \in \mathcal{G}$ nuestra solución golosa con $|\mathcal{G}| = n$ y además $n > m$, entonces para $i \in (1, \ldots, m)$

$$g_i \geq o_i$$

Optimalidad - Parte 1 - Caso base

Vamos a demostrar esto usando inducción.

Caso base: i $= 1$. Tenemos g_1 y o_1, como ambas parten del mismo lugar y nuestra estrategia golosa se movió lo máximo que se podía tenemos que

$$o_1 \leq g_1$$

Optimalidad - Parte 1 - caso inductivo

Caso inductivo: nuestra hipótesis inductiva es la siguiente, dado $1 \leq i \leq k$ tenemos que $o_i \leq g_i$, observemos que pasa con $i = k + 1$.
Por un lado tenemos

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

con $0 < D < C$
Queremos ver que $g_{k+1} \geq o_{k+1}$, cómo

$$o_{k+1} - g_k \leq_{\text{HI}} o_{k+1} - o_k$$

Si unimos las dos cosas anteriores conseguimos

$$o_{k+1} - g_k \leq C$$

Por lo cual g_{k+1} se puede mover al menos hasta o_{k+1} que es lo que queríamos probar.

Optimalidad - Cerrando

- ¿Terminamos la demostración?
- No, solo probamos que la golosa se mantiene adelante de la óptima.
- Ahora como $m < n$, tenemos que la óptima termina en o_m, pero $o_m \leq g_m < M - C$, lo cual contradice que $\mathcal{O}$ sea una solución válida.
- Como esto vino de suponer que $m < n$ llegamos a una contradicción y por lo tanto tiene que ser $n = m$.

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 era

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

Intervalo

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 era

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

En principio uno tal vez querría intentar a hacer una programación dinámica, pero estamos en la clase de algoritmos golosos.
Propuesta: ordenar un vector de menor a mayor y el otro de mayor a menor.
¿Funciona?
Costo del algoritmo $O(nlogn)$

De vuelta correctitud

- ¿Nuestra solución es valida?
- Si, solo reordenamos los valores de los vectores.
- ¿Y la optimalidad?
- Acá entra el segundo método que nombramos antes para demostrar cosas sobre algoritmos golosos que es el argumento de intercambio.
- La idea es transformar una solución óptima en la solución golosa sin haber perdido el valor óptimo.

Correctitud

Nuestra solución golosa es de la forma

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

y

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

Agarremos una solución optima cualquiera

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

y

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

Primera observación, podemos reordenar uno de los vectores de la solución golosa sin alterar el valor óptimo, por ejemplo $\hat{w}$, de lo que obtenemos

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

y

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

Optimalidad

- ¿En qué difieren ahora la solución golosa y esta nueva versión de la óptima, digamos $\mathcal{O}_\infty$?
- Si $\hat{v}$ y v están en el mismo orden ganamos, asi que supongamos que no, significa que tenemos un valor invertido en la secuencia creciente.
- Es decir $\hat{v}_j > \hat{v}_k$ con $j < k$.
- Veamos que si corregimos esto de vuelta no empeora la solución

Optimalidad

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

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

Lo que nos dice esto es que si intercambiamos v_k por v_j en esta solución óptima no vamos a empeorar el valor que habíamos encontrado.
Es decir obtuvimos $\mathcal{O}_2$ que no empeora la optimalidad y está más cerca de la golosa. Si repetimos asi hasta n vamos a haber convertido $\mathcal{O}$ en $\mathcal{G}$ sin haber empeorado la solución. Por lo tanto $\mathcal{G}$ es óptima.

Cierre técnicas algorítmicas

Repasemos lo que vimos hasta ahora:

- Backtracking
  - ! Búsqueda exhaustiva + Podas
- Programación dinámica
  - ! Búsqueda exhaustiva + solapamiento de subproblemas
- Dividir y conquistar
  - ! Partir subproblemas hasta que son resolubles de una manera más simple
- Algoritmos golosos
  - ! ???

Cierre técnicas algorítmicas

Repasemos lo que vimos hasta ahora:

- Backtracking
  - ! Búsqueda exhaustiva + Podas
- Programación dinámica
  - ! Búsqueda exhaustiva + solapamiento de subproblemas
- Dividir y conquistar
  - ! Partir subproblemas hasta que son resolubles de una manera más simple
- Algoritmos golosos
  - ! Heurística con la mejor opción local

Conclusiones

Entonces hoy vimos lo siguiente:

- Dimos un repaso corto de en que consistía cada una de las técnicas que vimos hasta ahora
- Definimos que era un algoritmo goloso.
- Hicimos dos ejercicios cada uno con una técnica distinta para probar que los algoritmos propuestos eran correctos

¿Preguntas?