Ejercicio de la cinta (3.5 de la Práctica)

1. Enunciado

Sean P_1,P_2, $\cdots$, P_n programas que se quieren almacenar en una cinta. El programa P_i requiere s_i Kb de memoria. La cinta tiene capacidad para almacenar todos los programas. Se conoce la frecuencia π_i con que se usa el programa P_i. La densidad de la cinta y la velocidad del drive son constantes. Después que un programa se carga desde la cinta la misma se rebobina hasta el principio. Si los programas se almacenan en orden i_1, i_2, $\cdots$, i_n el tiempo promedio de carga de un programa es

$$T = c \sum_j \left(\pi_{i_j} \sum_{k \leq j} s_{i_k} \right)$$

donde la constante c depende de la densidad de grabado y la velocidad del dispositivo. Queremos construir un algoritmo goloso para determinar el orden en que se almacenan los programas que minimice T.

Analizar las siguientes estrategias de almacenamiento:

1. en orden no decreciente de los s_i
2. en orden no creciente de los π_i
3. en orden no creciente de $\frac{\pi_i}{s_i}$

¿Cómo se podría demostrar que la última estrategia es la mejor? (Sugerencia: analizar en cada caso qué ocurre cuando dos elementos contiguos desordenados se ordenan).

2. Solución

Ignoraremos la constante $c > 0$ del enunciado en todo el análisis (es decir supondremos $c = 1$), ya que esta constante multiplicativa no afecta a la hora de verificar cuál de dos ordenamientos obtiene un costo menor ($cx \leq cy \Leftrightarrow x \leq y$).

2.1. Parte a

Esta estrategia no es buena. Tiene sentido intuitivo poner los programas más cortos al comienzo de la cinta, ya que de esta manera la "penalidad" o costo en tiempo de recorrer la cinta que deberán esperar los programas que vienen al final será menor. Sin embargo, esto no es suficiente para obtener buenos resultados.

Ejemplo: Dos programas, P_1, P_2. P_1 tiene $\pi = 1$, $s = 1000$. P_2 en cambio tiene $\pi = 1000$, $s = 1001$. Con la estrategia 1, ubicamos el programa P_1 primero y luego el P_2. El costo total resultante es $1 \cdot 1000 + 1000 \cdot (1000 + 1001) = 2002000$. Pero si se hubiera ordenado al revés, ubicando el P_2 primero, el costo total será $1000 \cdot 1001 + 1 \cdot (1001 + 1000) = 1003001$.

Es decir que la solución propuesta por la estrategia 1 tiene un costo que difiere del óptimo en casi un 100 %, en nuestro ejemplo con solamente dos programas. Se observa claramente mirando el ejemplo que el problema central de esta estrategia es no considerar para nada la frecuencia de uso de un programa: En el

ejemplo, si ambos programas tuvieran la misma frecuencia, la elección de poner el más corto primero sería óptima, pero como las frecuencias son extremadamente dispares, mientras que las longitudes son casi iguales, conviene ubicar primero el programa más frecuente, aunque sea un poco más largo, para evitar en la mayoría de los casos tener que leer dos programas de la cinta en lugar de solo el deseado.

2.2. Parte b

La estrategia 2 tiene un problema completamente análogo a la 1, pero en este caso solamente se considera la frecuencia, sin considerar la longitud de los programas. Se puede verificar que esta estrategia tiene un error igual que la anterior con los programas P_1, con $\pi = 1000$, $s = 1$ y P_2 con $\pi = 1001$, $s = 1000$.

2.3. Parte c

Queda claro de los ejemplos estudiados en las dos partes anteriores, que una estrategia óptima deberá necesariamente considerar ambos datos asociados a un programa: su frecuencia, y su longitud. Esto es lo que hace la estrategia 3. Y el enunciado nos pide demostrar que esta estrategia de ordenar de acuerdo al cociente de ambas magnitudes produce un costo óptimo.

Seguiremos la sugerencia del enunciado: vamos a estudiar qué ocurre con el costo cuando intercambiamos dos programas adyacentes en la cinta. Supongamos entonces que tenemos ordenados en la cinta los programas $P_1, P_2, \cdots, P_i, P_{i+1}, \cdots, P_n$, en ese orden, con $1 \leq i < n$. Vamos a estudiar el costo total de esta configuración antes y después de intercambiar P_i con P_{i+1}.

Llamemos C_{orig} al costo original. Por definición tenemos que el costo es:

$$\begin{aligned} C_{\text{orig}} &= \sum_{j=1}^{n} \left(\pi_j \sum_{k \leq j} s_k \right) \\ &= \sum_{j=1}^{i-1} \left(\pi_j \sum_{k \leq j} s_k \right) + \pi_i \sum_{k \leq i} s_k + \pi_{i+1} \sum_{k \leq i+1} s_k + \sum_{j=i+2}^{n} \left(\pi_j \sum_{k \leq j} s_k \right) \\ &= \sum_{j=1}^{i-1} \left(\pi_j \sum_{k \leq j} s_k \right) + \pi_i \left(s_i + \sum_{k<i} s_k \right) + \pi_{i+1} \left(s_i + s_{i+1} + \sum_{k<i} s_k \right) + \sum_{j=i+2}^{n} \left(\pi_j \sum_{k \leq j} s_k \right) \end{aligned} \tag{1}$$

Donde hemos separado la sumatoria en partes, para mostrar por separado los términos correspondientes a los programas consecutivos que vamos a intercambiar, los términos de los programas anteriores, y los de los programas posteriores.

Luego de intercambiar P_i con P_{i+1}, es decir, al considerar el ordenamiento $P_1, P_2, \cdots, P_{i-1}, P_{i+1}, P_i, P_{i+2}, \cdots, P_n$, el nuevo costo será el mismo pero intercambiando los roles de P_i y P_{i+1} en la cuenta. Esto es más fácil de escribir modificando la expresión en 1, que tiene los términos separados en partes: El primer y último términos, correspondientes a los costos de los programas anteriores a i y posteriores al $i+1$ no se ven modificados (en el primero no aparecen términos de i o $i+1$, y en el segundo aparecen ambos sumando, con lo cual al intercambiarlos solo se cambia el orden en que se suman). Los términos intermedios tendrán un nuevo valor correspondiente al intercambio de i con $i+1$, quedando el costo:

$$C_{\text{swap}} = \sum_{j=1}^{i-1} \left(\pi_j \sum_{k \leq j} s_k \right) + \pi_{i+1} \left(s_{i+1} + \sum_{k<i} s_k \right) + \pi_i \left(s_{i+1} + s_i + \sum_{k<i} s_k \right) + \sum_{j=i+2}^{n} \left(\pi_j \sum_{k \leq j} s_k \right) \tag{2}$$

Ahora podemos analizar la variación del costo al realizar el intercambio: restando 1 a 2:

$$C_{\text{swap}} = \sum_{j=1}^{i-1}\left(\pi_j \sum_{k\le j} s_k\right) + \pi_{i+1}\left(s_{i+1} + \sum_{k<i} s_k\right) + \pi_i\left(s_{i+1} + s_i + \sum_{k<i} s_k\right) + \sum_{j=i+2}^{n}\left(\pi_j \sum_{k\le j} s_k\right)$$

$-$

$$C_{\text{orig}} = \sum_{j=1}^{i-1}\left(\pi_j \sum_{k\le j} s_k\right) + \pi_i\left(s_i + \sum_{k<i} s_k\right) + \pi_{i+1}\left(s_i + s_{i+1} + \sum_{k<i} s_k\right) + \sum_{j=i+2}^{n}\left(\pi_j \sum_{k\le j} s_k\right)$$

$\Rightarrow$

$$\begin{aligned}
C_{\text{swap}} - C_{\text{orig}} &= \pi_{i+1}\left(s_{i+1} + \sum_{k<i} s_k\right) + \pi_i\left(s_{i+1} + s_i + \sum_{k<i} s_k\right) - \\
&\quad - \left(\pi_i\left(s_i + \sum_{k<i} s_k\right) + \pi_{i+1}\left(s_i + s_{i+1} + \sum_{k<i} s_k\right)\right) \\
\Delta_{\text{cost}} &= \pi_{i+1}s_{i+1} + \pi_{i+1}\sum_{k<i} s_k + \pi_i s_{i+1} + \pi_i s_i + \pi_i \sum_{k<i} s_k - \\
&\quad - \left(\pi_i s_i + \pi_i \sum_{k<i} s_k + \pi_{i+1}s_i + \pi_{i+1}s_{i+1} + \pi_{i+1}\sum_{k<i} s_k\right) \\
\Delta_{\text{cost}} &= \pi_i s_{i+1} - \pi_{i+1}s_i \qquad \text{(l1)}
\end{aligned}$$

(Hemos llamado Δ_{cost} a la variación del costo al hacer el intercambio). Llegamos a que l1 nos da la variación del costo al realizar el intercambio de dos programas consecutivos. Particularmente nos importa saber cuándo esta variación es un aumento, cuando un decremento, y cuando el costo se mantiene. Mirando l1 podemos enunciar el

Lema 1. *Si se intercambian los programas consecutivos P_i y P_{i+1}, el **signo** (positivo/negativo/cero) de la variación del costo viene dado por:*

$$signo\,(\Delta_{cost}) = signo\left(\frac{\pi_i}{s_i} - \frac{\pi_{i+1}}{s_{i+1}}\right)$$

Por lo tanto surgen inmediatamente tres casos:

- $\frac{\pi_i}{s_i} < \frac{\pi_{i+1}}{s_{i+1}} \Leftrightarrow C_{swap} < C_{orig}$
- $\frac{\pi_i}{s_i} = \frac{\pi_{i+1}}{s_{i+1}} \Leftrightarrow C_{swap} = C_{orig}$
- $\frac{\pi_i}{s_i} > \frac{\pi_{i+1}}{s_{i+1}} \Leftrightarrow C_{swap} > C_{orig}$

Demostración. Partiendo de l1, y teniendo en cuenta que dividir por números positivos como s_i y s_{i+1} no cambia el signo:

$$\begin{aligned}
\Delta_{\text{cost}} &= \pi_i s_{i+1} - \pi_{i+1} s_i \\
signo\left(\Delta_{\text{cost}}\right) &= signo\left(\pi_i s_{i+1} - \pi_{i+1} s_i\right) \\
signo\left(\Delta_{\text{cost}}\right) &= signo\left(\frac{\pi_i s_{i+1} - \pi_{i+1} s_i}{s_i s_{i+1}}\right) \\
signo\left(\Delta_{\text{cost}}\right) &= signo\left(\frac{\pi_i}{s_i} - \frac{\pi_{i+1}}{s_{i+1}}\right)
\end{aligned}$$

□

A partir del lema, podemos probar rápidamente los siguientes dos teoremas, que completan la demostración de que la estrategia planteada es óptima.

Teorema 1. *Si una permutación de programas tiene costo óptimo, entonces esta permutación está correctamente ordenada de acuerdo a la estrategia 3.*

Demostración. Sea $P_1, P_2, \cdots, P_n$ una permutación con costo óptimo. Supongamos que no está ordenada de acuerdo a la estrategia 3. Eso significa que existe i tal que P_i, P_{i+1} consecutivos no están correctamente ordenados (puesto que si todos los pares de elementos adyacentes estuvieran en orden correcto, la permutación completa estaría en orden, y estamos suponiendo que la permutación no está ordenada).

Como no están correctamente ordenados de acuerdo a la estrategia 3, debe valer necesariamente que $\frac{\pi_i}{s_i} < \frac{\pi_{i+1}}{s_{i+1}}$, pero entonces por el lema, si intercambiamos P_i y P_{i+1} se obtiene una permutación de menor costo, lo cual es absurdo porque estamos suponiendo que la permutación tiene costo óptimo. Como el absurdo proviene de suponer que no está ordenada de acuerdo a la estrategia 3, concluímos que toda permutación de costo óptimo está necesariamente ordenada como lo indica la estrategia 3. □

Teorema 2 (Correctitud de la estrategia 3)**.** *Si una permutación de programas está ordenada de acuerdo a la estrategia 3, tiene costo óptimo.*

Demostración. Sea Π_1 una permutación de programas que está ordenada de acuerdo a la estrategia 3, es decir, $\frac{\pi_i}{s_i} \geq \frac{\pi_{i+1}}{s_{i+1}}$ para $1 \leq i < n$. Queremos demostrar que Π_1 tiene costo óptimo. Tomamos alguna permutación óptima Π_2. Lo que queremos probar es que Π_1 y Π_2 tienen el mismo costo.

Como Π_2 es óptima, por el teorema 1 está ordenada, al igual que la Π_1. Por lo tanto, la única diferencia entre Π_1 y Π_2 es la forma de romper empates, es decir, la forma en que se ordenan entre sí los elementos P_i y P_j que verifican $\frac{\pi_i}{s_i} = \frac{\pi_j}{s_j}$. Por lo tanto, es posible pasar de Π_1 a Π_2 realizando únicamente intercambios de elementos consecutivos **que empatan**.

Pero por el lema, un intercambio de dos elementos que empatan no modifica el costo. Luego es posible pasar de Π_1 a Π_2 realizando intercambios que no modifican el costo de la permutación, y por lo tanto concluímos que Π_1 y Π_2 tienen el mismo costo, como queríamos. □