UNIVERSIDAD DE GRANADA

# Minimizando el tiempo medio de acceso

## Algorítmica

**Aarón Bueno Rodríguez**
**Bryan Moreno Picamán**
**Miguel Ángel Rodríguez Serrano**

2º Grado Ingeniería Informática

Sean $n$ programas $P_1, P_2, \dots P_n$ que hay que almacenar en una cinta.
El programa $P_i$ requiere $S_i$ kilobytes de espacio y la cinta es suficientemente larga para almacenar todos los programas.
Se sabe con qué frecuencia se utiliza cada programa: una fracción $\pi_i$ de las solicitudes al programa $P_i$ (y por tanto $\sum_{i-1}^{n} \pi_i = 1$). Los datos se almacenan en la cinta con densidad constante y la velocidad de la cinta también es constante. Una vez que se carga el programa, la cinta se rebobina hasta el principio.
Si los programas se almacenan por orden $i_1, i_2, \dots, i_n$ el tiempo requerido para cargar un programa es, por tanto:

$$\hat{T} = c \cdot \sum_{j=1}^{n} \left[ \pi_{ij} \cdot \sum_{k=1}^{j} s_{ik} \right]$$

donde la constante *c* depende de la densidad de grabación y de la velocidad de la cinta.
Se desea minimizar $\hat{T}$ empleando un algoritmo voraz. Demuestre la optimalidad del algoritmo o encuentre un contraejemplo que muestre que el algoritmo no es óptimo para los siguientes criterios de selección:

- Programas en orden no decreciente de $s_i$
- Programas en orden no creciente de $\pi_i$
- Programas en orden no creciente de $\frac{\pi_i}{s_i}$

## Elementos de Greedy

Greedy proporciona varios elementos que facilitan la implementación y su correcto uso. Para este problema, del cual tenemos tres estrategias diferentes, obtenemos todos los elementos comunes excepto la función de selección.
A continuación pasamos a describir los elementos comunes:

- Conjunto de Candidatos: Todos los programas que tenemos que almacenar en la cinta.
- Conjunto de Seleccionados: Programas a almacenar hasta el momento.
- Función de Factibilidad: (En este caso no es necesaria función de factibilidad, ya que la cinta tendría espacio suficiente para albergar todos los programas que se quieren almacenar)
- Función Solución: El orden de los programas.
- Función Objetivo: El orden de los programas de modo que se consiga el mínimo tiempo de espera medio.

En función de cada estrategia usada, se obtendrá una función de selección diferente. Pasamos a indicar cuáles podrían darse:

- Programas en orden no decreciente de $s_i$
- Programas en orden no creciente de $\pi_i$
- Programas en orden no creciente de $\frac{\pi_i}{s_i}$

En el primer caso, para los programas en orden no decreciente de $s_i$, se elegirían aquellos cuyo tamaño fuera menor.

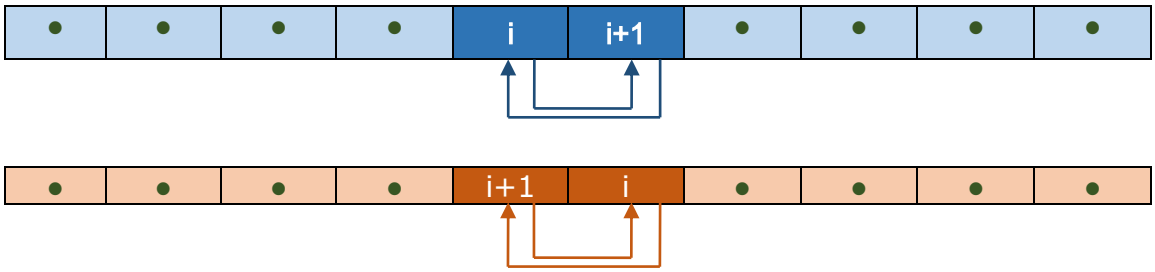
Para cuando se eligieran los programas basándonos en un orden no creciente de $\pi_i$, se escogerían aquellos programas que tuvieran un valor de solicitud mayor.

En cambio, si se decidiera usar como guía aquellos programas que estuvieran en un orden no creciente de $\frac{\pi_i}{s_i}$ , se tomarían primeros aquellos cuya proporción entre cuan solicitado sea ese programa y su tamaño fuera menor.

## Demostración de la optimalidad del Algoritmo

Se presenta el caso de la demostración, bien directamente o mediante un contraejemplo, de la no optimalidad de los criterios de selección facilitados para el algoritmo. Es por esto que, ayudándonos de un caso concreto, se pasa a demostrar lo que se pide.
Se comienza por el último criterio, el cual mediante el cociente entre la frecuencia con la que se utiliza cada programa y su tamaño, facilita una relación entre dichos parámetros que lleva a concluir de manera intuitiva será el óptimo.
Suponemos un tiempo promedio $\widehat{T}$ mayor o igual que el óptimo ($T^*$) con el fin de demostrar, usando *reducción al absurdo*, que esto no podría ocurrir, y, por consiguiente, que el óptimo lo es realmente.

A continuación se pasa a explicar mediante un ejemplo lo que posteriormente quedará demostrado de manera matemática. Consideramos como caso óptimo la organización de los programas en la primera representación (azul). En esta, se muestra una organización en orden no creciente de los programas, situándose al principio aquellos que tienen un tiempo de acceso medio menor y al final los que lo tienen mayor.
En la demostración posterior concluiremos que el caso en el que cambiamos el orden de dos de los programas en el orden no creciente establecido (naranja), nos devuelve como solución que este caso es el óptimo, quedando así demostrada la absurdez de la conclusión, y por tanto, que el que consideramos óptimo, lo es realmente.



$$\hat{T} \geq T^*$$

$$\hat{T} = \sum_{j=1}^{n} \left[ \pi_{lj} \cdot \sum_{k=1}^{j} S_{lk} \right] =$$

$$= \sum_{j=1}^{i-1} \left[ \pi_{lj} \cdot \sum_{k=1}^{j} S_{lk} \right] + \pi_{li} \cdot \sum_{k=1}^{i} S_{lk} + \pi_{l(i+1)} \cdot \sum_{k=1}^{i+1} S_{lk} + \sum_{j=i+2}^{n} \left[ \pi_{lj} \cdot \sum_{k=1}^{j} S_{lk} \right]$$

$$\geq$$

$$T^* = \sum_{j=1}^{i-1} \left[ \pi_{lj} \cdot \sum_{k=1}^{j} S_{lk} \right] + \pi_{l(i+1)} \cdot \left[ \sum_{k=1}^{i+1} S_{lk} - S_{li} \right] + \pi_{li} \cdot \left[ \sum_{k=1}^{i} S_{lk} + S_{l(i+1)} \right] + \sum_{j=i+2}^{n} \left[ \pi_{lj} \cdot \sum_{k=1}^{j} S_{lk} \right]$$

---

$$\pi_{li} \cdot \sum_{k=1}^{i} S_{lk} + \pi_{l(i+1)} \cdot \sum_{k=1}^{i+1} S_{lk} - \left( \pi_{l(i+1)} \cdot \left[ \sum_{k=1}^{i+1} S_{lk} - S_{li} \right] + \pi_{li} \cdot \left[ \sum_{k=1}^{i} S_{lk} + S_{l(i+1)} \right] \right) \geq 0$$

$$\pi_{li} \cdot \left[ \sum_{k=1}^{i} S_{lk} - \sum_{k=1}^{i} S_{lk} - S_{l(i+1)} \right] + \pi_{l(i+1)} \cdot \left[ \sum_{k=1}^{i+1} S_{lk} - \sum_{k=1}^{i+1} S_{lk} + S_{li} \right] \geq 0$$

$$\pi_{li} \cdot \left[ -S_{l(i+1)} \right] + \pi_{l(i+1)} \cdot \left[ S_{li} \right] \geq 0$$

$$-\pi_{li} \cdot S_{l(i+1)} + \pi_{l(i+1)} \cdot S_{li} \geq 0$$

$$\frac{\pi_{l(i+1)}}{S_{l(i+1)}} \geq \frac{\pi_{li}}{S_{li}}$$

## Demostración de los criterios $\boldsymbol{\pi_i}$ & $\boldsymbol{S_i}$

Pasamos a mostrar un ejemplo en el que queda demostrado que las otras dos estrategias no son optimales debido a que el tiempo de acceso que dan no es el mínimo, pues se encuentra un tiempo menor usando $\frac{\pi_i}{S_i}$ (que ya se sabe que es optimal)

En el siguiente ejemplo usaremos los siguientes datos:

| | | |
|---|---|---|
| $\boldsymbol{S_1 = 50K}$ | $\boldsymbol{\pi_1 = 0,25}$ | $\boldsymbol{\frac{\pi}{S_1} = 0,05}$ |
| $\boldsymbol{S_2 = 3K}$ | $\pi_2 = 0,1$ | $\frac{\pi}{S_2} = 0,3$ |
| $\boldsymbol{S_3 = 10K}$ | $\pi_3 = 0,65$ | $\frac{\pi}{S_3} = 0,065$ |

Usando $S_1$
Orden: 2,3,1

$$\hat{T} = \pi_2 \cdot S_2 + (\pi_3 \cdot (S_2 + S_3)) + (\pi_1 \cdot (S_2 + S_3 + S_1)) =$$
$$= 0,3 + (8,45) + 15,75 = \mathbf{24,5}$$

Usando $\pi_i$
Orden: 3,1,2

$$\hat{T} = (\pi_3 \cdot S_3) + (\pi_1 \cdot (S_3 + S_1)) + (\pi_2 \cdot (S_1 + S_2 + S_3)) =$$
$$6,5 + 15 + 6,3 = \mathbf{27,8}$$

Usando $\frac{\pi_i}{S_i}$
Orden: 3,2,1

$$\hat{T} = (\pi_3 \cdot S_3) + (\pi_2 \cdot (S_3 + S_2)) + (\pi_1 \cdot (S_1 + S_2 + S_3)) =$$
$$6,5 + 1,3 + 15,75 = \mathbf{23,55}$$

Con esto queda demostrado que $\frac{\pi_i}{S_i}$ es la única función de selección optimal.