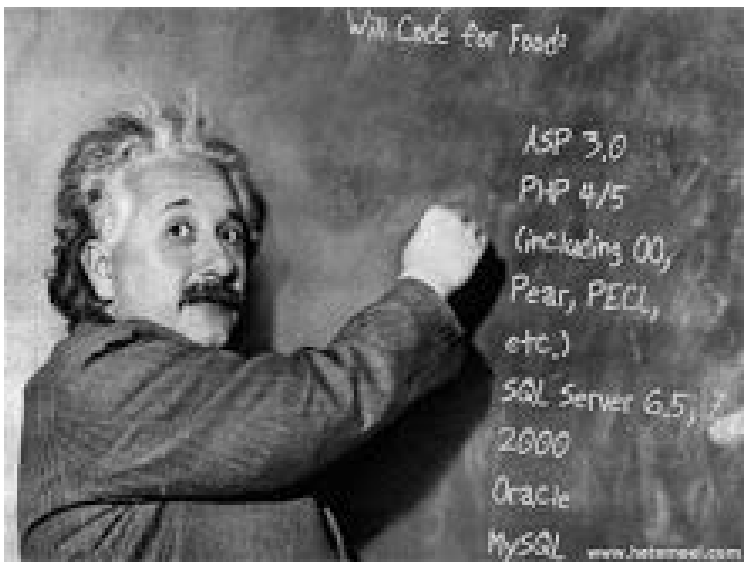
# Algoritmos Voraces

Practica 3, Algorítmica

Realizada por:
— Sergio Cervilla Ortega
— Daniel Díaz Pareja
— Marina Hurtado Rosales
— Adrián Morente Gabaldón
— Joaquín Rodríguez Claverías



# Índice

# 1. Introducción al problema

Supongamos una cinta en la que hay que almacenar *n* programas. Cada programa requiere en la cinta un espacio, pero esta tiene suficiente espacio para poder albergar los n programas.

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 i1, i2, … in el tiempo medio requerido para cargar un programa es, por tanto:

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

Donde la constante c depende de la densidad de grabación y de la velocidad de la cinta. Se desea minimizar 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 si.
— Programas en orden no creciente πi.
— Programas en orden no creciente de πi/si.

# 2. Descripción del algoritmo implementado.

- Elementos del algoritmo greedy:

- **Conjunto de candidatos (C):** Conjunto de programas.
- **Conjunto de seleccionados (S) :** Conjunto de programas ordenados en la cinta.
- **Función solución**: Que no haya elementos en C, es decir, que todos los elementos de C se hayan colocado en S.
- **Función de Factibilidad:** Siempre se encuentra una solución, ya que tenemos espacio suficiente para todos los programas y siempre hay programas candidatos.
- **Función de Selección:** En este caso encontramos tres funciones de selección distintas, cuya optimalidad estudiaremos a continuación.
    1. Programas más pequeños primero.
    2. Programas más usados primero.

3. Programas ordenados de mayor a menor según su frecuencia de uso/tamaño ($\pi i/si$).

- **Función objetivo:** Devuelve el valor que tarda la cinta en cargar todos los programas.

## 2.1. Pseudocódigo.

```
procedure Greedy(C)
{
        S = ø

        while C =/= ø do
                x = Selección(C)
                S = S U {x}
                C = C - {x}
        end

        return calcular_tiempo(S)
}
```

Donde:

- C: Conjunto de candidatos.
- S: Conjunto de seleccionados.
- x: Elemento a introducir en seleccionados
- Selección: Función que elige el mejor candidato a introducir en el conjunto selección.
- calcular_tiempo: Función que calcula el tiempo de carga de los programas según el conjunto de seleccionados.

# 3. Aplicación del algoritmo.

Para probar qué algoritmo es óptimo y cuál no, vamos a partir de una serie de ejemplos para ver qué función de selección es la más adecuada.

Supongamos el ejemplo de la tabla y supongamos n=5 y c=1, donde n es el número de programas y c la constante oculta que depende de la densidad de grabación y de la velocidad de la cinta.

| | P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|---|
| $S_i$ | 27 | 15 | 4 | 7 | 13 |
| $n_i$ | 0.4 | 0.3 | 0.1 | 0.15 | 0.05 |
| $\frac{\pi i}{Si}$ | 0.014 | 0.02 | 0.025 | 0.021 | 0.003 |

Mostraremos a continuación el valor de T para las diferentes funciones de selección, donde T es el tiempo que tarda la cinta en cargar todos los programas:

- **Programas más pequeños primero ($s_i$ más pequeño).**
  Se cargarían en el siguiente orden: P3, P4, P5, P2 y P1.

$$T= (\pi_3*S_3+\pi_4*(S_3+S_4)+\pi_5*(S_3+S_4+S_5)+\pi_2*(S_3+S_4+S_5+S_2)+\pi_1*(S_3+S_4+S_5+S_2+S_1) )$$

sustituyendo por los valores de la tabla:

$$T= (0.4)+(1.65)+(1.2)+(11.7)+(26.4)= \mathbf{41.35}$$

- **Programas con mayor frecuencia de uso primero ($\pi_i$ más grande).**
  Se cargarían en el siguiente orden: P1, P2, P4, P3 y P5.

$$T=( \pi_1*S_1+\pi_2*(S_1+S_2)+\pi_4*(S_1+S_2+S_4)+\pi_3*(S_1+S_2+S_4+S_3)+\pi_5*(S_1+S_2+S_4+S_3+S_5) )$$

sustituyendo por los valores de la tabla:

$$T=(10.8)+(12.6)+(6.9)+(5)+(3.15)= \mathbf{38.45}$$

- **Programas en orden decreciente según $\pi_i/s_i$.**
  Se cargarían en el siguiente orden: P3, P4, P2, P1 y P5.

$$T=(\pi_3*S_3+\pi_4*(S_3+S_4)+\pi_2*(S_3+S_4+S_2)+\pi_1*(S_3+S_4+S_2+S_1)+\pi_5*(S_3+S_4+S_2+S_1+S_5))$$

sustituyendo por los valores de la tabla:

T=(0.4)+(1.65)+(7.8)+(21.2)+(3.3)= **34.35**

Claramente se ve que el tercero es contraejemplo del primero y el segundo. Se han realizado varios procesos similares con otros datos y el resultado es siempre el mismo, el último caso es el mejor.

# 4. Demostración.

Como hemos visto anteriormente, se demuestra por contraejemplos que las funciones de selección "Programas más pequeños primero" y "Programas con mayor frecuencia de uso primero" no son óptimas.

Demostraremos a continuación que la función "Programas en orden decreciente según $\pi_i/s_i$" es óptima para un número n de programas.

Suponemos que $T_a$ es el tiempo de ejecución de los programas de la cinta según el orden que nos ha dado nuestro algoritmo voraz, y $T_b$ el tiempo resultante de la ejecución de los programas cambiando de orden dos cualesquiera de ellos.

Vamos a demostrar que el tiempo $T_b$ intercambiando dos programas consecutivos es siempre mayor que $T_a$. No es necesario demostrarlo en el caso de que intercambiemos dos programas cualesquiera, ya que se puede conseguir alternar dos programas cualesquiera haciendo solamente intercambios de programas en posiciones consecutivas.

Partimos de la premisa dada en el criterio de selección:

$$\frac{\Pi i}{Si} \geq \frac{\Pi i+1}{Si+1}$$

El tiempo $T_a$, siguiendo la fórmula de T, sería igual a:

$$Ta = c \cdot \{\Pi 1 \cdot S1 + \Pi 2 \cdot (S1 + S2) + ... + \Pi i \cdot (S1 + S2 + ... + Si) +$$
$$+ \ \Pi(i+1) \cdot [S1 + S2 + ... + Si + S(i+1)] ... + \Pi n \cdot [S1 + S2 + ... + Si + S(i+1) + ... + Sn]\}$$

Si intercambiamos los programas en las posiciones i e i+1, el tiempo $T_b$ sería:

$$Tb = c \cdot \{\Pi 1 \cdot S1 + \Pi 2 \cdot (S1 + S2) + ... + \Pi(i+1) \cdot [S1 + S2 + ... + S(i+1)] +$$
$$+ \Pi i \cdot [S1 + S2 + ... + Si + S(i+1)] + ... + \Pi n \cdot [S1 + S2 + ... + Si + S(i+1) + ... + Sn]\}$$

Como $T_b$ debe ser mayor que $T_a$, la diferencia $T_b$ - $T_a$ debe ser mayor que 0:

$$Tb - Ta > 0$$

$$c \cdot \{\Pi i \cdot (S1 + S2 + ... + Si) + \Pi(i+1) \cdot [S1 + S2 + ... + Si + S(i+1)] -$$
$$- \Pi(i+1) \cdot [S1 + S2 + ... + S(i+1)] - \Pi i \cdot [S1 + S2 + ... + Si + S(i+1)]\} > 0$$

Como la constante c es siempre mayor que 0, podemos eliminarla de la desigualdad. Sacamos ahora factor común:

$$\Pi i \cdot [S1 + S2 + ... + Si - S1 - S2 - ... - Si - S(i+1)] + \Pi(i+1) \cdot$$
$$\cdot [S1 + S2 + ... + Si + S(i+1) - S1 - S2 - ... - S(i+1)] > 0$$

Haciendo las restas nos quedaría la siguiente desigualdad:

$$\Pi(i+1) \cdot (-Si) + \Pi i \cdot S(i+1) > 0$$

$$\Pi i \cdot S(i+1) > \Pi(i+1) \cdot Si$$

$$\frac{\Pi i}{Si} > \frac{\Pi i+1}{Si+1}$$

Esta conclusión a la que hemos llegado cumple la premisa que teníamos inicialmente, por lo que podemos decir que la demostración es válida, y el criterio de selección "Programas en orden decreciente según $\pi_i/s_i$" da soluciones óptimas para cualquier número de programas n.