Universidad del Valle de Guatemala

Facultad de Ingeniería

Analisis y Diseño de Algoritmos

***PROYECTO 3***

Gabriel Paz 221087

02 de junio del 2025, Guatemala de la Asunción

*Repositorio (GitHub): https://github.com/gabrielpaz2003/PROYECTO3ALGORITMOS.git  
Video de YouTube (no listado): https://youtu.be/v71FHYyYXSA*

**1. Respuesta Caso 1**

**Configuración inicial:** 0 1 2 3 4  
**Secuencia solicitada (20 accesos):** 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4  
El programa imprimió, para cada acceso, la lista antes de atender la solicitud, el costo individual (posición + 1) y la lista reacomodada después de aplicar MTF. El costo total acumulado fue **90**.

**2. Respuesta Caso 2**

**Configuración inicial:** 0 1 2 3 4  
**Secuencia solicitada (17 accesos):** 4 3 2 1 0 1 2 3 4 3 2 1 0 1 2 3 4  
El programa registró el estado de la lista y el costo de cada acceso; la suma final fue **67**.

**3. Secuencia de 20 solicitudes con costo mínimo usando MTF**

La forma de minimizar el costo es solicitar siempre el primer elemento, de modo que cada acceso valga 1.  
**Secuencia óptima:** 0 0 0 … (20 veces)  
**Costo total:** 20

**4. Secuencia de 20 solicitudes con peor caso usando MTF**

Para forzar que cada acceso sea al último elemento, basta con repetir el patrón inverso de la lista:  
**Secuencia de peor caso:** 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0 4 3 2 1 0  
**Costo total:** 100 (5 unidades por cada una de las 20 solicitudes).

**5. Patrón cuando se repite el mismo elemento 20 veces**

* **Secuencia 2 × 20:** costo = 22  
  (Primer acceso cuesta 3 porque el 2 está en la tercera posición; los siguientes 19 cuestan 1).
* **Secuencia 3 × 20:** costo = 23  
  El costo general sigue la regla: costo = (posición inicial + 1) + 19 × 1. Es decir, tras el primer acceso el elemento se queda al frente y los siguientes accesos valen siempre 1.

**6. Costos totales con el algoritmo IMTF**

Se ejecutó el mismo programa con la política “look-ahead” propuesta por Mohanty y Tripathy: después de acceder a la posición *i*, el elemento sólo se mueve al frente si vuelve a aparecer dentro de las siguientes *i − 1* solicitudes.

| **Escenario** | **Secuencia usada** | **Costo MTF** | **Costo IMTF** | **Ahorro** |
| --- | --- | --- | --- | --- |
| Mejor caso | 0 × 20 | 20 | **20** | 0 % |
| Peor caso | 4 3 2 1 0 repetido | 100 | **60** | 40 % |

En la secuencia favorable no cambia nada porque el elemento siempre reaparece en la “ventana” de look-ahead y continúa al frente. En la secuencia adversa, IMTF evita mover elementos que no se reutilizarán inmediatamente, reduciendo el costo de 5 a 3 promedio por acceso.

**Conclusiones**

* El algoritmo MTF es muy sensible al orden de las solicitudes: puede variar desde el costo mínimo teórico (20) hasta el máximo posible (100) con la misma configuración inicial de cinco elementos.
* La versión IMTF atenúa significativamente los peores casos al usar información de futuro cercano; en nuestra prueba redujo el costo total un 40 %.
* Repetir el mismo elemento demuestra que, una vez que el ítem llega al frente, el costo marginal se estabiliza en 1, independientemente de su posición original.
* Para listas pequeñas, MTF es simple y efectivo, pero en aplicaciones donde la secuencia es adversa o impredecible, IMTF ofrece una alternativa con mejor rendimiento competitivo sin sacrificar demasiada simplicidad de implementación.