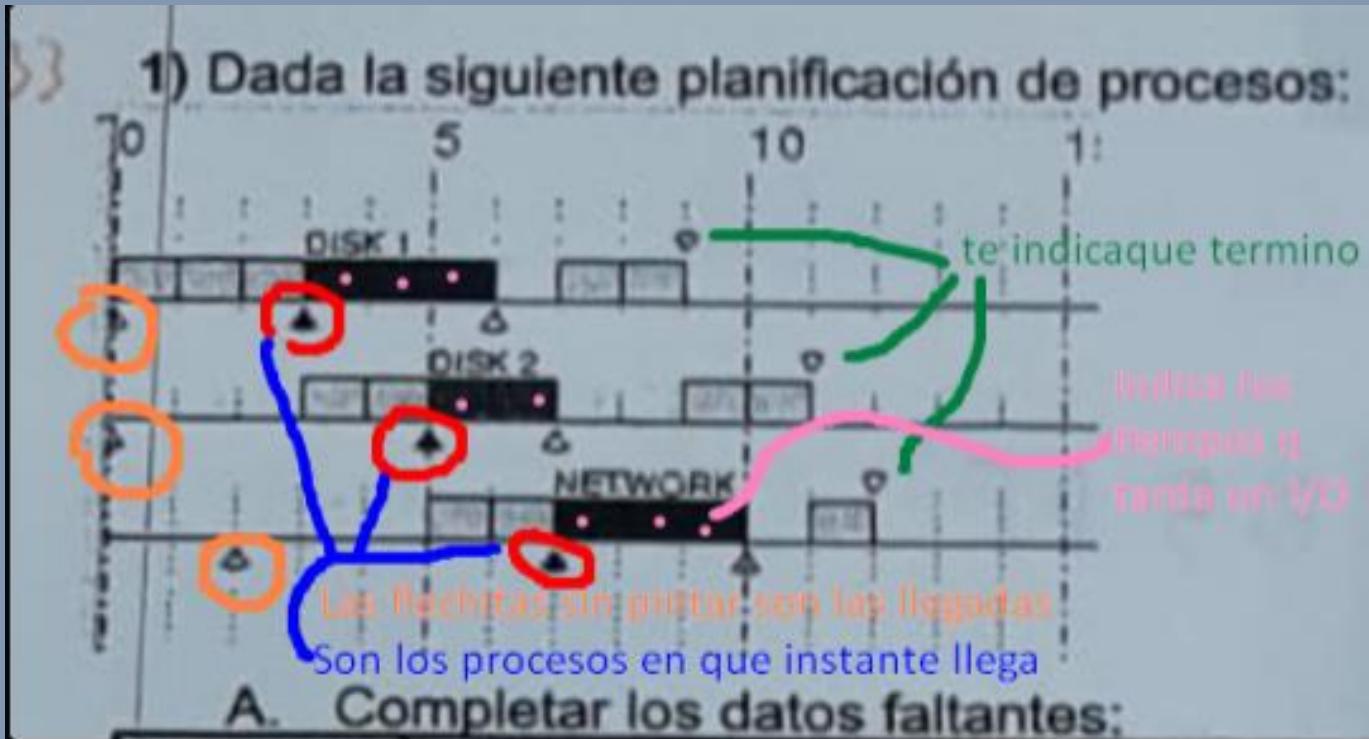
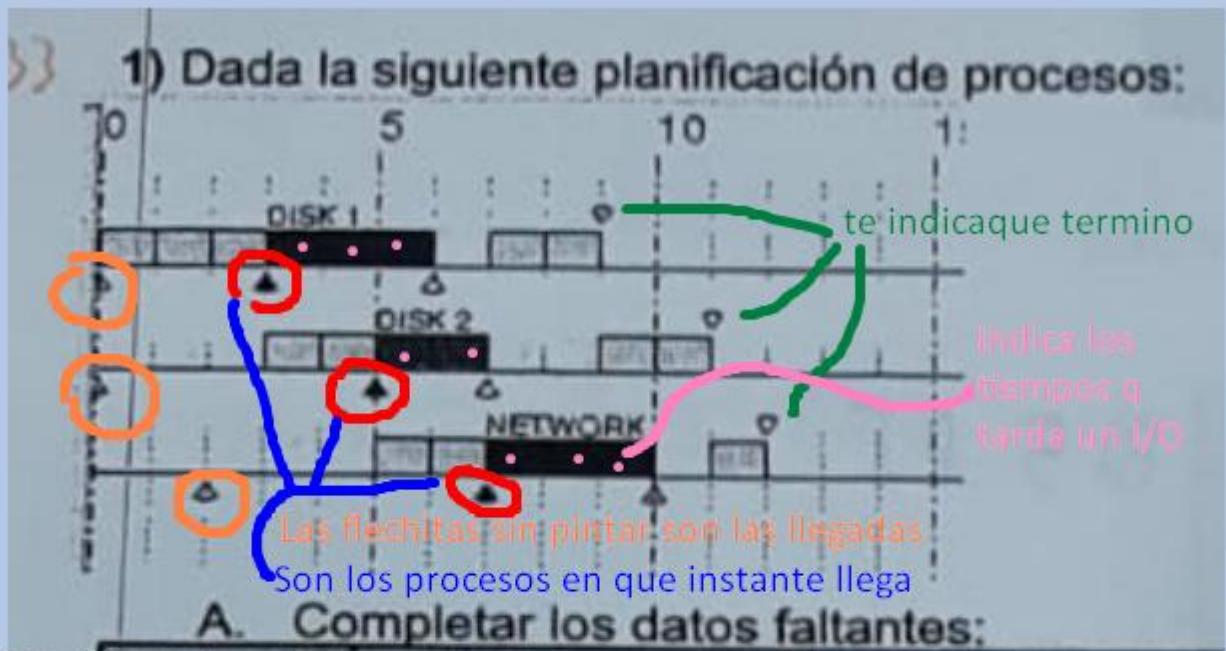


ISO resumen total



Cause





A. Completar los datos faltantes:

Proceso	Instante de llegada	CPU	I/O (instante, duración)
Tarea 1	0	5	(Disk 1, 3, 3)
Tarea 2	0	4	(Disk 2, 2, 2)
Tarea 3	2	3	(Network, 3, 3)

B. Indicar cuál(es) es el algoritmo utilizado:

FCFS	SJF	Round Robin, Q=3, TV	Round Robin, Q=2, TV

C. Complete el siguiente análisis:

Proceso	Tiempo de Retorno	Tiempo de Espera	Tiempo Promedio de retorno: 10
Tarea 1	9	4	
Tarea 2	11	5	Tiempo Promedio de espera: 6
Tarea 3	10	3	

D. En general, para evaluar qué tan rápido es un Sistema para terminar la ejecución de un conjunto de procesos la métrica a utilizar es:

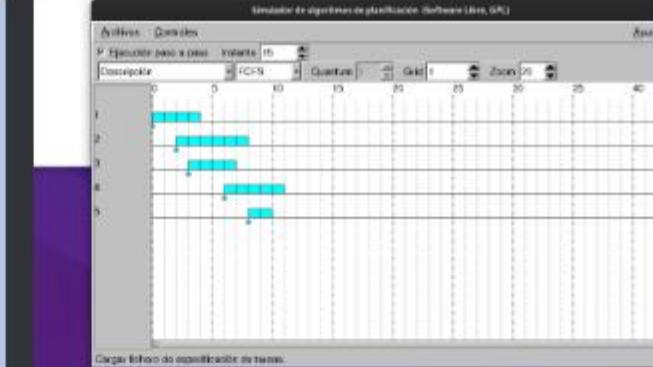
Tiempo promedio de retorno Tiempo promedio de Espera Ambas Ninguna

E. En general, para evaluar qué tan rápido es el tiempo de respuesta de un Sistema, la métrica a utilizar es:

Tiempo promedio de retorno Tiempo promedio de Espera Ambas Ninguna

F. ¿En general, cuál de los siguientes algoritmos de administración de CPU podrían causar inanición?

SJF SRJT FCFS Round Robin, TV Round Robin, TF



para mi es:

triangulo sin rellenar: LLEGADA PROCESO

triangulo con relleno: INICIA OPERACION I / O

el proceso finaliza con el triangulo pero en la parte de arriba
la operacion de i/o finaliza con el triangulo en la parte de abajo

gianca la operacion de i/o finaliza con el triangulo en la parte de ...

La SemaNa hoy a las 2:03

También es sin rellenar no?

B) DIRECCION LOGICA 11264:

PAG = 11264 DIV 2048 = 5. DIRECCION INVALIDA, LA PAGINA 5 NO SE ENCUENTRA DENTRO DE LAS DIRECCIONES LOGICAS POSIBLES.

C) DIRECCION LOGICA 3072:

PAG = 3072 DIV 2048 = 1
DESPLAZAMIENTO = 3072 MOD 2048 = 1024
DIRECCION FISICA: BASE DEL MARCO APUNTADO POR LA PAGINA1 + DESPLAZAMIENTO
DIRECCION FISICA: 10240 + 1024 = 11264

D) DIRECCION LOGICA 0:

PAG = 0 DIV 2048 = 0
DESPLAZAMIENTO = 0 MOD 2048 = 0
DIRECCION FISICA: BASE DEL MARCO APUNTADO POR LA PAGINA1 + DESPLAZAMIENTO
DIRECCION FISICA: 12288

E) DIRECCION LOGICA 8704:

PAG = 8704 DIV 2048 = 4
DESPLAZAMIENTO = 512
DIRECCION FISICA: BASE DEL MARCO APUNTADO POR LA PAGINA1 + DESPLAZAMIENTO
DIRECCION FISICA: 0 + 512 = 512

Cada pagina ocupa 2KIB = $2 * 1024 = 2048$ Bytes.

Pagina	Dir Logica	Calculos aux
0	0..2047	$0 * 2048 = 0$
1	2048..4095	$1 * 2048 = 2048$
2	4096..6143 (esta producirá PF)	$2 * 2048 = 4096$
3	6144..8191	$3 * 2048 = 6144$
4	8192..10239	$4 * 2048 = 8192$

Marco	Dir Fisica	Calculos aux
6	12288..14335	$6 * 2048 = 12288$
5	10240..12287	$5 * 2048 = 10240$
-	PF	PF
1	2048..4095	$1 * 2048 = 2048$
0	0..2047	$0 * 2048 = 0$

A) DIRECCION LOGICA 5120:

PAG = 5120 DIV 2048 = 2, LA PAGINA 2 NO ESTA ASIGNADA EN NINGUN MARCO, POR LO TANTO NO ESTA EN MEMORIA Y SE PRODUCE PF.

Tp4: Procesos

FORMULA DEL CALCULO DEL TIEMPO DE RETORNO PARA TODOS ESTOS ALGORITMOS

((LIMITE SUPERIOR <) - (LIMITE INFERIOR >)) + 1

TIEMPO DE RETORNO= (LIMITE SUPERIOR - LIMITE INFIERIOR) +1

TIEMPO DE ESPERA=(CUANTO TIEMPO ESPERO A SER EJECUTADO) SERIA TR-CPU

Tp4: Procesos

FCFS: FIRST COME FIRST SERVED.....**FIFO**

SJF: SHORTEST JOB FIRST.....**PRIORIZAMOS EL PROCESO CON MENOR CPU**

RR TV: ROUND ROBIN, TV..... **NO REUTILIZA LOS QUATUMS(TIENE QUANTUMS)**

RR TF: ROUND ROBIN, TF.....**REUTILIZA LOS QUATUMS**

ALGORITMO DE PRIORIDADES... **SE LLEVA UNA QUEUE X PRIORIDAD, DE ARRIBA A ABAJO.. SE ATIENDE POR ORDEN DE PRIORIDAD..ES APROPIATIVO**

SRTF: SHORTER REMAIND TIME FIRST... PRIORIZAMOS EL PROCESO CON MENOR CPU Y EL Q SE ESTA EJECUTANDO LE VAMOS RESTANDO CPUS

Tp4: Procesos

(9+5+3+7) = 24, TENGÓ 24 instancias

FIRST COME FIRST SERVED

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9		>p1	p1																9	0								
P2	1	5			>								P2	P2	P2	P2	P2											13	8	
P3	2	3				>												P3	P3	P3								15	12	
P4	3	7					>													P4		21	14							
FCFS			QUEUE																									14,5	8,5	

En este algoritmo, se resuelve todo el proceso y despues se pasa al siguiente (mirar orden de llegada)

(9+5+3+7) = 24, TENGÓ 24 instancias

SHORTEST JOB FIRST

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9		>P1	P1	P1<															9	0								
P2	1	5			>											P2	P2	P2	P2	P2<								16	11	
P3	2	3				>										P3	P3	P3<										10	7	
P4	3	7					>															P4	P4	P4	P4	P4	P4<		21	14
SJF			QUEUE	P1	P2	P3	P4																							

En resumen este algoritmo toma el trabajo mas corto y ese va ser su orden de ejecucion

Se encolaN los procesos que se van a activando en llegadas, se van encolando, despues cuando termine el proceso se mira la Queue y se elige al proceso con menor CPU

Tp4: Procesos

(9+5+3+7) = 24, TENGO 24 instancias

ROUND ROBIN TIMER VARIABLE

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9		>P1	P1	P1	P1											P1	P1	P1	P1					P1<		24	0	
P2	1	5			>			P2	P2	P2	P2																		19	14
P3	2	3				>						P3	P3	P3<															9	6
P4	3	7					>								P4	P4	P4	P4						P4	P4	P4<			20	13
RR TV Q=4			QUEUE	P1	P2	P3	P4	P1	P2	P4	P1																	18	8,25	

En resumen este algoritmo se fija en el orden de llegada y ejecuta la rafaga de Quantums que este seteado, si no termina el proceso en esa rafaja se vuelve a encolar y se sigue con el siguiente proceso, se lleva una Queue para ir viendo que tengo que ejecutar

Se mira al Quantum

El límite es el Quantum

Si es menos no pasa nada

SE cambia al siguiente

No se necesita continuar

Desde el quantum anterior

(9+5+3+7) = 24, TENGO 24 instancias

ROUND ROBIN TIMER FIJO

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9		>P1	P1	P1	P1											P1	P1	P1	P1					P1<		21	12	
P2	1	5			>			P2	P2	P2	P2																	16	11	
P3	2	3				>						P3	P3	P3<														9	6	
P4	3	7					>							P4									P4	P4	P4		P4	21	14	
RR TF Q=4			QUEUE	P1	P2	P3	P4	P1	P2	P4	P1																16,75	10,75		

En resumen este algoritmo se ejecuta en rafagas de X quantums, si un proceso termina antes de terminar un quantum otro proceso debe aprovechar ese/esos instantes

Tp4: Procesos

(9+5+3+7) = 24, TENDO 24 instancias

ALGORITMO DE PRIORIDADES

ES APROPIATIVO

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9	3	>P1																P1	P1	P1	P1	P1	P1<		24	15		
P2	1	5	2		>P2					P2	P2	P2	P2<															8	3	
P3	2	3	1			>P3	P3	P3<																				3	0	
P4	3	7	2			>							P4	P4	P4	P4	P4	P4	P4<								13	6		
PRIORIDAD			QUEUE1	P3																							12	6		
			QUEUE2	P2	P2	P4																								
			QUEUE3	P1	P1																									

Se mira el orden de llegada y el nivel de prioridad (si el nivel de prioridad es mayor al proceso actual se cambia al proceso de mayor prioridad)

(9+5+3+7) = 24, TENDO 24 instancias

SHORTEST REMAINING TIME FIRST

ES APROPIATIVO

ES LA VERSION APROPIATIVA DEL SJF

PROCESO	LLEGADA	CPU	PRIORIDAD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	TR	TE
P1	0	9		>P1																P1	P1	P1	P1	P1	P1<		24	15		
P2	1	5			>P2					P2	P2	P2	P2<															8	3	
P3	2	3				>P3	P3	P3<																				3	0	
P4	3	7				>							P4	P4	P4	P4	P4	P4	P4<								13	6		
SRTF			QUEUE1	P1	P2	P1	P3	P2	P4																			12	6	
				P1	P2	P1	P3	P2	P4																					

8 4 7

8 7

8 7

8

Es la version apropiativa del SJF, se fija el proceso de menor CPU y va restando por cada ejecucion de ese proceso, si despues de descontar su valor de CPU hay otro con mayor prioridad, me fijo en ese

En resumen tenes que ir viendo/ descontando CPUs una vez que ya se ejecuto un instante de cada proceso, es solo mirar la QUEU y el siguiente va ser el menor de todos esos

Tp5

Memoria

a) Si se disponen de 5 marcos. ¿Cuántos fallos de página se producirán si se utilizan las siguientes técnicas de selección de víctima? (Considere una política de Asignación Dinámica y Reemplazo Global)

- i) Segunda Chance
- ii) FIFO
- iii) LRU
- iv) OPT

b) Suponiendo que cada atención de un fallo se pagina requiere de 0,1 seg.

FIFO: FIRST IN FIRST OUT

Un LRU (Least Recently Used) es una estrategia de reemplazo de caché que se utiliza para gestionar qué elementos eliminar cuando una caché alcanza su capacidad máxima. Se basa en el principio de eliminar el elemento que no se ha utilizado recientemente.

SEGUNDA CHANCE: marcar ticks y mandar al final

LRU: Least Recently Used... EL QUE MIRA AL ACTUAL

OPTIMO: EL QUE MIRA AL FUTURO

Tp5

Memoria

Asignacion Dinamica y Reemplazo Global

EJE22

1, 2, 15, 4, 6, 2, 1, 5, 6, 10, 4, 6, 7, 9, 1, 6, 12, 11, 12, 2, 3, 1, 8, 1, 13, 14, 15, 3, 8

FIFO																													Total PF	
MARCOS	1	2	15	4	6	2	1	5	6	10	4	6	7	9	1	6	12	11	12	2	3	1	8	1	13	14	15	3	8	
1	1	1	1	1	1	1	1	5	5	5	5	5	5	5	5	6	6	6	6	1	1	1	1	1	1	1	3	3		
2		2	2	2	2	2	2	2	2	10	10	10	10	10	10	10	10	12	12	12	12	12	8	8	8	8	8	8		
3			15	15	15	15	15	15	15	15	15	15	15	7	7	7	7	7	11	11	11	11	11	11	13	13	13	13		
4				4	4	4	4	4	4	4	4	4	4	9	9	9	9	9	9	2	2	2	2	2	14	14	14	14		
5					6	6	6	6	6	6	6	6	6	6	1	1	1	1	1	3	3	3	3	3	3	15	15	15		
PF	X	X	X	X	X					X				X	X	X	X	X		X	X	X	X		X	X	X	X	0.1 * 21 = 2,1 seg	
QUEUE	4	2	15	4	6	5	10	7	9	1	6	12	11	2	3	4	8	13	14	15	3									
BIT REFERE																														

Normalon cargar los marcos y despues si el proxima pagina ya se encontraba en memoria no se hace literalmente, es copiar y pegar la columna. Si no se encontraba se marca como PF y se apila en la Queue y la victima va ser la primera que este en la Queue

Segunda Chance	2																												Total PF	
MARCOS	1	2	15	4	6	2	1	5	6	10	4	6	7	9	1	6	12	11	12	2	3	1	8	1	13	14	15	3	8	
1	1	1	1	1	1	1	1*	1	1	1	4	4	4	4	4	4	12	12	12*	12*	12*	12*	12*	12	12	12	12	15	15	
2		2	2	2	2	2*	2*	2	2	2	2	2	7	7	7	7	7	11	11	11	11	11	11	8	8	8	8	3	3	
3			15	15	15	15	15	15	5	5	5	5	5	5	9	9	9	9	9	9	2	2	2	2	13	13	13	13		
4				4	4	4	4	4	4	4	10	10	10	10	10	10	1	1	1	1	1	1	3	3	3	3	14	14		
5					6	6	6	6	6	6*	6*	6	6*	6*	6*	6*	6*	6	6	6	6	6	1	1	1*	1*	1*	1		
PF	X	X	X	X	X					X	X			X	X	X		X	X		X	X	X		X	X	X	X	0.1 * 22 = 2,2 seg	
QUEUE	1*	2*	15	4	6*	4	2	5	10	6*	4	7	9	1	6	12*	11	2	3	1*	12	8	13	14	1	15	3	8		
BIT REFERE																														

Si viene una pagina que ya esta en el cause, copio y pego la columna y le pongo un tick Esa es la segunda chance

Si viene una pagina que ya esta en el cause y ya esta marcada, copio y pego la columna

Si viene una pagina que ya esta en el cause y no tiene marca, se toca la marca

Tp5 Memoria

Asignacion Dinamica y Reemplazo Global

LRU	2																						Total PF						
MARCOS	1	2	15	4	6	2	1	5	6	10	4	6	7	9	1	6	12	11	12	2	3	1	8	1	13	14	15	3	8
1	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	11	11	11	11	11	8	8	8	8	3	3		
2		2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	12	12	12	12	12	12	12	13	13	13	13	13	
3			15	15	15	15	15	5	5	5	5	5	5	9	9	9	9	9	9	2	2	2	2	2	14	14	14	14	
4				4	4	4	4	4	4	10	10	10	10	10	1	1	1	1	1	1	3	3	3	3	15	15	15	15	
5					6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	1	1	1	1	1	1	1	8	
PF	x	x	x	x	x			x	x		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	0.1 *22=2,2 seg		
QUEUE	1	2	15	4	6	2	1	5	6	10	4	6	7	9	1	6	12	11	12	2	3	1	8	1	13	14	15	3	8
BIT REFERE																													

Llevas una Queue y vas a tachando, este algoritmo consiste en ir reemplazando por el que es mas actual

Llega uno que ya esta cargado en memoria, lo quitas de la queue y lo mandas full derecha

Este es algoritmo que va mirando en que pos actual aparece la pagina, **NO ES EL DEL FUTURO**

Es nuevo? Miras a la izquierda y que este libre y lo reemplizas por ese

OPTIMO	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	16º	17º	18º	19º	20º	21º	22º	23º	24º	25º	26º	27º	28º	29º	Total PF
MARCOS	1	2	15	4	6	2	1	5	6	10	4	6	7	9	1	6	12	11	12	2	3	1	8	1	13	14	15	3	8	
1	1	1	1	1	1(7)	1(7)	1(15)	1(15)	1(15)	1(15)	1(15)	1(15)	1(15)	1(15)	1(15)	1(22)	1(22)	1(22)	1(22)	1(22)	1(22)	1(22)	1(X6)	1(X6)	1(X6)	1(X6)	1(X6)			
2		2	2	2	2(6)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(20)	2(X5)									
3			15	15	15(27)	15(27)	15(27)	5(X1)	5(X1)	10(X1)	10(X1)	10(X1)	7(X1)	9(X1)	9(X1)	9(X1)	12(19)	12(19)	12(X4)	12(X4)	12(X4)	12(X4)	12(X4)	13(X4)	14(X4)	15(X4)	15(X4)			
4				4	4(11)	4(11)	4(11)	4(11)	4(11)	4(11)	4(X2)	4(X2)	4(X2)	4(X2)	4(X2)	4(X2)	4(X2)	11(X2)	11(X2)	3(28)	3(28)	3(28)	3(28)	3(28)	3(X7)	3(X7)				
5					6(9)	6(9)	6(9)	6(9)	6(12)	6(12)	6(12)	6(16)	6(16)	6(16)	6(X3)	6(X3)	6(X3)	6(X3)	8(29)	8(29)	8(29)	8(29)	8(29)	8(29)	8(29)	8(X8)				
PF	x	x	x	x	x			x			x			x	x		x		x	x	x	x	x	x	x	x	16			
QUEUE																														
BIT REFERE																														

Es el algoritmo que mira a futuro

1 ro, llena los marcos, si viene un repetido se actualiza su valor temporal asociado(mirar a futuro)

2 DO, si el valor que viene se encuentra en el cause, pero no tiene valor a futuro, se lo modela como X

3 ro, si la victimia ya tiene un X y entra una nueva pagina que no tiene proximo el X lo hereda(con su prioridad)

el reemplazo se hace con el valor mas grande. Si hay un X se lo reemplaza por ese ()

PRESTA ATENCION CON LA POLITICA QUE PIDEN

- POLITICA: ASIGNACION DINAMICA Y REEMPLAZO GLOBAL
SEGUNDA CHANCE, FIFO, LRU, OPT... SON COMO VENGO
HACIENDOLOS NORMALES
- POLITICA: ASIGNACION FIJA CON REPARTO EQUITATIVO Y REEMPLAZO LOCAL

PRESTAR ATENCION A LOS MARCOS QUE TENGO Y LA CANTIDAD
UNIVOCA DE PAGINAS.. ENTONCES HAGO
cantMarcos / cantPagUnivocas = cantidad de Queues

EJEMPLO DE POLITICA: ASIGNACION FIJA CON REPARTO EQUITATIVO Y REEMPLAZO LOCAL

	1 ^a	2 ^a	3 ^a	4 ^a	5 ^a	6 ^a	7 ^a	8 ^a	9 ^a	10 ^a	11 ^a	12 ^a	13 ^a	14 ^a	15 ^a	16 ^a	17 ^a	18 ^a	19 ^a	20 ^a	21 ^a	22 ^a	23 ^a	24 ^a	25 ^a	26 ^a	27 ^a	28 ^a	29 ^a	30 ^a			
MARCOS	B2	B4	A1	A3	A1	C1	C2	B6	B2	B4	A2	A4	A1	C4	C8	B1	B8	C6	C1	C4	C1	A5	A1	A4	B3	B1	B8	A7	A9	A4	CANT. DE FALLOS		
1	B2	B2*	B2*	B2*	B2*	B2*	B2*	B2*	B2*	B2*	B2	B8	B8*	B8*	B8*	B8*	20																
2		B4	B4*	B4	B3	B3	B3	B3																									
3			A1	A1	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A1	A1*	A1	A9	A9																	
4			A3	A3	A4	A4*																											
5					C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C8	C8	C8	C8	C4														
6						C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C2	C6															
7								B6	B6	B6	B6	B6	B6	B6	B6	B6	B1	B1*	B1*	B1*	B1*	B1*											
8												A2	A5	A5	A5	A5	A5	A7	A7														
9															C4	C4	C4	C4	C4	C1	C1	C1*											
FALLO DE PÁG.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					
COLA_A			A1	A1	A*	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A1*	A2	A1*	A1*	A1*	A1*	A4*											
			A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A3	A7										
												A2																					
															A1																		
																	A4																
COLA_B	B2	B2*	B2	B2	B2	B4	B4	B4	B4	B4	B4	B1*	B1*	B1*	B1*																		
	B4	B4*	B4	B4	B4	B1	B8*	B8*	B8*	B8*																							
																		B6	B6														
COLA_C															C1	C1																	
															C2	C2																	
															C4	C4																	
															C8	C8																	

LOGICA:

Al ser de asignacion fija y reemplazo local, en criollo te dice
 De la cantidad total de marcos dividilas por cada letra unitaria que tengas(paginas)
 Tenemos A, B, C
 $9/3 = 3$
 Entonces tenemos 3 Marcos para Cada proceso
 Ahora el algoritmo de Segunda chance (es el mismo) PERO se aplica a cada
 Proceso de forma independiente pero siguiendo el HILO de los q van llegando

Tp5 Memoria

Descarga Asincronica

LAU	1	2	4	2	1M	3	4M	1M	6	2M
M1	1	1	1	1	1M	1M	1M	1M	1M	1M
M2		2	2	2	2	4M	4M	4M	DA	
M3		4	4	4	3	3	3	6	6	
M4	DA	2M								
PF	X	X	X		X	X		X	X	
QUEU	X	Z	Y	Z	X	Y	4M	1M	6	2M
- OCURRE "DA" CUANDO LA VICTIMA ES UNA PAGINA MODIFICADA <u>NO IGUAL</u>										
- CUANDO SE VA A "DA" LA PAGINA, LA SACO DE LA QUEU										
- SI LA PAGINA QUE INGRESA YA ESTA EN EL CAUSE, SOLO ENTRA MODIFICADA, EN LA QUEU, SE LA MANDA FULL DERECHA										

LRU

Cuando la pagina victim es una pagina modificada (ahi se habilita el swap/DA, la pagina modificada entrante, debe ser Distinta a la victim, si tenes 1M y te llega 1M no se activa el DA)

Cuando se va para Swap/DA la pagina modificada(q estaba en el cause), la saco de la queue

Cuando te viene una pagina modificada y en memoria principal esa pagina no esta modificada, se cambia a modificada y nada mas si es FIFO, se refleja en la Que como modificada. (Si es LRU, se tacha y mueve full derecha en la Queue)

3. Considere un esquema de paginación bajo demanda y complete la siguiente asignación de marcos de memoria según el algoritmo LRU, con 4 marcos, destinando uno para descarga asincrónica y determine la cantidad de page fault.

Página	1	2	4	2	1M	3	4M	1M	6	2M
M1	1	1	1	1	1M	1M	1M	1M	1M	1M
M2		2	2	2	2	4M	4M	4M	///	
M3			4	4	4	3	3	3	6	6
M4	DA	2M								
PF?	X	X	X		X	X		X	X	
QUEU	X	Z	Y	Z	X	Y	4M	1M	6	

7 PF

Cuando ingresa una pagina y en memoria esa pagina esta modificada, no se hace nada si es FIFO, se copia y pega la columna

Ahora si es LRU, se tacha y se pone full derecha en la Queue

Tp5 Memoria

Descarga Asincronica

FIFO

Cuando la pagina victim es una pagina modificada (ahi se habilita el swap/DA, la pagina modificada entrante, debe ser Distinta a la victim, si tenes 1M y te llega 1M no se activa el DA)

Cuando se va para Swap/DA la pagina modificada(q estaba en el cause), la saco de la queue

Cuando te viene una pagina modificada y en memoria principal esa pagina no esta modificada, se cambia a modificada y nada mas si es FIFO, se refleja en la Que como modificada. (Si es LRU, se tacha y mueve full derecha en la QEueue)

Cuando ingresa una pagina y en memoria esa pagina esta modificada, no se hace nada si es FIFO, se copia y pega la columna

Ahora si es LRU, se tacha y se pone full derecha en la Queue

Tp5 Calculos Marcos/Paginas/Direcciones

- Memoria administrada por sistema de paginación
- Tamaño de página → 515 Bytes
- Cada dirección de memoria referencia a 1 Byte
- Los *marco* en memoria principal se encuentran desde la dirección física 0
- Tenemos un proceso con un tamaño de 2000 Bytes y con la siguiente tabla de páginas

Página	Marco
0	6
1	5
2	-
3	1
4	0

Direccion logicas a Fisicas

Tengo paginas de 2 KIB = a 2048 kibbytes

A: DIRECCION LOGICA 5120

nro de pagina = $5120 \text{ div } 2048 = 2$ en la pagina 2 tenemos que no tiene un marco asignado

Desplazamiento = $5120 \text{ MOD } 2048 = 1024$

La direccion fisica es: 0 (aca es la base del frame) + 1024 = No hay base de frame, porq no hay marco asignado

MARCO | INICIO

0 | 0 - 2047

1 | 2048 - 4095

2 | 4096 - 6143

3 | 6144 - 8191

4 | 8192 - 10239

5 | 10240 - 12287

6 | 12288 - 14335

Al no tener un marco asignado, ocurre PF

Página	Marco
0	1
1	2
2	3
3	0

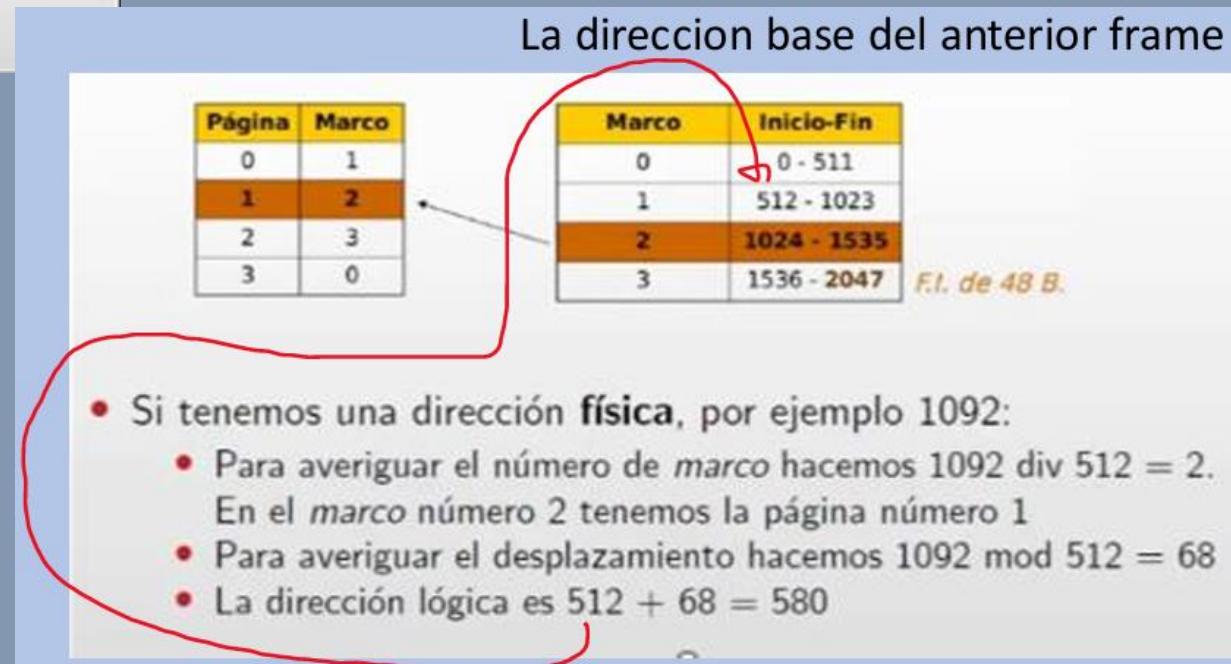
Marco	Inicio-Fin
0	0 - 511
1	512 - 1023
2	1024 - 1535
3	1536 - 2047

F.I. de 48 B.

- Si tenemos una dirección lógica, por ejemplo 580:
 - Para averiguar el número de página hacemos $580 \text{ div } 512 = 1$. Luego esta dirección corresponde a la página 1 que se encuentra en el *marco* 2
 - Para averiguar el desplazamiento hacemos $580 \text{ mod } 512 = 68$
 - La dirección física es $1024 + 68 = 1092$

Tp5 Calculos Marcos/Paginas/Direcciones

- Memoria administrada por sistema de paginación
- Tamaño de página → 515 Bytes
- Cada dirección de memoria referencia a 1 Byte
- Los *marco* en memoria principal se encuentran desde la dirección física 0
- Tenemos un proceso con un tamaño de 2000 Bytes y con la siguiente tabla de páginas



Dirección Virtual = (Número de Página × Tamaño de Página) + Desplazamiento

Tp5 Memoria virtual calcular cantidad de marcos q se le asignan a cada proceso

Tecnica Asignacion fisica
Con reparto proporcional



4) 1,5 pts. Suponga un SO con administración de memoria virtual por medio de paginación por demanda. Si la cantidad de marcos disponibles para los procesos es 30, indique cuántos marcos se le asignan a cada proceso si utiliza la técnica de asignación fija con reparto proporcional:

Proceso	Páginas del Proceso	Marcos Asignados
1	10	6
2	15	9
3	25	15

SH

Se suma las páginas del proceso
 $10 + 15 + 25 = 50$

$$(\text{paginasDelProceso}(10) / 50) * \text{marcos disponibles}(30) = 6$$

$$(15 / 50) * 30 = 9$$

$$(25 / 50) * 30 = 15$$

Tecnica Asignacion fisica
Con reparto equitativo



Reparto Equitativo:

- Se divide el total de marcos entre la cantidad de procesos.
- Cálculo:
$$\text{Cálculo: Marcos por proceso} = \frac{40}{4} = 10$$
- Cada proceso recibe 10 marcos.

Tp6

Discos

Algunas formulas

Prefijos - Equivalencias

Unidades básicas de información (en bytes)				
Prefijos del Sistema Internacional			Prefijo binario	
Múltiplo - (Símbolo)	Estándar SI	Binario	Múltiplo - (Símbolo)	Valor
kilobyte (kB)	10^3	2^{10}	kibibyte (KiB)	2^{10}
megabyte (MB)	10^6	2^{20}	mebibbyte (MiB)	2^{20}
gigabyte (GB)	10^9	2^{30}	gibibbyte (GiB)	2^{30}
terabyte (TB)	10^{12}	2^{40}	tebibbyte (TiB)	2^{40}

Unidades de medida de almacenamiento		
Medida	Símbolo	Equivalente
Byte	byte	8 bits
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 KB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

BPA

AUMENTA

Tp6

Discos

Capacidad de un HDD

- Supongamos un disco con 6 platos, 2 caras útiles, 1500 pistas por cara y 700 sectores por pista de 256 bytes cada uno
- Si queremos calcular la capacidad total del disco, hacemos:

$tamaño_disco = \#caras * \#pistas_cara * \#sectores_pista * tamaño_sector$

$$(6 * 2) * 1500 * 700 * 256 \text{ bytes} = 3225600000 \text{ bytes}$$

= 3.00407 GiB (Gibibytes)

Ocupacion sobre un HDD

- Supongamos un disco con 6 platos, 2 caras útiles, 1500 pistas por cara y 700 sectores por pista de 256 bytes cada uno
- Si queremos cuantas caras ocupará un archivo de 513 Mibbytes almacenado de manera contigua a partir del primer sector de la primera pista de una cara determinada:
 - Calculamos la capacidad de 1 cara:
 $1500 * 700 * 256 \text{ bytes} = 268800000 \text{ bytes}$
 - Dividimos el tamaño del archivo por la capacidad de una cara:
 $513 \text{ MiB} = 537919488 \text{ bytes}$
 $537919488 / 268800000 = 2.00118 \rightarrow 3 \text{ caras}$

Tp6

Discos

Tiempo de acceso a un HDD

- Supongamos un disco con 6 platos, 2 caras útiles, 1500 pistas por cara y 700 sectores por pista de 256 bytes cada uno. El disco gira a 12600 RPM , tiene un tiempo de posicionamiento (seek) de 2 milisegundos y una velocidad de transferencia de 15 Mib/s (Mebibits por segundo)
- Si queremos saber cuantos milisegundos se tardarían en transferir un archivo almacenado de manera contigua y aleatoria de 4500 sectores

1º ↓

2º ↗

$$\begin{array}{r} 12600 \quad | \quad 60000 \\ \times \quad \quad \quad \diagdown \\ 0,5 \quad x = 2,3809 \text{ MS} \end{array}$$

- Calculamos los datos que faltan:

- Latencia:

$$12600 \text{ vueltas} \rightarrow 1' = 60 \text{ s} = 60000 \text{ ms}$$
$$0,5 \text{ vueltas} \rightarrow x = 2,3809 \text{ ms}$$

- Transferencia:

$$15 \text{ Mibits} \rightarrow 1 \text{ s} = 1000 \text{ ms}$$
$$256 \text{ bytes} \rightarrow x$$

Unificamos unidades:

$$15728640 \text{ bits} \rightarrow 1000 \text{ ms}$$

$$2048 \text{ bits} \rightarrow x = 0,1302 \text{ ms}$$

8 bits = 1 byte...

Necesitamos bits entonces hacemos $8 * 256 = 2048$ bits

- Datos obtenidos:
 - Seek time: 2 ms
 - Latency time: 2,3809 ms
 - Tiempo transferencia bloque: 0,1302 ms
 - #bloques: 4500 → eventualmente se tienen que calcular

- Resultados:

- Almacenamiento secuencial:

$$\text{seek} + \text{latency} + \text{tiempo.transferencia.bloque} * \# \text{bloques}$$
$$2 + 2,3809 + 0,1302 * 4500 = 590,2809 \text{ ms}$$

- Almacenamiento aleatorio:

$$(\text{seek} + \text{latency} + \text{tiempo.transferencia.bloque}) * \# \text{bloques}$$
$$(2 + 2,3809 + 0,1302) * 4500 = 20299,95 \text{ ms}$$

Tp6

Discos

Calcular latencia: $60/\text{RPM} * 2$

Calcular Velocidad (en bits): Unidad de medición (EJ Mib/s) * cantidad.
EJ: $15 \text{ Mib/s} = 15 * 2^{20} = 15 * 1,048,576 = 15,728,640 \text{ bits/segundo} = V$

Calcular transferencia: $(\text{tamaño Sector en bits}) / V (* 1000) = \text{tiempo en ms}$

Tiempo total de acceso: Seek + latencia + transferencia * bloques los () varían

Almacenamiento total: $(\text{platos} * \text{caras} / \text{total caras}) * \text{pistas} * \text{sectores} * \text{tamaño sectores}$

Almacenamiento por cara: pistas por cara * sectores por pista * tamaño sector

Caras ocupadas por un archivo: tamaño del archivo / capacidad por cara

(c) **Calcule el tiempo de transferencia real de un archivo de 15 MiB(Mebibytes). grabado en el disco de manera secuencial (todos sus bloques almacenados de manera consecutiva)**

9000 vueltas = 60 segundos = 60000ms

Tiempo en dar una vuelta = $60 / 9.000 = 0.0067 \text{ segundos} = 6.66 \text{ ms}$

O la otra es hacer

Tiempo en dar una vuelta = $60.000 / 9.000 = 6.66 \text{ ms}$

Tiempo por media vuelta = $6.66 \text{ ms} / 2 = 3.33 \text{ ms}$

Latencia = $0.5 * \text{tiempo por vuelta} = 0.5 * 6.66 \text{ ms} = 3.33 \text{ ms}$

Velocidad de transferencia = $10\text{MiB} = 10 * 1024 * 1024 = 10,485,760 \text{ bytes/s.}$

Tamaño bloque : 512 bytes

Tiempo por bloque = TAMAÑO DEL BLOQUE / VELOCIDAD DE TRANSFERENCIA

Tiempo por bloque = $512 / 10,485,760 \text{ bytes} = 0.0488 \text{ ms}$

Numero de bloques = Tamaño archivo (bytes) / Tamaño de un bloque (bytes)

Tamaño del archivo = $1\text{MiB} = 15 \times 1024 \times 1024 = 15,728,640 \text{ bytes}$

Tamaño de un bloque = 512bytes

Numero de bloques = $15,728,640 \text{ bytes} / 512 \text{ bytes} = 30.720 \text{ bloques}$

- Almacenamiento secuencial:

$\text{seek} + \text{latency} + \text{tiempo.transferencia.bloque} * \# \text{bloques}$
 $2 + 2.3809 + 0.1302 * 4500 = 590.2809 \text{ ms}$

- Almacenamiento aleatorio:

$(\text{seek} + \text{latency} + \text{tiempo.transferencia.bloque}) * \# \text{bloques}$
 $(2 + 2.3809 + 0.1302) * 4500 = 20299.95 \text{ ms}$

Almacenamiento Secuencial:

Seek = 10ms (lo da el enunciado), Latenci = 3.33ms, Tiempo por bloque: 0.0488ms

Numero de bloques = 30.720

Reemplazando en la formula de almacenamiento secuencial:

$$10 + 3.33 + 0.0488 * 30.720 = 1.513,33 \text{ ms} = 1.5 \text{ s}$$

Tp6

Algoritmos de planificación

Algoritmos de planificación en un HDD

- Objetivo: minimizar el movimiento de la cabeza
- Como: ordenando lógicamente los requerimientos pendientes (*que estan en la cola*) al disco, considerando el número de cilindro de cada requerimiento. En cualquier momento se pueden encolar nuevo movimientos
- La atención de requerimientos a pistas duplicadas se resuelven según el algoritmo de planificación:
 - **FCFS**: se atienden de manera separada (tantas veces como se requieran). Por ejemplo, si tengo {10, 40, 70, 10}, al 10 lo atiendo 2 veces
 - **SSTF/SCAN/LOOK/C-SCAN/C-LOOK**: se atienden de manera consecutiva

Algoritmos - Ejemplo de enunciado sin page faults

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

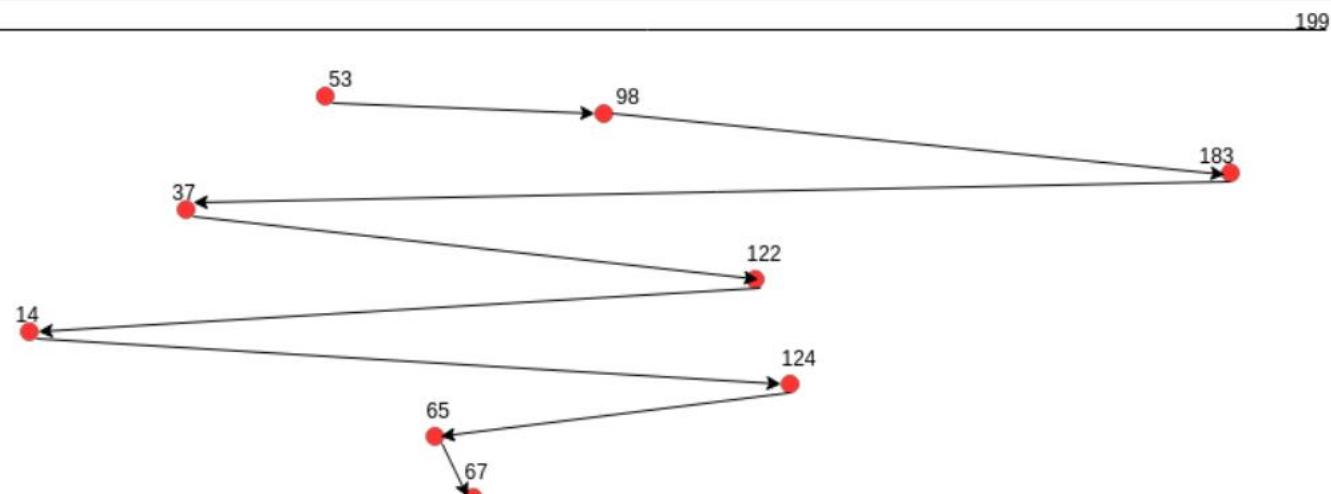
Ejemplos FCFS

ESTOS SON SIN
PAGE FAULD?

Sí

ESTOS SON SIN PF

- **FCFS:** atiende los requerimientos por orden de llegada



First Come First Served

Tp6 Algoritmos de planificación

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

Como sacar la cantidad de movimientos:

La cantidad de movimientos en el algoritmo FCFS (First Come First Served), que atiende los requerimientos por orden de llegada, se calcula sumando la distancia entre las posiciones de los cilindros que visita el cabezal, en el orden en que se presentan las solicitudes.

Para calcular los 640 movimientos del ejemplo, se realiza lo siguiente:

1. **Lista de solicitudes:** Las posiciones de los cilindros que debe visitar son: 53, 98, 183, 37, 14, 122, 124, 65, 67 .
 2. **Posición inicial:** El cabezal comienza en el cilindro 0.
 3. **Cálculo de movimientos:**
 - Del 0 al 53: $|53 - 0| = 53$
 - Del 53 al 98: $|98 - 53| = 45$
 - Del 98 al 183: $|183 - 98| = 85$
 - Del 183 al 37: $|37 - 183| = 146$
 - Del 37 al 14: $|14 - 37| = 23$
 - Del 14 al 122: $|122 - 14| = 108$
 - Del 122 al 124: $|124 - 122| = 2$
 - Del 124 al 65: $|65 - 124| = 59$
 - Del 65 al 67: $|67 - 65| = 2$
 4. **Suma total de movimientos:** $53 + 45 + 85 + 146 + 23 + 108 + 2 + 59 + 2 = 640$.
- Así se obtiene el total de movimientos. El gráfico ayuda a visualizar estos recorridos. ¿Te gustaría realizar el cálculo para otro conjunto de datos o algoritmo?

Ejemplos SSTF

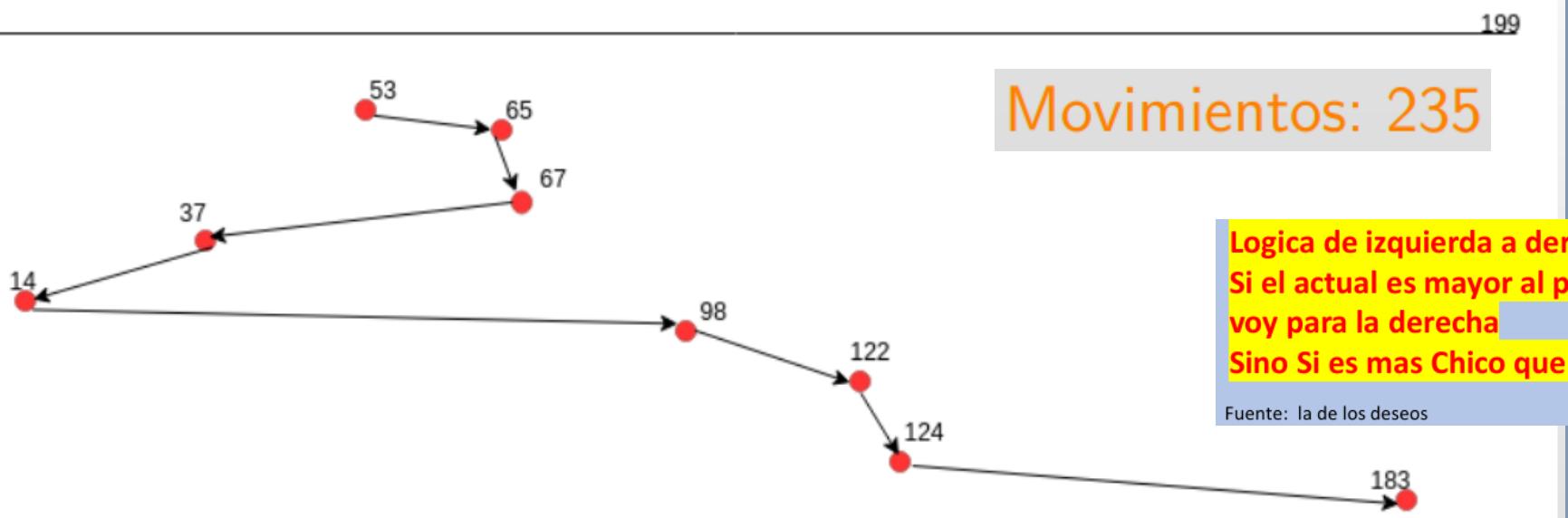
Agarro de la Queue el valor minimo para ser mi proximo (minimo desde donde estoy parado)

ESTOS SON SIN PF

Tp6 Algoritmos de planificacion

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

- **SSTF**: selecciona el requerimiento que requiere el menor movimiento del cabezal



Logica de izquierda a derecha:
Si el actual es mayor al proximo numero (menor movimiento de cabezal)
voy para la derecha
Sino Si es mas Chico que el actual me voy para la izquierda

Fuente: la de los deseos

Sortest Seek Time First

Ejemplos SCAN

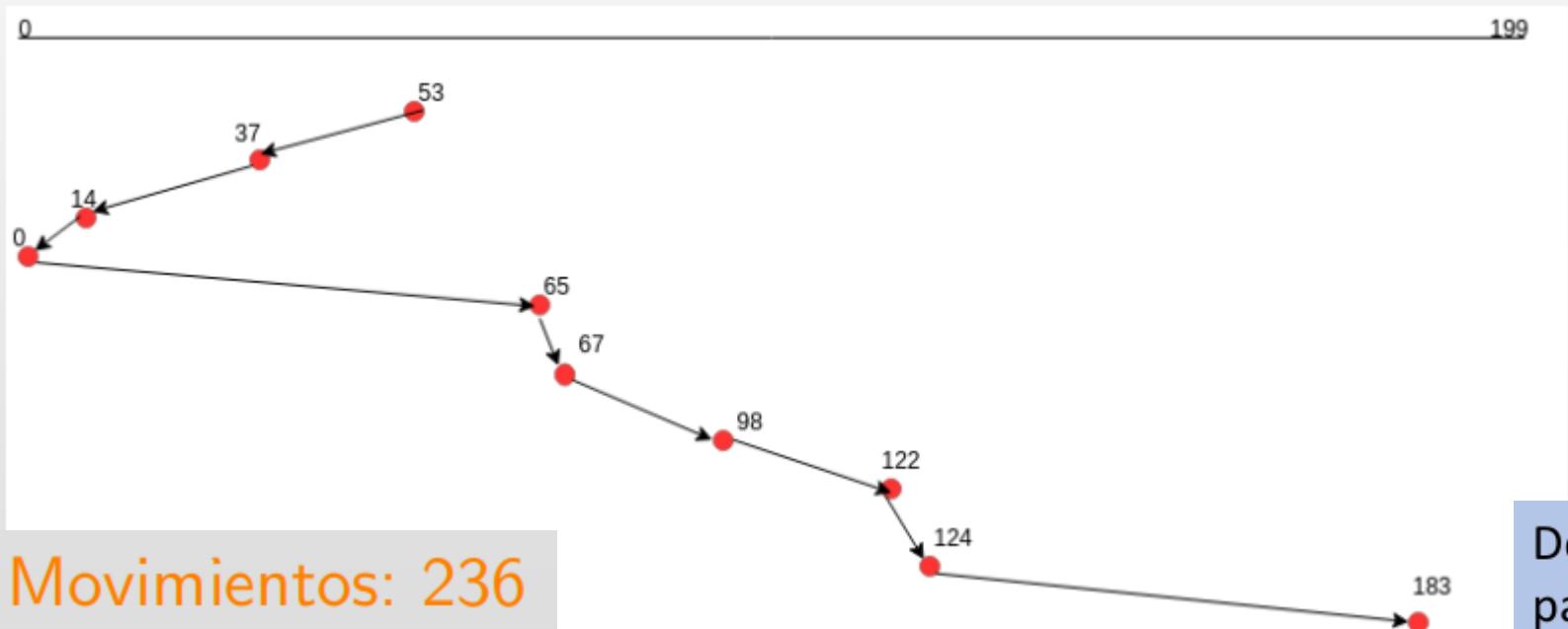
Fuente: la de los deseos

Recorro por los valores menores, cuando llego al **ultimo valor menor, llego al 0..**

Cambia de rumbo, para los valores mayores

Tp6 Algoritmos de planificación

- **SCAN:** barre el disco en una dirección atendiendo los requerimientos pendientes en esa ruta hasta llegar a la última pista del disco y cambia la dirección. Es **importante** saber en que pista se **está** y de que pista se **viene** para determinar el sentido del cabezal



Movimientos: 236

Llega hasta la ultima pista del disco (0) y despues cambia la direccion

Solo me importa la Queue, desde donde viene y desde donde empieza

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

La cantidad de movimientos, misma logica del anterior..
Primero hacer igual el grafico y despues sacar las cantidades

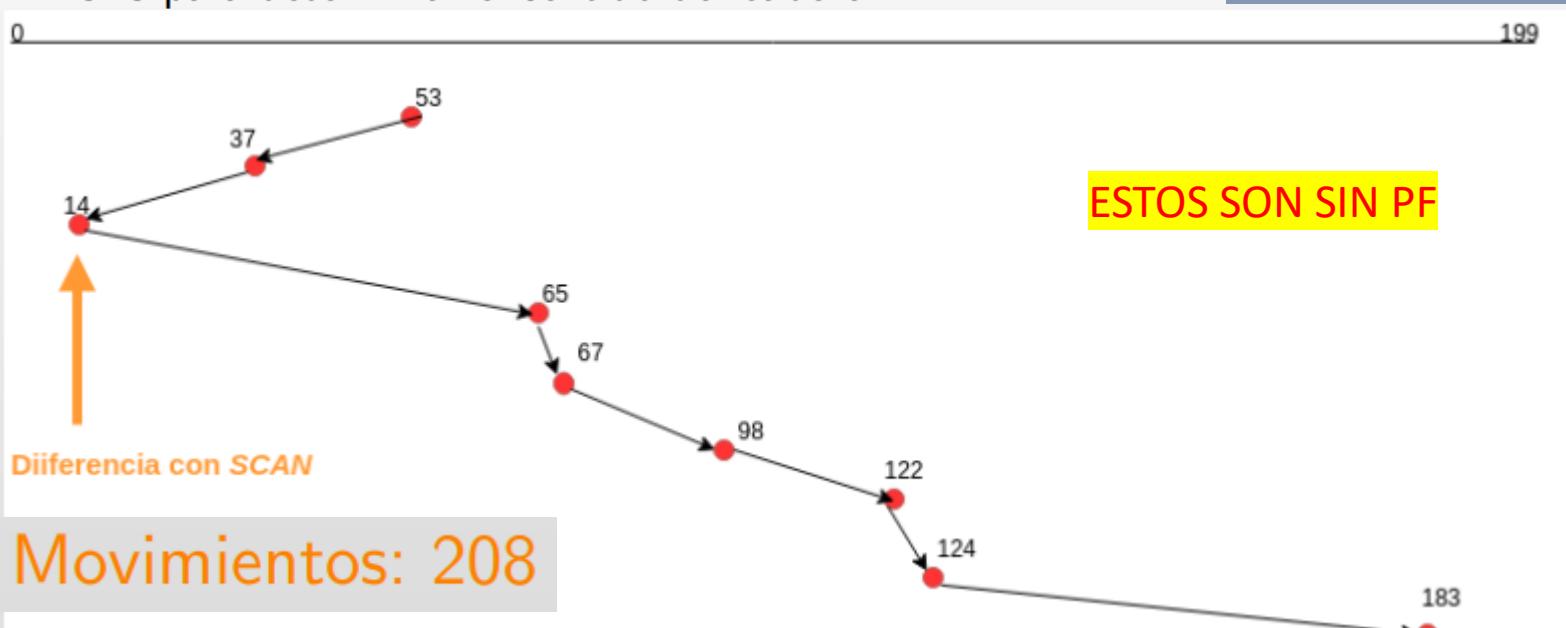
Desde donde viene la pista, me sirve para determiner el sentido del barrido
61 es un Mayor.. Venia de mayores (DERECHA)
Ahora tocan los numeros menores (IZQUIERDA)

SCAN

Ejemplos LOOK

Recorro por los valores menores, cuando **llego al ultimo valor menor**. Cambia de rumbo, para los valores mayores

- **LOOK:** se comporta igual que el *SCAN* pero no llega hasta la última pista del disco sobre la dirección actual sino que llega hasta el último requerimiento de la dirección actual. Es **importante** saber en que pista se **está** y de que pista se **viene** para determinar el sentido del cabezal



LOOK

Llega hasta la ultima pista del disco y despues cambia la direccion..
Es igual que el SCAN, SOLO LA DIFERENCIA ES Q NO LLEGA HASTA EL 0, LLEGA HASTA DONDE LE DA LA QUEUE

Solo me importa la Queue, desde donde viene y desde donde empieza

Tp6 Algoritmos de planificación

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

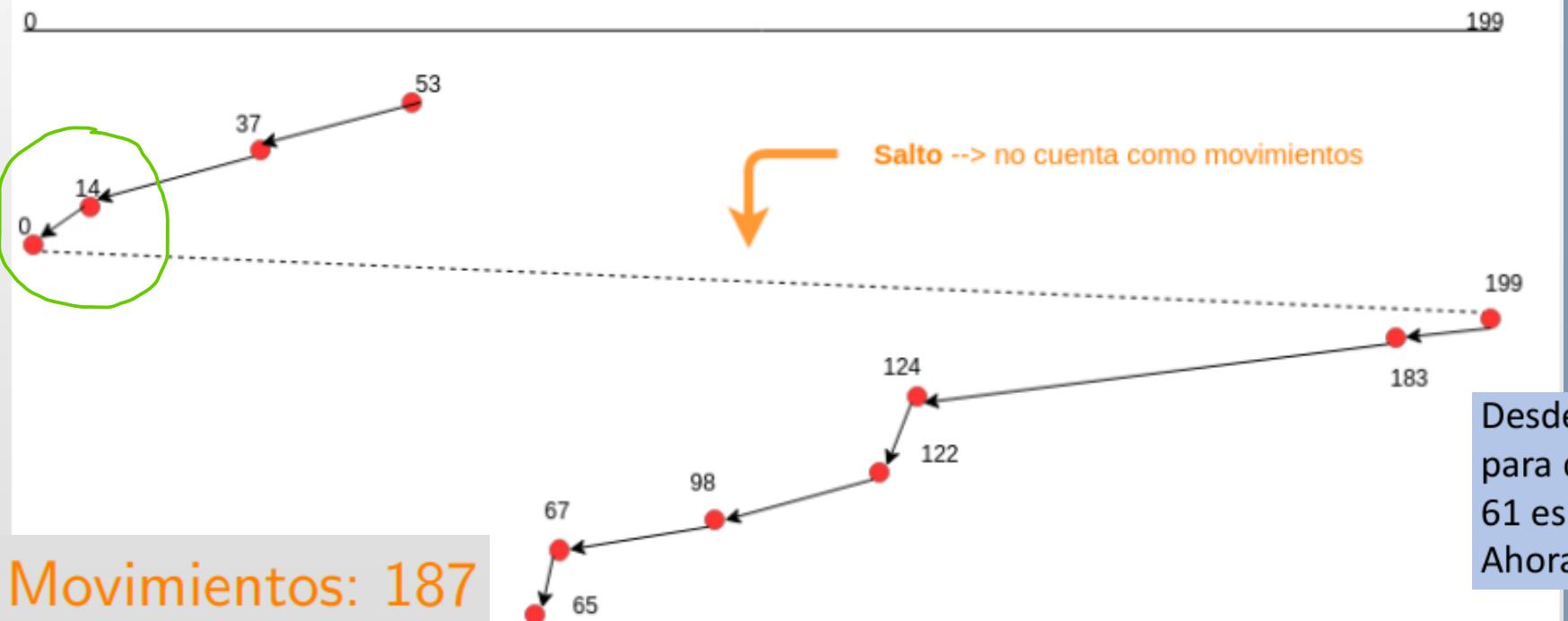
La cantidad de movimientos, misma logica del anterior..
Primero hacer igual el grafico y despues sacar las cantidades

Desde donde viene la pista, me sirve para determiner el sentido del barrido
61 es un Mayor.. Venia de mayores (DERECHA)
Ahora tocan los numeros menores (IZQUIERDA)

Ejemplos C-SCAN

Recorro por los valores menores, cuando **llego al ultimo valor menor**. Cambia de rumbo, para los valores mayores

- **C-SCAN:** se comporta igual que el *SCAN* pero restringe la atención en un solo sentido. Al llegar a la última pista del disco en el sentido actual vuelve a la pista del otro extremo (**salto** → no se cuentan los movimientos) y sigue barriendo en el mismo sentido



Llega hasta la ultima pista del disco(0) y despues cambia la direccion (PEGANDO UN SALTO)..

Es igual que el SCAN, LLEGA HASTA EL 0, PEGA UN SALTO Y SIGUE BARRIENDO EN EL MISMO SENTIDO

Circular SCAN

Solo me importa la Queue, desde donde viene y desde donde empieza

Tp6 Algoritmos de planificacion

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

C-Scan (*Circular Scan*)

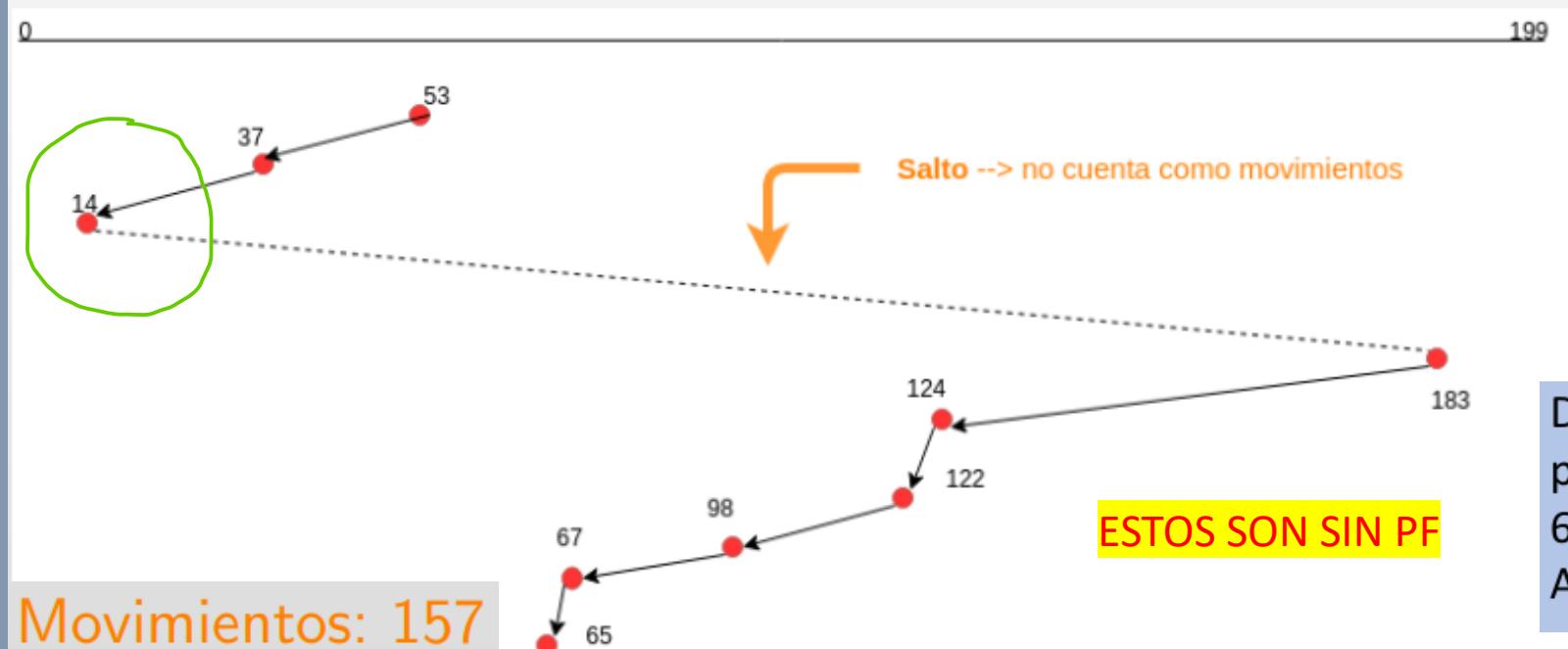
La cantidad de movimientos, misma logica del anterior..
Primero hacer igual el grafico y despues sacar las cantidades

Desde donde viene la pista, me sirve para determinar el sentido del barrido
61 es un Mayor.. Venia de mayores (DERECHA)
Ahora tocan los numeros menores (IZQUIERDA)

Ejemplos C-LOOK

Recorro por los valores menores, cuando **llego al ultimo valor menor**. Cambia de **rumbo**, para los valores mayores

- **C-LOOK:** se comporta igual que el *LOOK* pero restringe la atención en un solo sentido. Al llegar a la última pista de los requerimientos en el sentido actual vuelve a la primer pista más lejana del otro extremo (**salto** → no se cuentan los movimientos) y sigue barriendo en el mismo sentido



Tp6 Algoritmos de planificación

- Cantidad de pistas: 200 (0..199)
- Requerimientos en la cola: {98 , 183 , 37, 122, 14, 124, 65, 67}
- Viene de: pista 61
- Ubicación actual del cabezal: pista 53 → derecha-izquierda

C-Look (*Circular Look*)

La cantidad de movimientos, misma logica del anterior..
Primero hacer igual el grafico y despues sacar las cantidades

Desde donde viene la pista, me sirve para determiner el sentido del barrido
61 es un Mayor.. Venia de mayores (DERECHA)
Ahora tocan los numeros menores (IZQUIERDA)

13. El *Seek Time* es el parámetro que posee mayor influencia en el tiempo real necesario para transferir datos desde o hacia un disco. Es importante que el SO planifique los diferentes requerimientos que al disco para minimizar el movimiento de la cabeza lecto-grabadora.

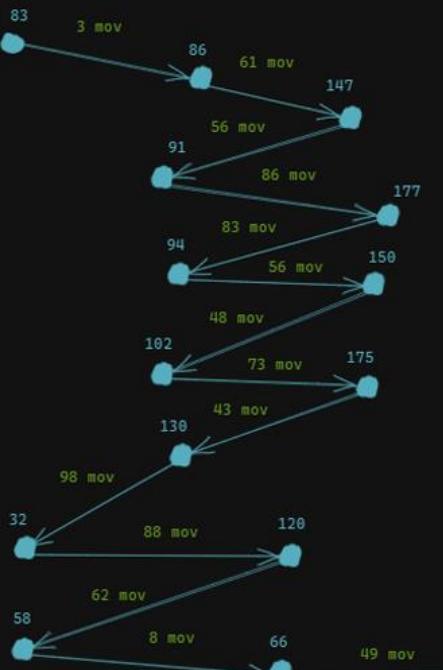
Analicemos las diferentes políticas de planificación de requerimientos a disco con un ejemplo: Supongamos un *Head* con movimiento en 200 tracks (numerados de 0 a 199), que está en el track 83 atendiendo un requerimiento y anteriormente atendió un requerimiento en el track 75.

Si la cola de requerimientos es: 86, 147, 91, 177, 94, 150, 102, 175, 130, 32, 120, 58, 66, 115. Realice los diagramas para calcular el total de movimientos de head para satisfacer estos requerimientos de acuerdo a los siguientes algoritmos de scheduling de discos:

- (a) FCFS (*First Come, First Served*)
- (b) SSTF (*Shortest Seek Time First*)
- (c) Scan
- (d) Look
- (e) C-Scan (*Circular Scan*)
- (f) C-Look (*Circular Look*)

Si no dice nada se asume que viene de derecha a izquierda???

ESTOS SON SIN PF



FCFS (FIRST COME, FIRST SERVED)

Anteriormente estaba en el track 73

Actualmente esta en el track 83

SIC SAC

- 1) Hacemos el dibujo
- 2) Calculamos los movimientos

limite superior(donde llega la flecha) - limite inferior(desde donde sale la flecha)

$$86-83 = 3$$

$$147-86 = 61$$

$$|91-147| = 56$$

$$|177-91| = 86$$

$$|94-177| = 83$$

$$|150-94| = 56$$

$$|102-150| = 48$$

$$|175-102| = 73$$

$$|130-175| = 45$$

$$|32-130| = 98$$

$$|120-32| = 88$$

$$|58-120| = 62$$

$$|66-58| = 8$$

$$|115-66| = 49$$

SUMA DE TODO = 816 MOVIMIENTOS

Tp6 Algoritmos de planificación

EJEMPLO

tEORIA

La inanición de requerimientos (también conocida como starvation) ocurre cuando un algoritmo no atiende ciertos procesos o solicitudes porque prioriza constantemente a otros, lo que deja a algunos procesos esperando indefinidamente. Esto es un problema en la planificación de recursos en sistemas operativos, bases de datos y redes.

Ejemplo en sistemas operativos:

En un planificador de CPU, la inanición puede suceder cuando los procesos de alta prioridad o los procesos más cortos son atendidos continuamente, mientras que los de baja prioridad quedan relegados.

Escenario:

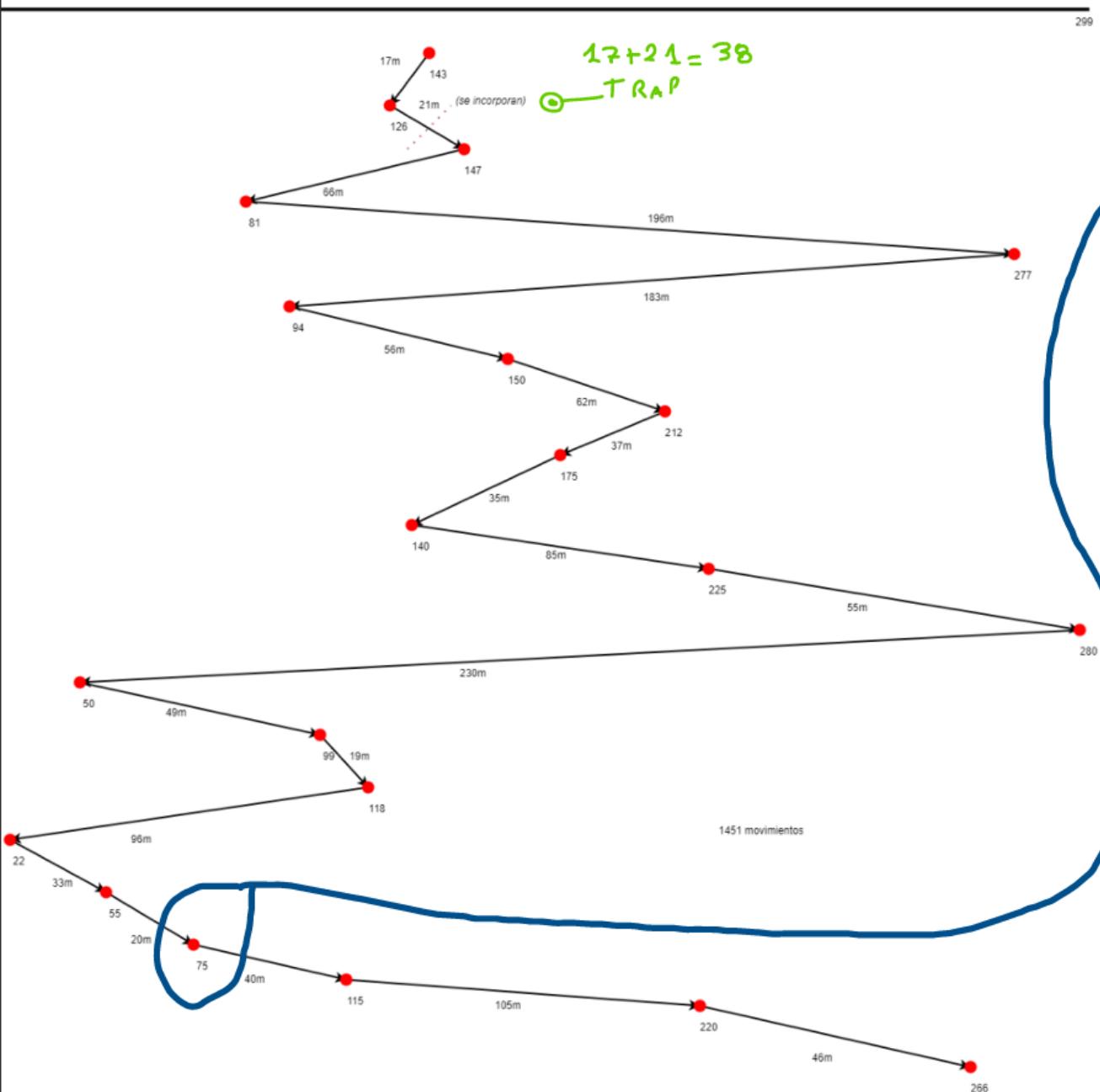
- Un algoritmo de planificación como **SJF** (Shortest Job First) siempre selecciona el proceso más corto.
- Si llegan continuamente procesos más cortos, un proceso largo puede quedarse esperando indefinidamente, ya que nunca es seleccionado.

Soluciones a la inanición:

1. **Algoritmo de envejecimiento (aging):** Incrementa la prioridad de los procesos que han estado esperando por mucho tiempo, para garantizar que eventualmente sean atendidos.
2. **Justicia en la asignación:** Usar técnicas como **round-robin** o límites en el tiempo de ejecución para cada proceso.
3. **Límite en la espera:** Definir un tiempo máximo de espera antes de que un proceso sea forzadamente atendido.

Otra variante de la practica

(a) FCFS



Otra variante que puede aparecer, misma logica q arriba, solo se agrega un trap de activacion de otra Queue despues de X movimientos “Se incorpora”

Lo que hace es tirarlo al final de la primera Queue

15. Supongamos un Head con movimiento en 300 pistas (numerados de 0 a 299), que esta en la pista 143 atendiendo un requerimiento y anteriormente atendió un requerimiento en la pista 125.

Si la cola de requerimientos es: 126, 147, 81, 277, 94, 150, 212, 175, 140, 225, 280, 50, 99, 118, 22, 55; y después de 30 movimientos se incorporan los requerimientos de las pistas 75, 115, 220 y 266. Realice los diagramas para calcular el total de movimientos de head para satisfacer estos requerimientos de acuerdo a los siguientes algoritmos de scheduling de discos:

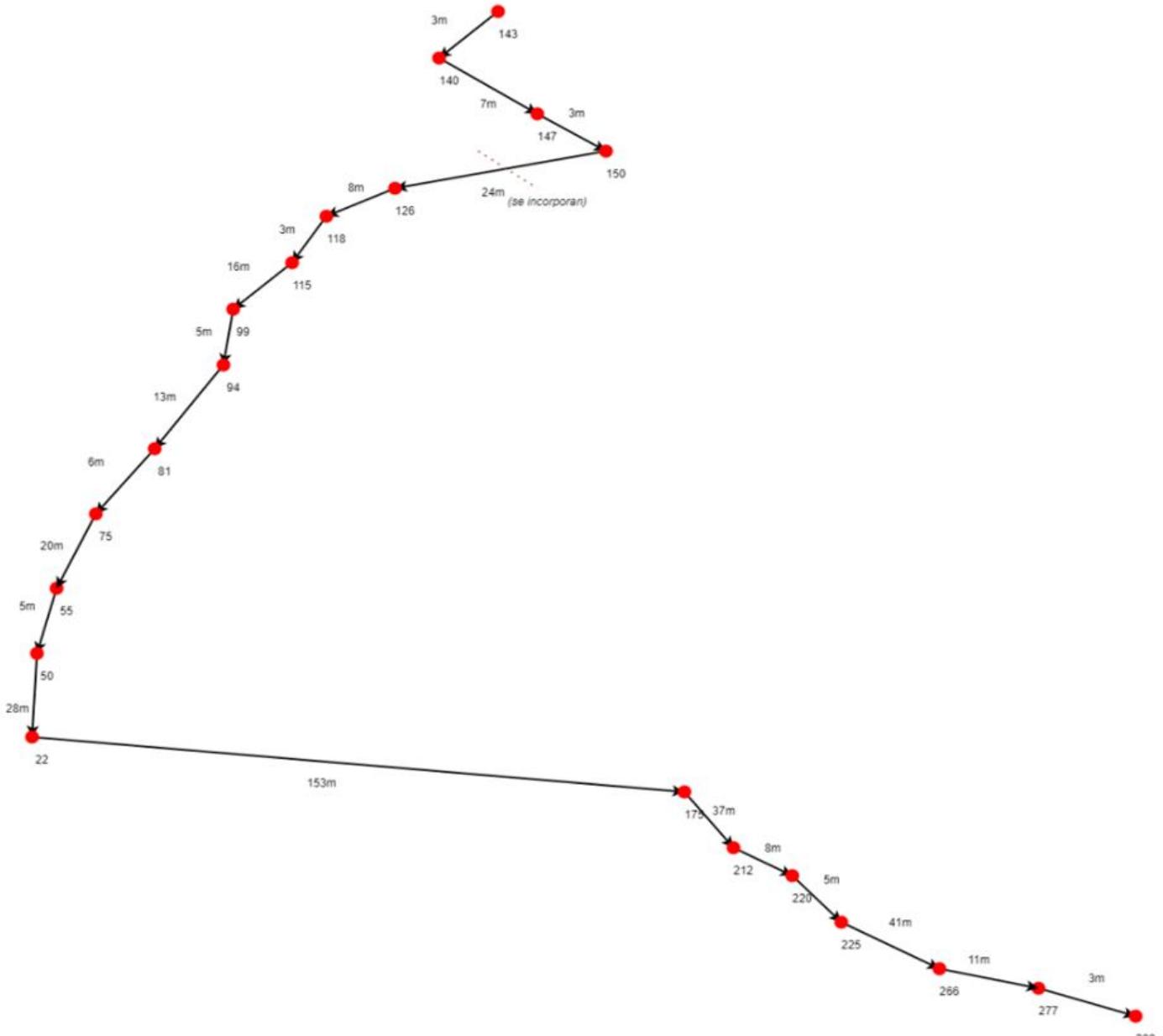
ESTOS SON SIN PF

Otra variante de la practica

Lo mismo q arriba, en X punto se encola la nueva cola
Al final y agrega mas numeros a la bolsa para elegir
los numeros menores o ir por los numeros mayores

ESTOS SON SIN PF

(b) SSTF



Algoritmos–Atencion de Page Fould

Algoritmos - Ejemplo de enunciado con page faults

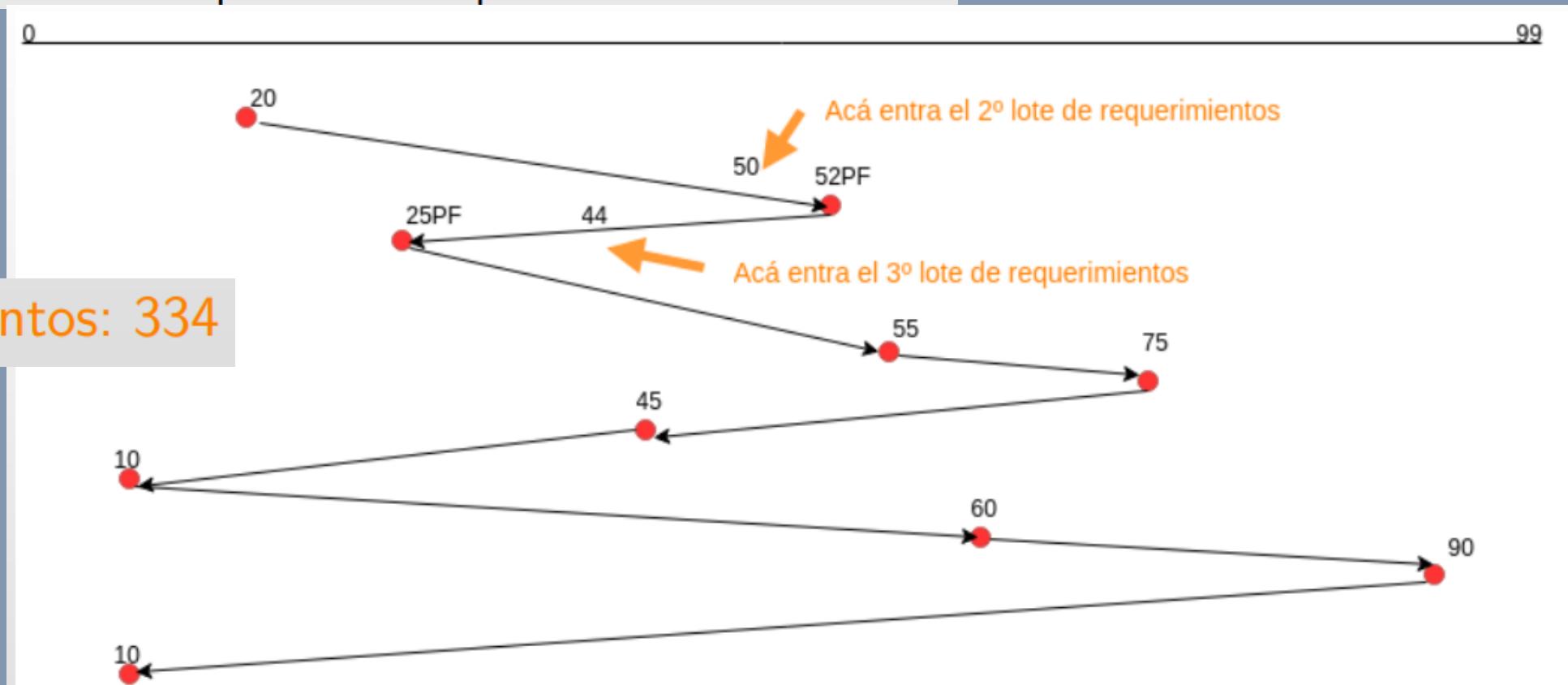
- Existen requerimientos especiales que deben atenderse con urgencia. Los *fallos de página* indican simplemente que tienen mayor prioridad con respecto a los requerimientos convencionales, por lo tanto deben ser atendidos inmediatamente después del requerimiento que se está atendiendo actualmente
- La lógica de atención de múltiples *PF* se maneja según el algoritmo de planificación. Ejemplos:
 - **FCFS:** Si tengo {10, 40PF, 70PF, 10}, primero se atiende al 40PF y luego al 70PF luego sigo desde donde me había quedado
 - **SSTF:** si tengo {10, 40PF, 70PF, 10} y estoy en la pista 65, primero atiendo al 70PF y luego al 40PF Sigue la logica del mas cercano
- En todos los algoritmos, los movimientos utilizados para atender estos requerimientos especiales deben ser contados

- Una vez que no existan más requerimientos por *page faults* en la cola, se procede:
 - **FCFS:** en orden *FCFS*
 - **SSTF:** en orden *SSTF*
 - **SCAN:** con el sentido que determina la atención de los últimos dos requerimientos → puede cambiar de sentido
 - **C-SCAN:** con el sentido original → el sentido no cambia
 - **LOOK:** del mismo modo en que lo hace el *SCAN*
 - **C-LOOK:** del mismo modo en que lo hace el *C-SCAN*

Algoritmos–Atencion de Page Fould

- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: $\{55, 75, 52^{PF}, 45, 10\}$. Luego de 30 movimientos $\{25^{PF}, 60\}$ y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra $\{90, 10\}$
- Se viene de la pista 15
- Se está atendiendo la pista 20 \rightarrow izquierda-derecha

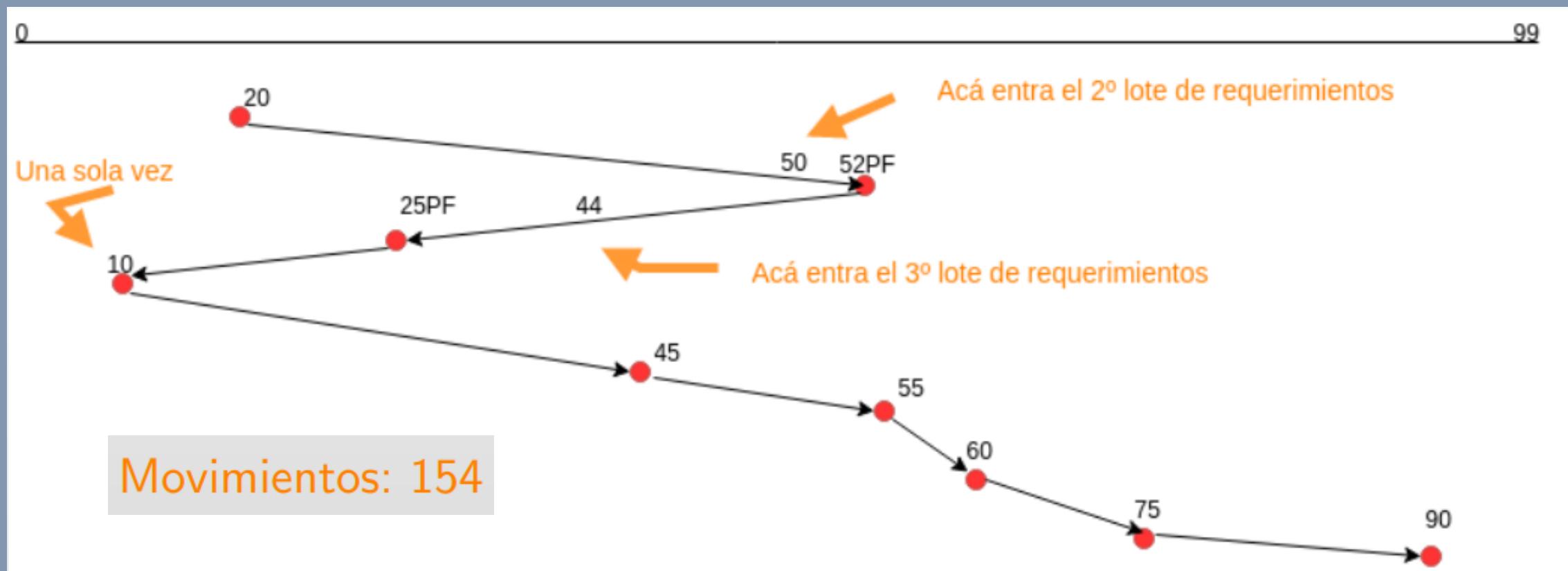
First Come First Served



Algoritmos–Atencion de Page Fould

- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: {55 , 75 , 52^{PF} , 45, 10}. Luego de 30 movimientos {25^{PF} , 60} y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra {90, 10}
- Se viene de la pista 15
- Se está atendiendo la pista 20 → izquierda-derecha

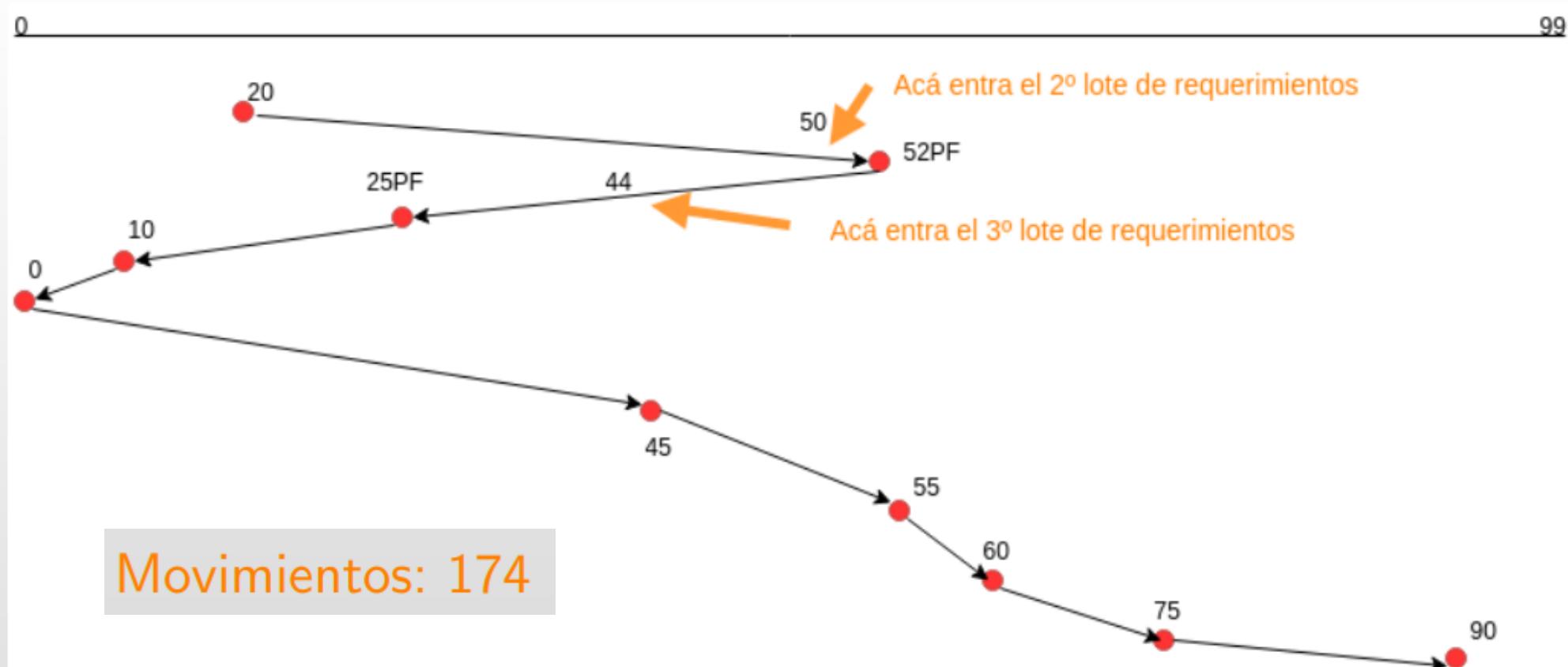
Sortest Seek Time First



Algoritmos–Atencion de Page Fould

SCAN

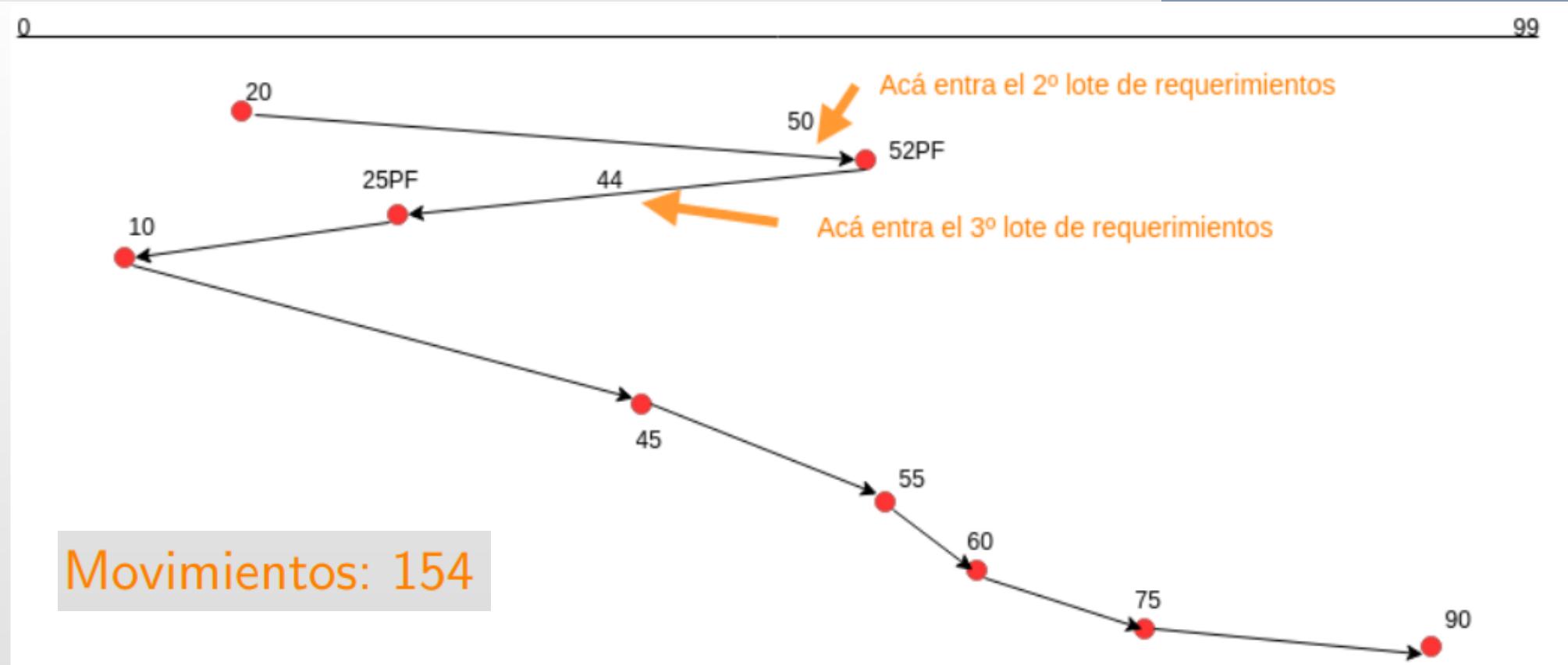
- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: $\{55, 75, 52^{PF}, 45, 10\}$. Luego de 30 movimientos $\{25^{PF}, 60\}$ y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra $\{90, 10\}$
- Se viene de la pista 15
- Se está atendiendo la pista 20 \rightarrow izquierda-derecha



Algoritmos–Atencion de Page Fould

- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: $\{55, 75, 52^{PF}, 45, 10\}$. Luego de 30 movimientos $\{25^{PF}, 60\}$ y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra $\{90, 10\}$
- Se viene de la pista 15
- Se está atendiendo la pista 20 \rightarrow izquierda-derecha

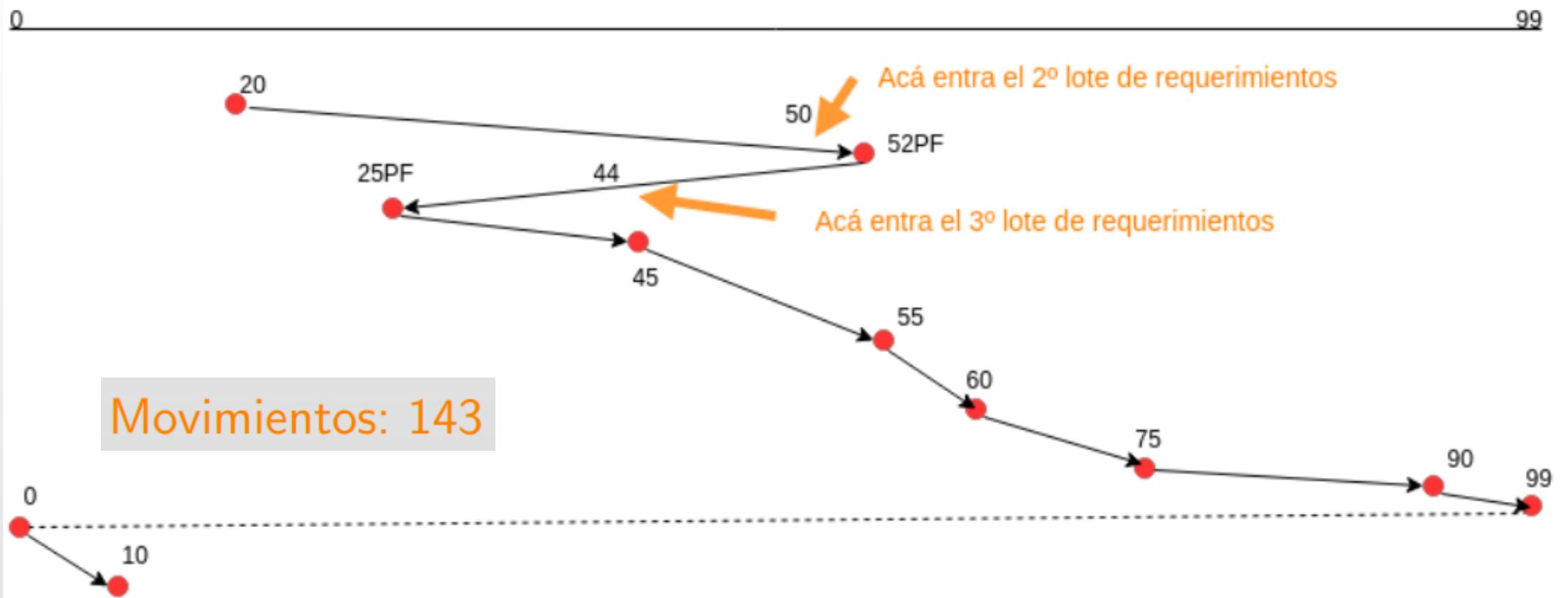
LOOK



Algoritmos–Atencion de Page Fould

- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: $\{55, 75, 52^{PF}, 45, 10\}$. Luego de 30 movimientos $\{25^{PF}, 60\}$ y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra $\{90, 10\}$
- Se viene de la pista 15
- Se está atendiendo la pista 20 \rightarrow izquierda-derecha

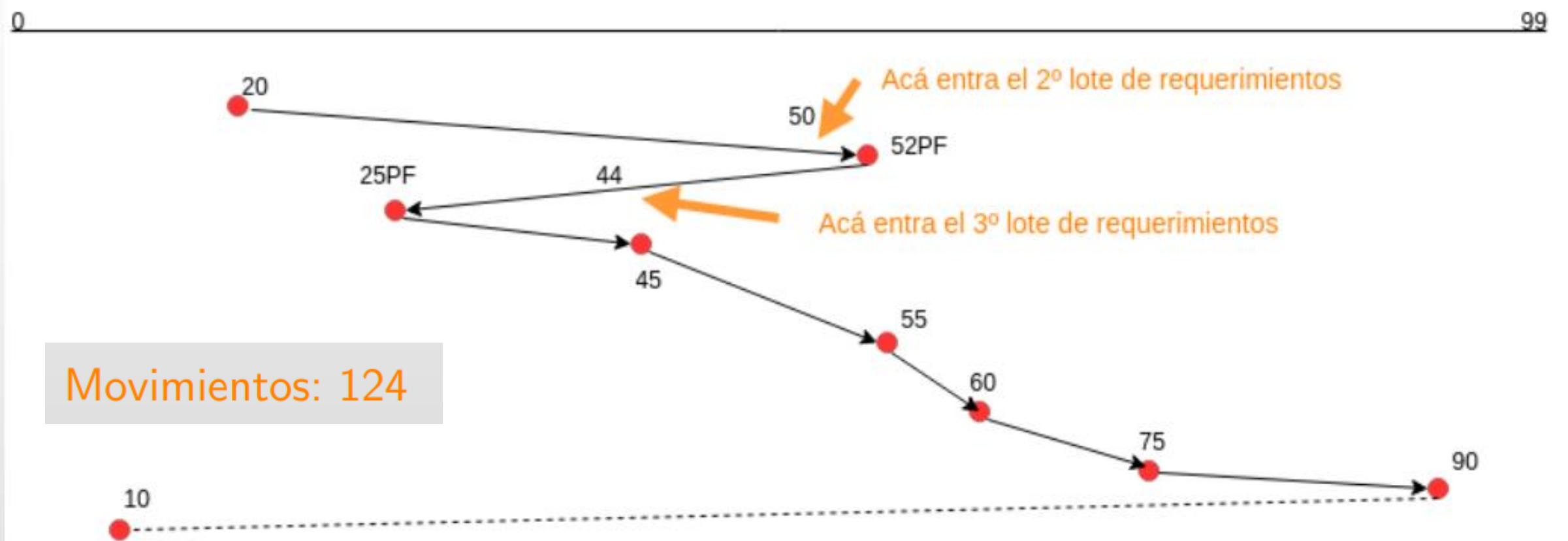
Circular SCAN



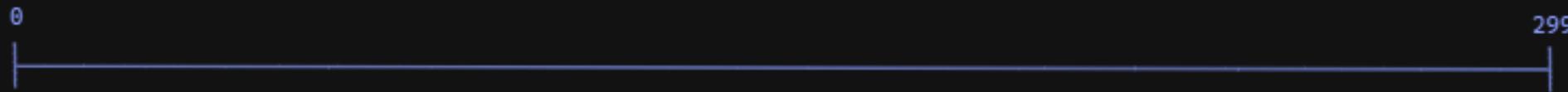
Algoritmos–Atencion de Page Fould

- Cantidad de pistas: 100 (0..99)
- Requerimientos en la cola: $\{55, 75, 52^{PF}, 45, 10\}$. Luego de 30 movimientos $\{25^{PF}, 60\}$ y luego de 10 movimientos más (40 desde el comienzo de la planificación) entra $\{90, 10\}$
- Se viene de la pista 15
- Se está atendiendo la pista 20 \rightarrow izquierda-derecha

Circular LOOK



Ejemplardo, con combinada



Viene de la pista 135

Pista Actual 140

{99, 110, 42, 25, 186, 270, 50, 99, 147PF, 81, 257, 94, 133, 212, 175, 130 } Lote 1 de requerimientos

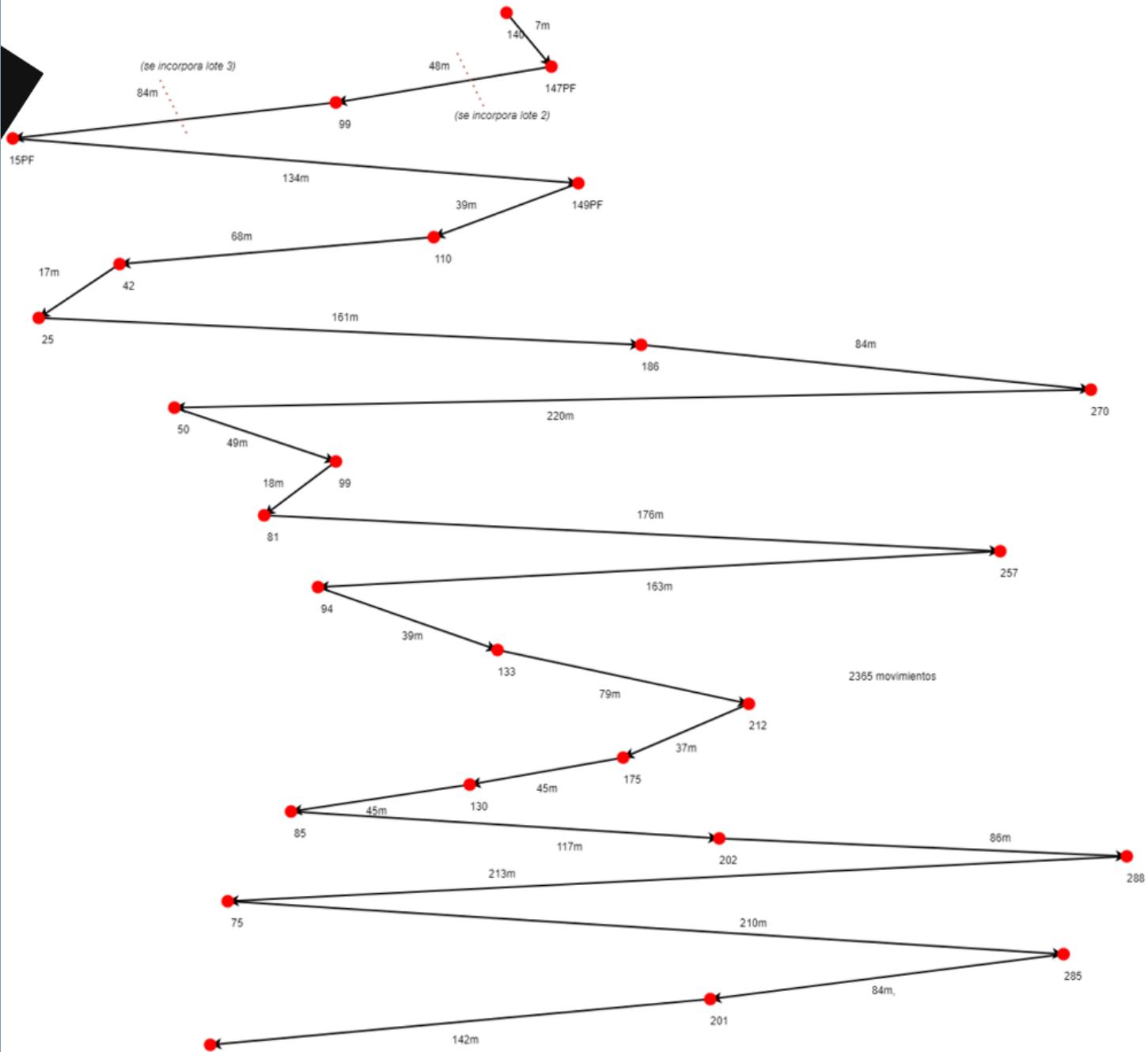
Despues de 30 movimientos {85, 15PF, 202, 288 } Lote 2 de requerimientos

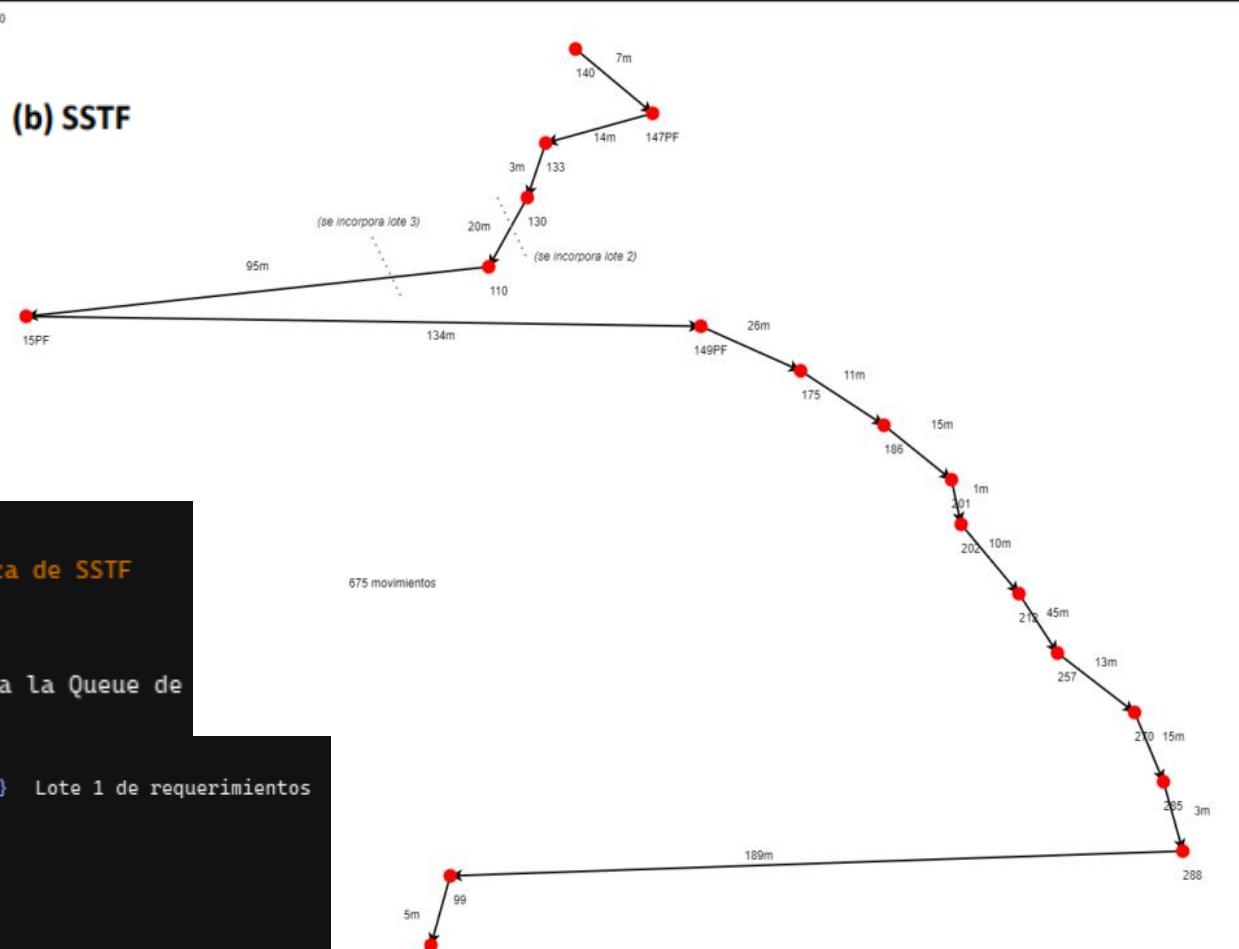
Despues de 40 movimientos {75, 149PF, 285, 201, 59 } Lote 3 de requerimientos

Fifo normalon, con page faults.. sigo al norte y vuelvo cuando se terminaron las page faults

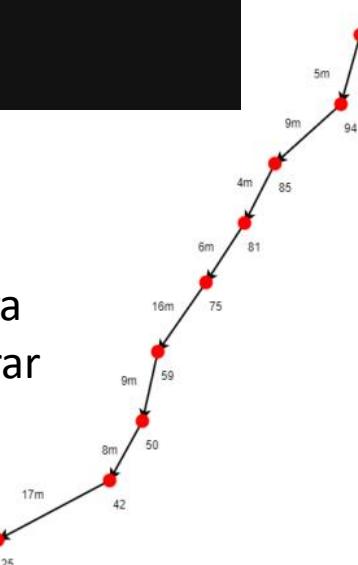
Ejemplardo, con combinada

(a) FCFS

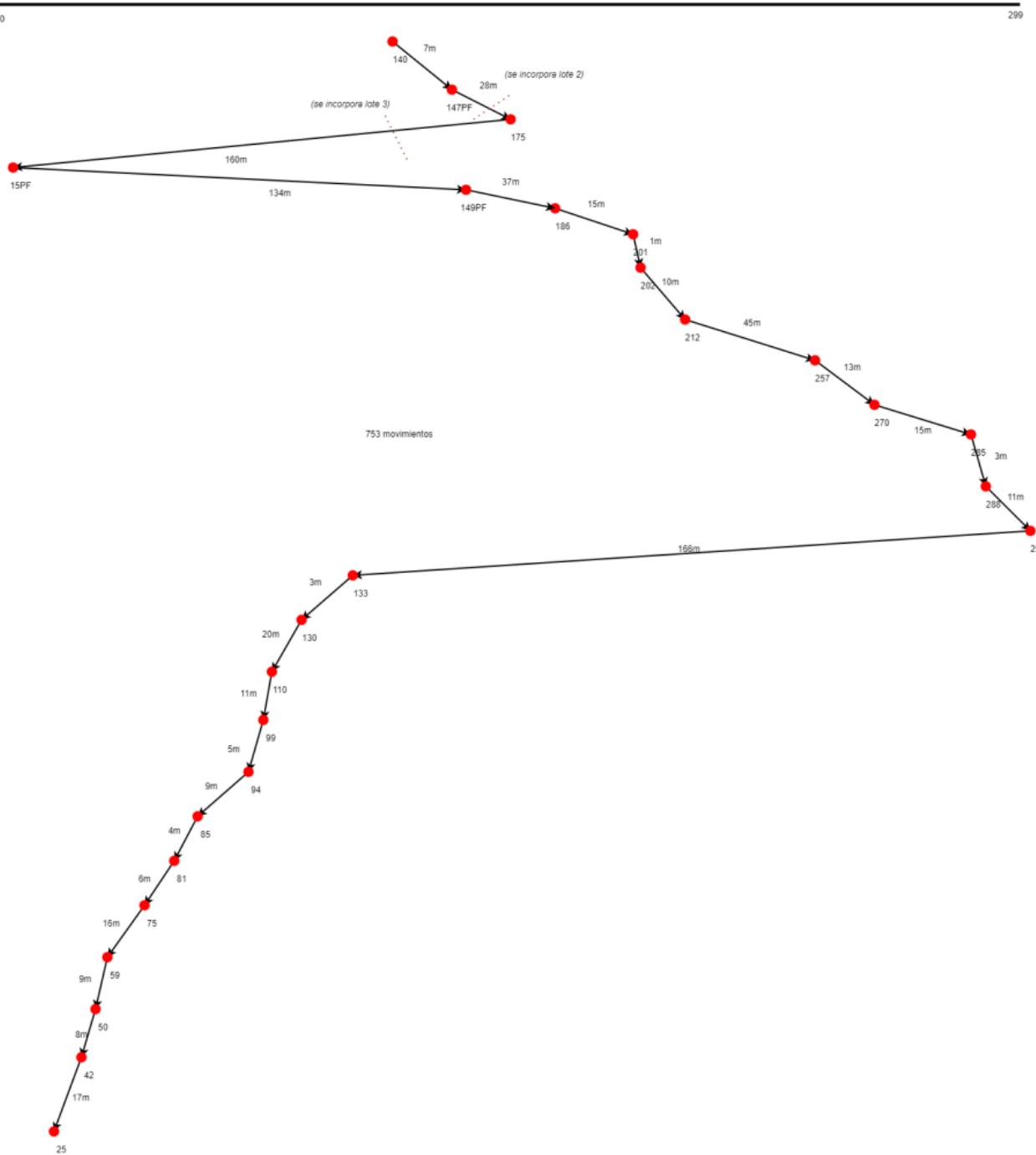




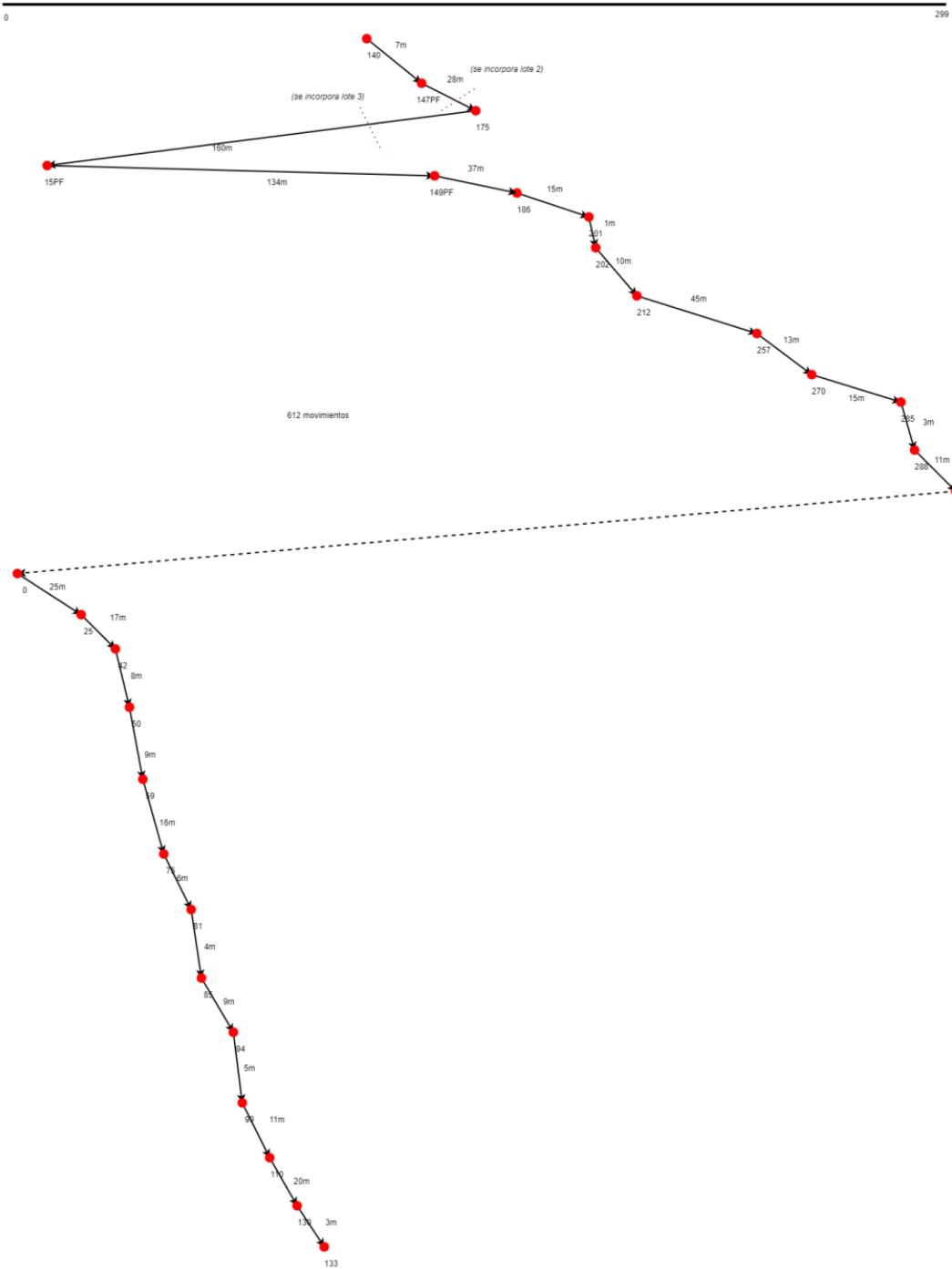
Las incorporaciones, son secuenciales
Si llego primero un valor mayor a 30 de movimientos, primero entra el de 30 movimientos y despues si hay otro q tambien deberia entrar entra en la proxima Secuencia

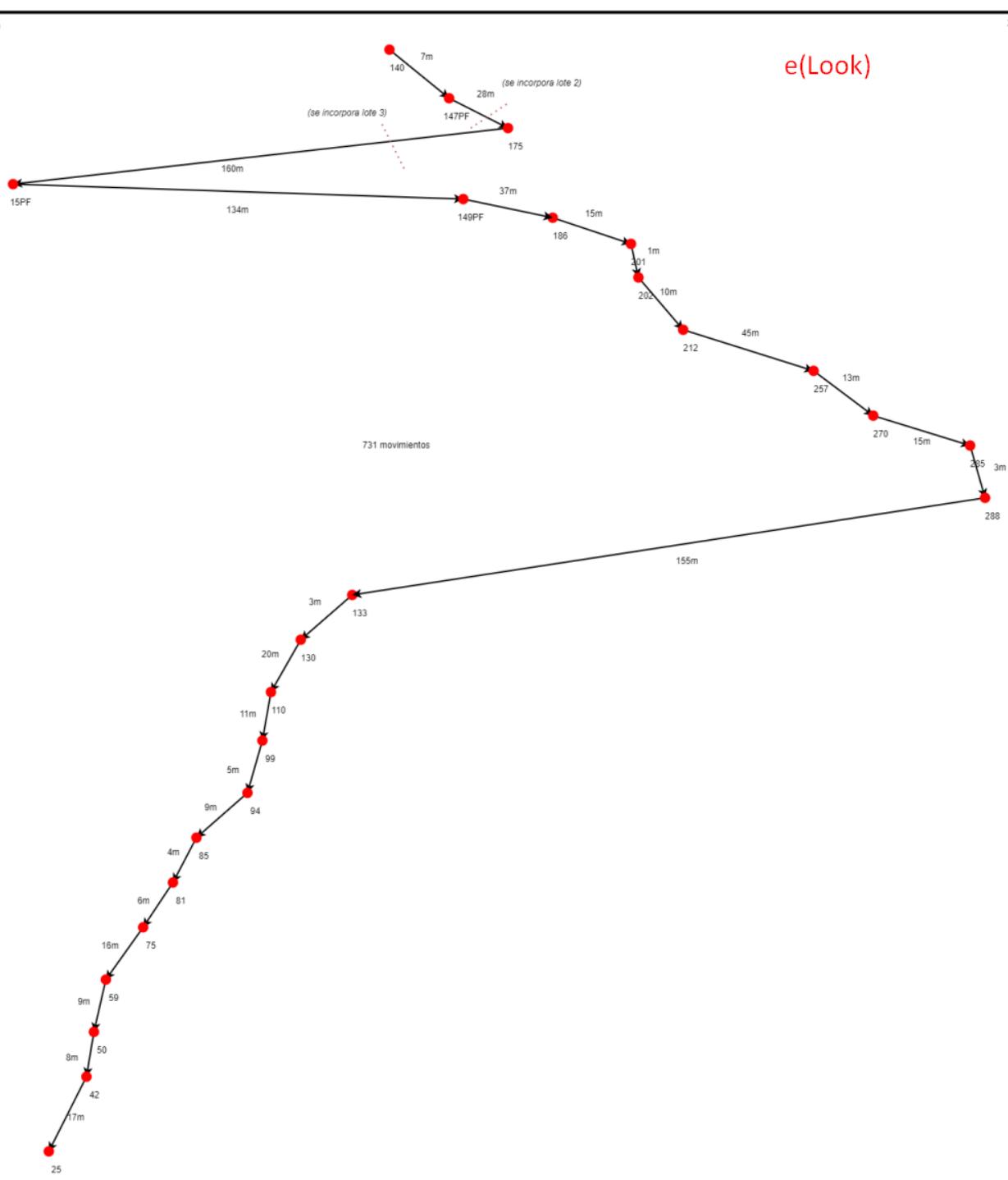


(d) Scan

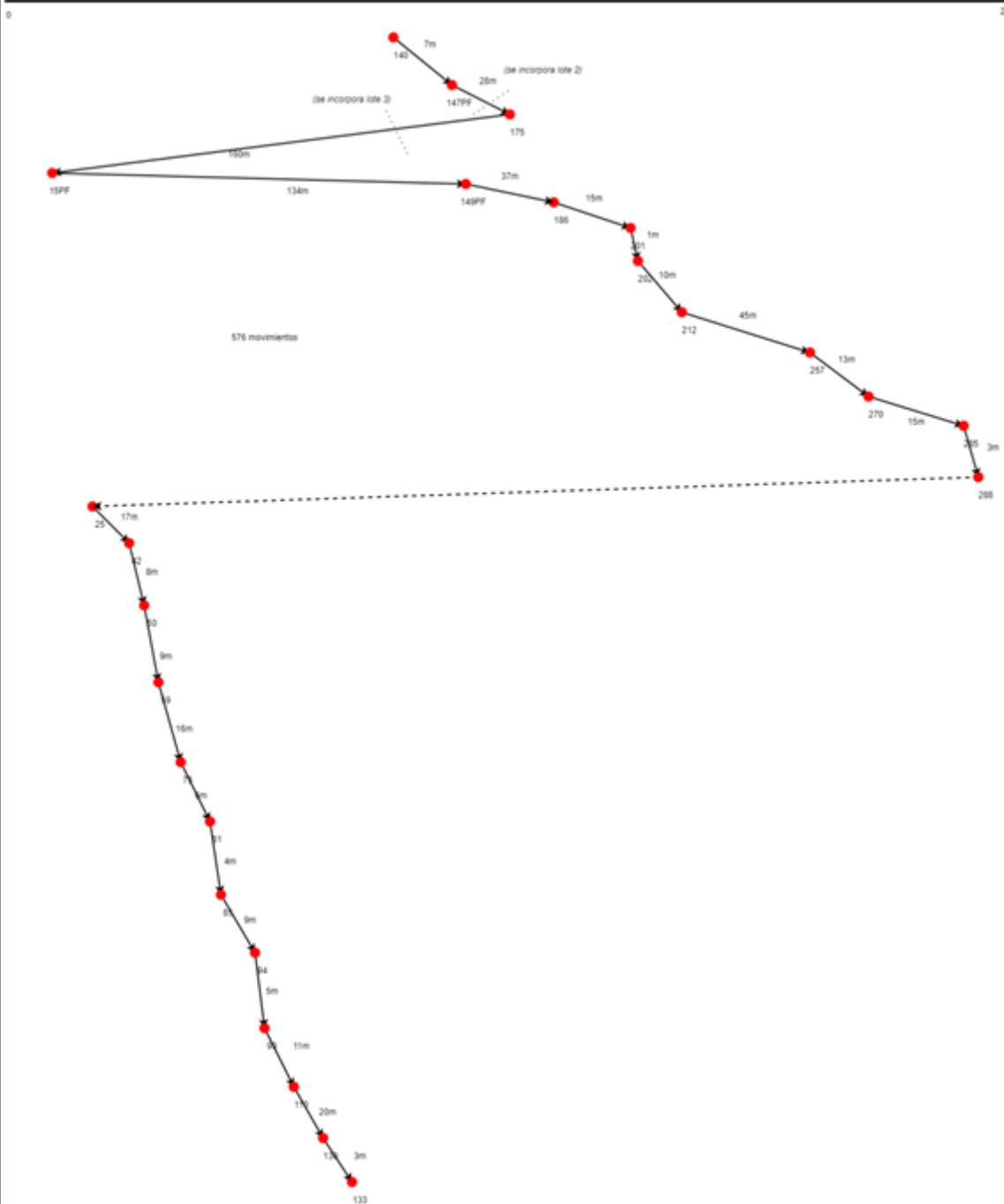


(c) C-Scan





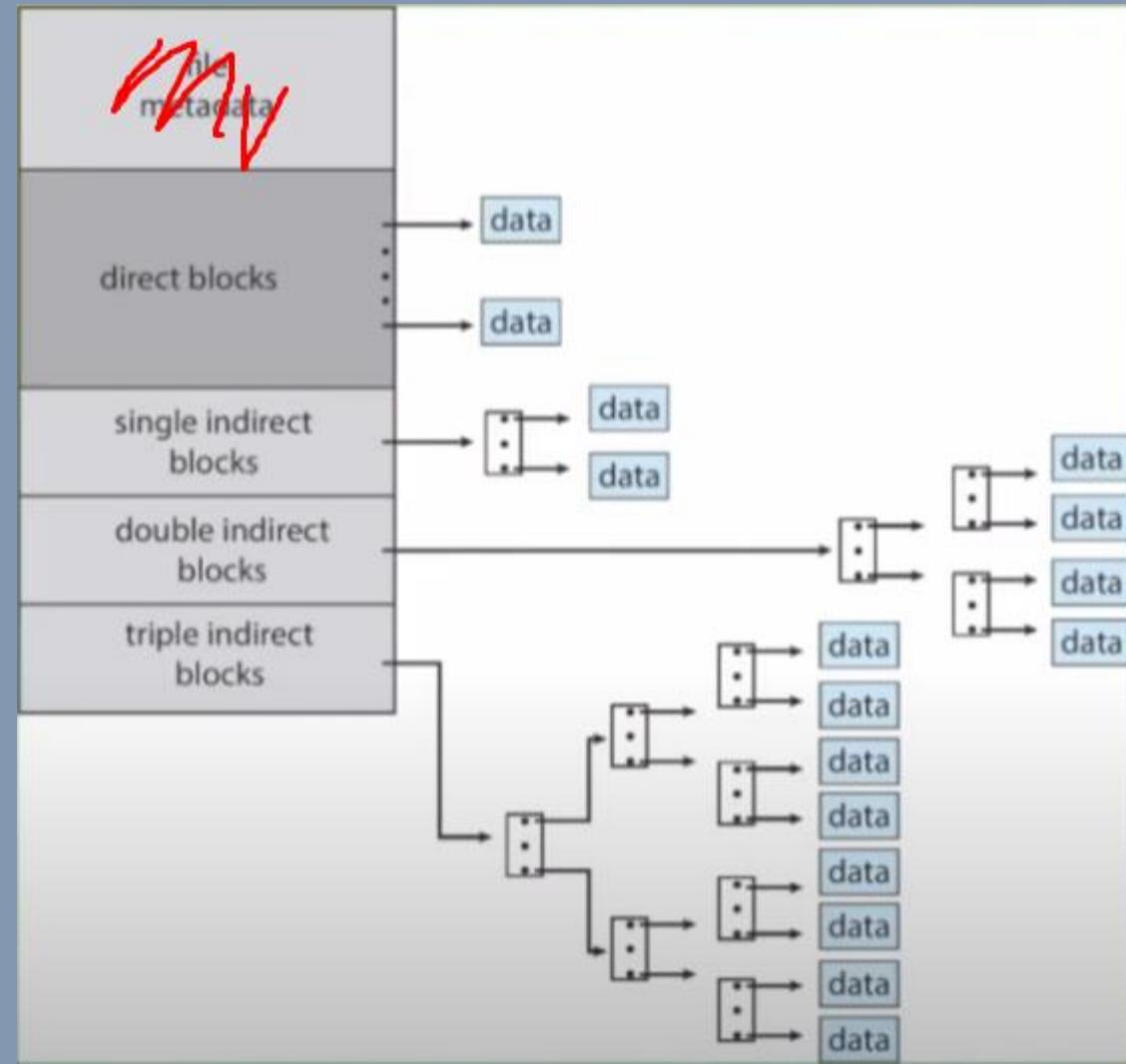
(f) C-Look



Inodos

Estructura auxiliar que permite, en la mayoría de los sistemas de archivos de sistemas operativos *nix, referenciar los archivos y acceder a ellos.

- Información de bajo nivel sobre archivos (regulares, directorios, enlaces).
- Se identifica con un número.
- Contiene:
 - Metainformación del archivo.
 - Punteros a los bloques de datos en el disco que conforman el archivo.



$$1KB = 1024 \text{ Bytes} = (1024 \times 8 = 8192) = 8192 \text{ Bits}$$

1 DIRECCION PARA REFERENCIAR UN BLOQUE DE 32 BITS
ATA = UN BLOQUE PUEDE TENER 256 REFERENCIAS

$$I = \frac{8192}{32} = 256 \quad \text{ATA = UN BLOQUE PUEDE TENER 256 REFERENCIAS}$$

II = TAMAÑO MAXIMO DE UN ARCHIVO

$$10) 10 \text{ BLOQUES} \times 1024 \text{ Bytes/Bloque} = 10240 \text{ Bytes} = 10 \text{ KiB}$$

$$11) 256 \text{ REFERENCIAS A BLOQUES} \times 1024 \text{ Bytes QUE PESA UN BLOQUE} = \frac{262,144}{1024} = 256 \text{ KiB}$$

$$12) (256 \text{ DIRECCIONES} \times (\underbrace{256 \text{ REFERENCIAS A BLOQUES} \times 1024 \text{ Bytes QUE PESA UN BLOQUE}}_{\text{BLOQUE INDIRECTO SIMPLE}})) = 67,108,864 = 65,536 \text{ KiB}$$

$$13) (256 \text{ DIRECCIONES} \times (65,536 \text{ KiB} \times 1024 \text{ Bytes QUE PESA UN BLOQUE})) = 16,777,216 \text{ KiB}$$

FORMULAS PARA LOCALIZAR UN iNODO

$$\text{NRO BLOQUE} = ((\text{NRO DE iNODO} - 1) / \text{NRO DE iNODOS POR BLOQUE}) + \text{BLOQUE DE COMIENZO DE LA LISTA DE iNODOS}$$

$$A) ((8-1)/8) + 2 = (7/8) + 2 = 0.875 + 2 = 0 + 2 = 2 \quad \therefore \quad \text{EL iNODO 8 SE ENCUENTRA EN EL BLOQUE 2}$$

$$(9-1)/8 + 2 = (8/8) + 2 = 1 + 2 = 3 \quad \therefore \quad \text{EL iNODO 9 SE ENCUENTRA EN EL BLOQUE 3}$$

CON 16 iNODOS

$$(8-1)/16 + 2 = (7/16) + 2 = 0.4375 + 2 = 0 + 2 = 2 \quad \therefore \quad \text{SE ENCUENTRA EN EL BLOQUE 2}$$

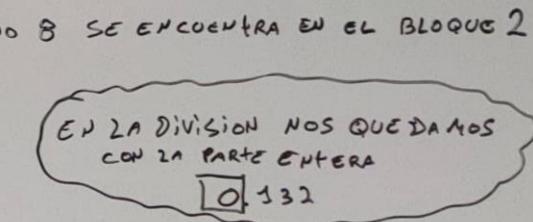
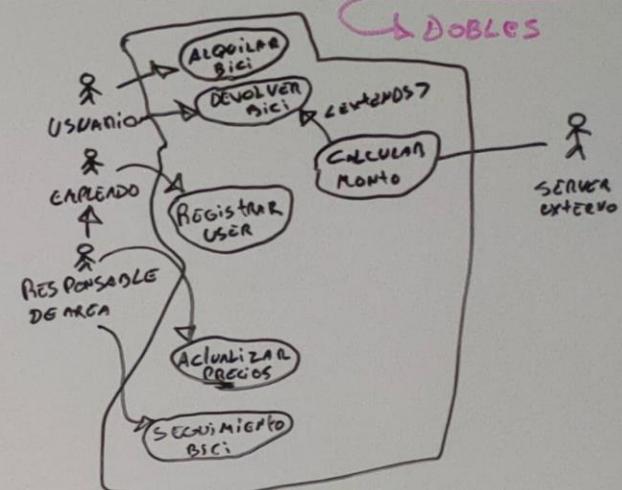
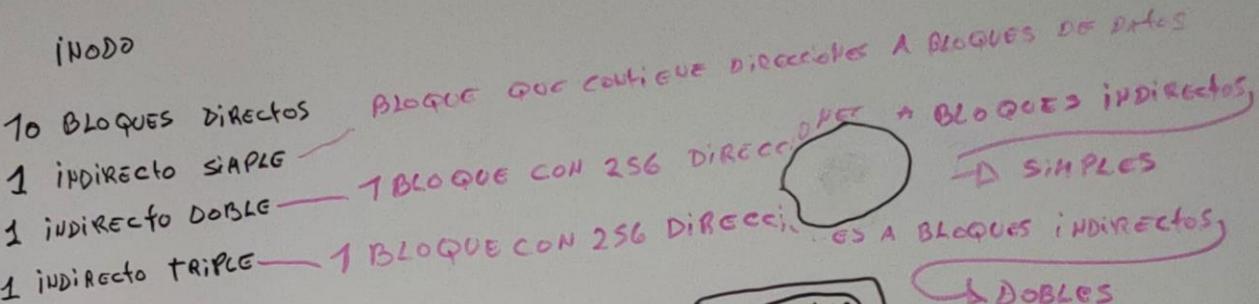
$$(9-1)/16 + 2 = (8/16) + 2 = 0.5 + 2 = 0 + 2 = 2 \quad \therefore \quad \text{SE ENCUENTRA EN EL BLOQUE 2}$$

$$B) \text{DESPLAZAMIENTO DEL iNODO EN EL BLOQUE} = ((\text{NRO DE iNODO} - 1) \text{ MOD } (\text{NUMERO DE iNODOS POR BLOQUE})) \times \text{MEDIDA DE iNODO DEL DISCO}$$

$$\text{¿DÓNDE EMPIEZA EL 6? } ((6-1) \text{ MOD } 8) \times 64 \text{ Bytes} = (5 \text{ MOD } 8) \times 64 = 5 \times 64 = 320 \quad \therefore \quad \text{EL iNODO 6 COMIENZA EN EL DESPLAZAMIENTO 320 DEL BLOQUE DEL DISCO}$$

$$\text{¿DÓNDE EMPIEZA EL 8? } ((8-1) \text{ MOD } 24) \times 128 \text{ Bytes} = (7 \text{ MOD } 24) \times 128 = 7 \times 128 = 896 \quad \therefore \quad \text{EL iNODO 8 COMIENZA EN EL DESPLAZAMIENTO 896 DEL BLOQUE DEL DISCO}$$

iNODO



EL iNODO 8 CON 16 iNODOS X BLOQUE
TANTO EL iNODO 8 Y 9 SE ENCUENTRAN EN EL BLOQUE 2

EL iNODO 8 CON 16 iNODOS X BLOQUE
TANTO EL iNODO 8 Y 9 SE ENCUENTRAN EN EL BLOQUE 2