

Administración de E/S

1. Dispositivos.

(a) Los dispositivos, según la forma de transferir los datos, se pueden clasificar en 2 tipos:

- **Orientado a bloques:** estos dispositivos transmiten datos en bloques (paquetes) y por esa razón son usados a menudo para la transmisión paralela de datos utilizando el buffer de datos del S.O. Almacenan la información en bloques de tamaño fijo, cada uno con una dirección propia (permite leer, escribir o buscar un bloque sin dependencia de los demás).
 - Por ejemplo: Disco magnético.
- **Orientado a flujos (de caracteres):** utilizan transmisión en secuencia de datos sin usar buffer, transmitiendo un bit o un byte a la vez. Aceptan o entregan un flujo de caracteres sin considerar estructuras de bloques. No son direccionables y, por tanto, no permiten operaciones de búsqueda.
 - Por ejemplo: Impresora, terminales, cintas de papel, interfaz de redes.

Describe las diferencias entre ambos tipos.

(b) Cite ejemplos de dispositivos de ambos tipos.

(c) Enuncie las diferencias que existen entre los dispositivos de E/S y que el SO debe considerar.

- Heterogeneidad de dispositivos
- Características de los dispositivos
- Velocidad
- Nuevos tipos de dispositivos
- Diferentes formas de realizar E/S

2. Técnicas de E/S Describe como trabajan las siguientes técnicas de E/S

- **E/S programada:** El procesador envía un mandato de E/S, a petición de un proceso, a un módulo de E/S; a continuación, ese proceso realiza una espera activa hasta que se complete la operación antes de continuar.
- **E/S dirigida por interrupciones:** El procesador emite un mandato de E/S a petición de un proceso y continúa ejecutando las instrucciones siguientes, siendo interrumpido por el módulo de E/S cuando éste ha completado su trabajo. Las siguientes instrucciones pueden ser del mismo proceso, en el caso de que ese proceso no necesite esperar hasta que se complete la E/S. En caso contrario, se suspende el proceso en espera de la interrupción y se realiza otro trabajo.
- **DMA (Acceso Directo a Memoria):** Un módulo de DMA controla el intercambio de datos entre la memoria principal y un módulo de E/S. El procesador manda una petición de transferencia de un bloque de datos al módulo de DMA y resulta interrumpido sólo cuando se haya transferido el bloque completo.

	Sin interrupciones	Con interrupciones
Transferencia de E/S a memoria a través del procesador	E/S programada	E/S dirigida por interrupciones
Transferencia directa de E/S a memoria		Acceso directo a memoria (DMA)

3. La tecnica de E/S programa puede trabajar de dos formas:

- **E/S mapeada:** Los dispositivos y memoria comparten el espacio de direcciones. I/O es como escribir/leer en la memoria por lo que no hay instrucciones especiales para I/O (ya se dispone de muchas instrucciones para la memoria)
- **E/S aislada:** Hay un espacio separado de direcciones para I/O. Se necesitan líneas de I/O. Puertos de E/S. Instrucciones especiales (conjunto limitado)

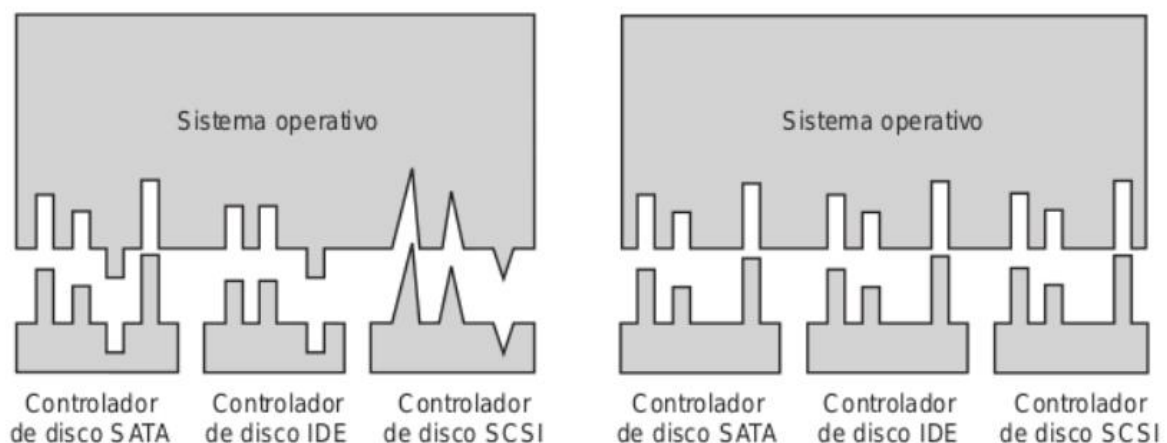
Indique como trabajan estas 2 técnicas.

4. Enuncie las metas que debe perseguir un SO para la administración de la entrada salida.

Generalidad:

- Es deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada.
- Ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” para que los procesos vean a los dispositivos, en términos de operaciones comunes como: read, write, open, close, lock, unlock.

Interfaz Uniforme:



Eficiencia:

- Los dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria y la CPU.
- El uso de la multi-programación permite que un procesos espere por la finalización de su I/O mientras que otro proceso se ejecuta.

Planificación:

- Organización de los requerimientos a los dispositivos
 - Ej: Planificación de requerimientos a disco para minimizar tiempos

5. Drivers

(a) ¿Qué son?

Consiste en una interfaz entre el SO y el HARDWARE cargadas como módulos en el espacio de memoria del kernel conteniendo el código dependiente del dispositivo para manejar el mismo y traducir requerimientos abstractos en los comandos para el dispositivo manejado.

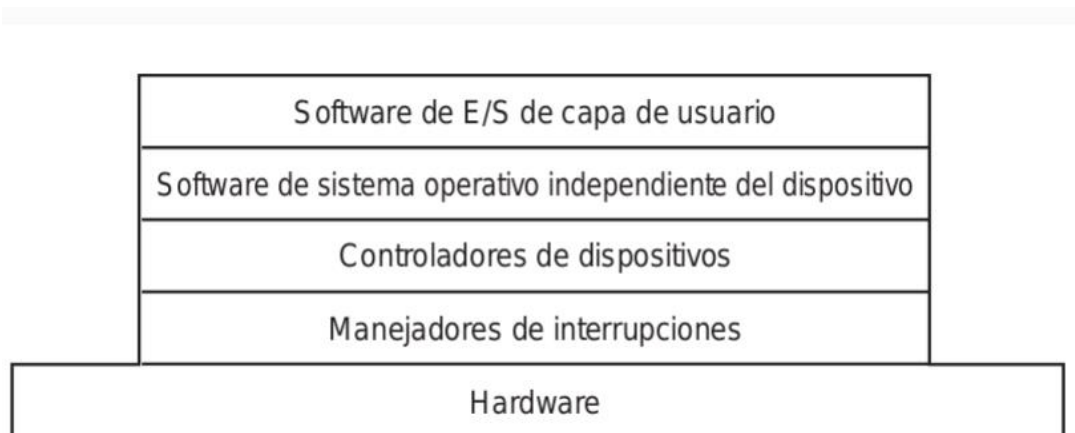
(b) ¿Qué funciones mínimas deben proveer?

- `init_module`: Para instalarlo
- `cleanup_module`: Para desinstalarlo.

(c) ¿Quién determina cuales deben ser estas funciones?

El fabricante

6. Realice un grafico que marque la relación entre el Subsistema de E/S, los drivers, los controladores de dispositivos y los dispositivos.



7. Describa mediante un ejemplo los pasos mínimos que se suceden desde que un proceso genera un requerimiento de E/S hasta que el mismo llega al dispositivo

Consideremos la lectura sobre un archivo en un disco:

- Determinar el dispositivo que almacena los datos
 - Traducir el nombre del archivo en la representación del dispositivo.
- Traducir requerimiento abstracto en bloques de disco (Filesystem)
- Realizar la lectura física de los datos (bloques) en la memoria
- Marcar los datos como disponibles al proceso que realizó el requerimiento
- Desbloquearlo

- Retornar el control al proceso

9. Enuncie que servicios provee el SO para la administración de E/S.

Buffering – Almacenamiento de los datos en memoria mientras se transfieren.

- Solucionar problemas de velocidad entre los dispositivos.
- Solucionar problemas de tamaño y/o forma de los datos entre los dispositivos.

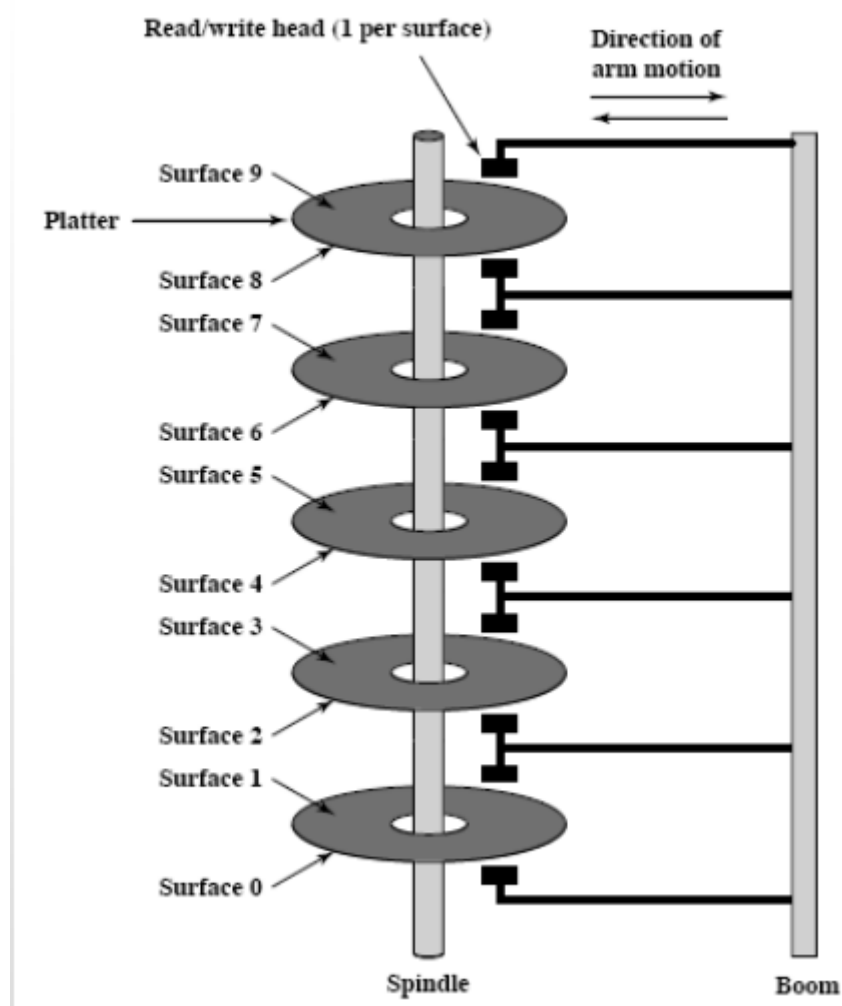
Caching – Mantener en memoria copia de los datos de reciente acceso para mejorar performance.

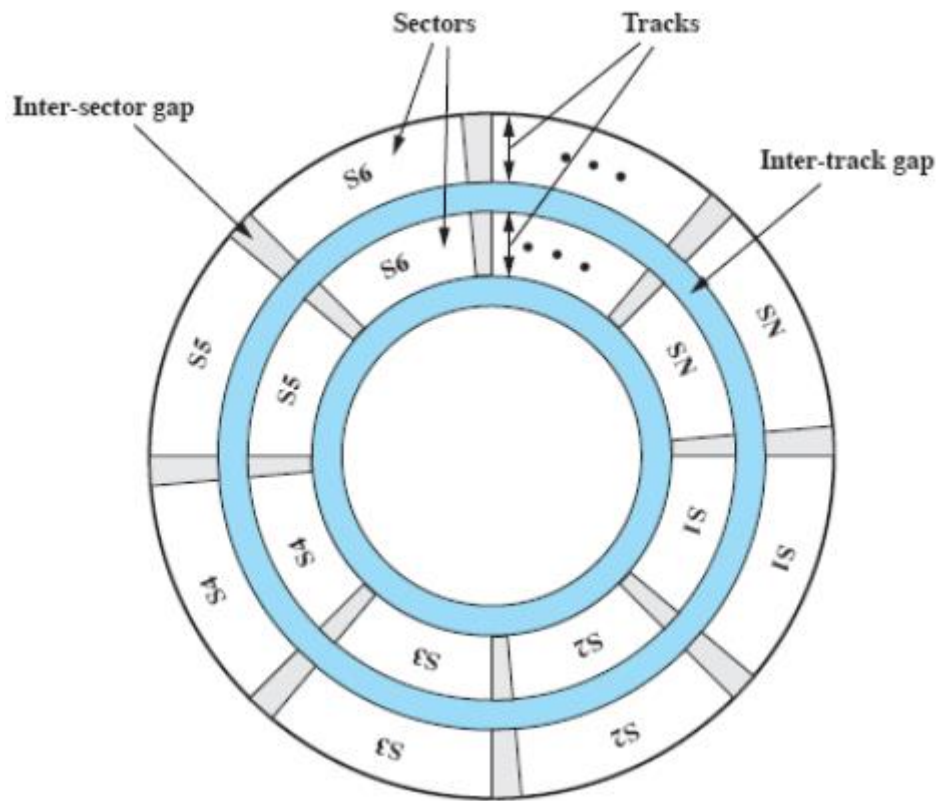
Spooling – Administrar la cola de requerimientos de un dispositivo.

- Algunos dispositivos de acceso exclusivo, no pueden atender distintos requerimientos al mismo tiempo: Por ej. Impresora.
- Spooling es un mecanismo para coordinar el acceso concurrente al dispositivo.

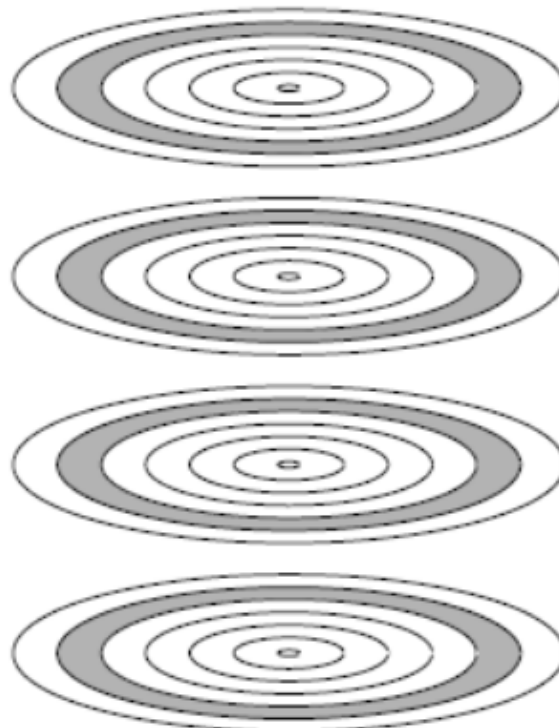
Administración de Discos

10. Describa en forma sintética, cómo es la organización física de un disco, puede utilizar gráficos para mayor claridad.





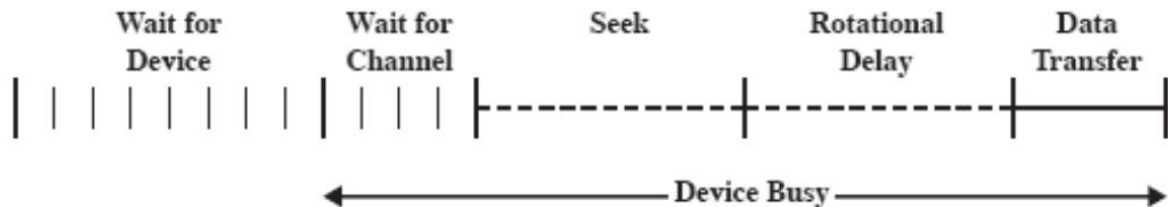
- Cilindro N: todas las n-esimas pistas de todas las caras



11. La velocidad promedio para la obtención de datos de un disco está dada por la suma de los siguientes tiempos:

- **Seek Time** (posicionamiento): tiempo que tarda en posicionarse la cabeza en el cilindro
- **Latency Time** (latencia): tiempo que sucede desde que la cabeza se posiciona en el cilindro hasta que el sector en cuestión pasa por debajo de la misma
- **Transfer Time** (transferencia): tiempo de transferencia del sector (bloque) del disco a la memoria

De una definición para estos tres tiempos.



12.

Suponga un disco con las siguientes características:

- 7 platos con
- 2 caras utilizables cada uno.
- 1100 cilindros
- 300 sectores por pista, donde cada sector de es 512 bytes.
- Seek Time de 10 ms
- 9000 RPM .
- Velocidad de Transferencia de 10 MiB/s (Mebibytes por segundos).

(a) Calcule la capacidad total del disco.

Tamaño_disco = $7 * 2 * 1100 * 300 * 512$ bytes

tamaño_disco= 2365440000 bytes = 2310000 KiB = 2255.859375 MiB= 2.2029876709 GiB

(b) ¿Cuántos sectores ocuparía un archivo de tamaño de 3 MiB(Mebibytes)?

- Capacidad de 1 sector:
512 bytes
- Dividimos el tamaño del archivo por la capacidad de una cara:
 $3 \text{ MiB} = 3145728 \text{ bytes}$
 $3145728 / 512 = 6144 \text{ sectores}$

(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)

Latencia:

9000 vueltas \rightarrow 60000 ms

0,5 \rightarrow 3,333 ms

Transferencia:

10 MiB \rightarrow 1000 ms

512 bytes \rightarrow x

Unificar unidades

10485760 bytes \rightarrow 1000 ms

512 bytes \rightarrow 0,048828125 ms

Bloques:

15728640 bytes / 512 bytes = 30720

Datos obtenidos:

Seek time: 10 ms

Latency time: 3,333 ms

Tiempo transferencia bloque: 0,048828125 ms

#bloques: 30720

Almacenamiento secuencial:

10 ms + 3,333 ms + (0,048828125 ms * 30720) = 1513,33 ms

(d) Calcule el tiempo de transferencia real de un archivo de 16 MiB(Mebibytes). grabado en el disco de manera aleatoria.

Latencia:

9000 vueltas \rightarrow 60000 ms

0,5 \rightarrow 3,333 ms

Transferencia:

10 MiB \rightarrow 1000 ms

512 bytes \rightarrow x

Unificar unidades

10485760 bytes \rightarrow 1000 ms

512 bytes \rightarrow 0,048828125 ms

Bloques:

$16777216 \text{ bytes} / 512 \text{ bytes} = 32768$

Datos obtenidos:

Seek time: 10 ms

Latency time: 3,333 ms

Tiempo transferencia bloque: 0,048828125 ms

#bloques: 32768

Almacenamiento aleatorio:

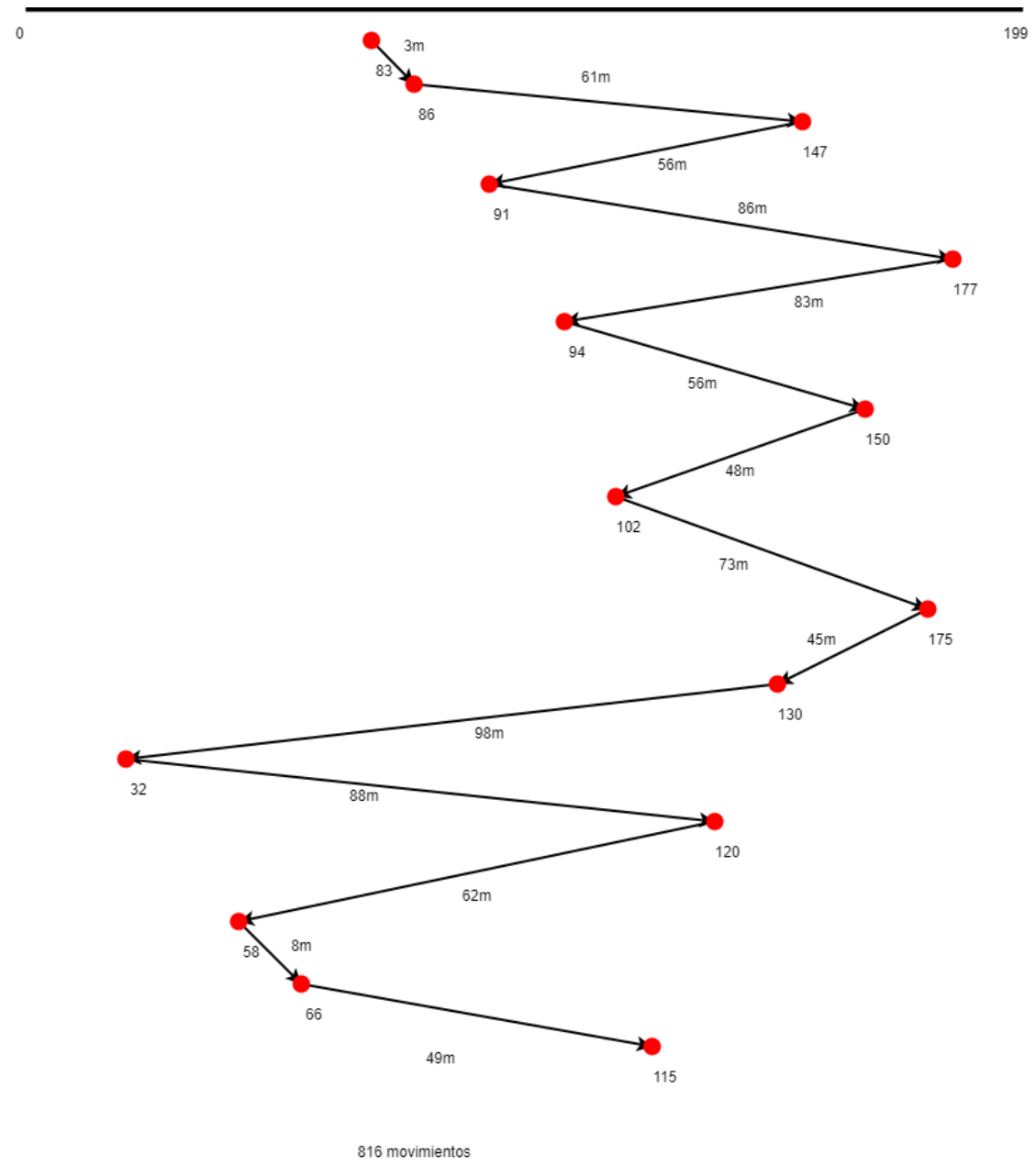
$(10 \text{ ms} + 3,333 \text{ ms} + 0,048828125 \text{ ms}) * 32768 = 438495,744 \text{ ms}$

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 lectora-grabadora.

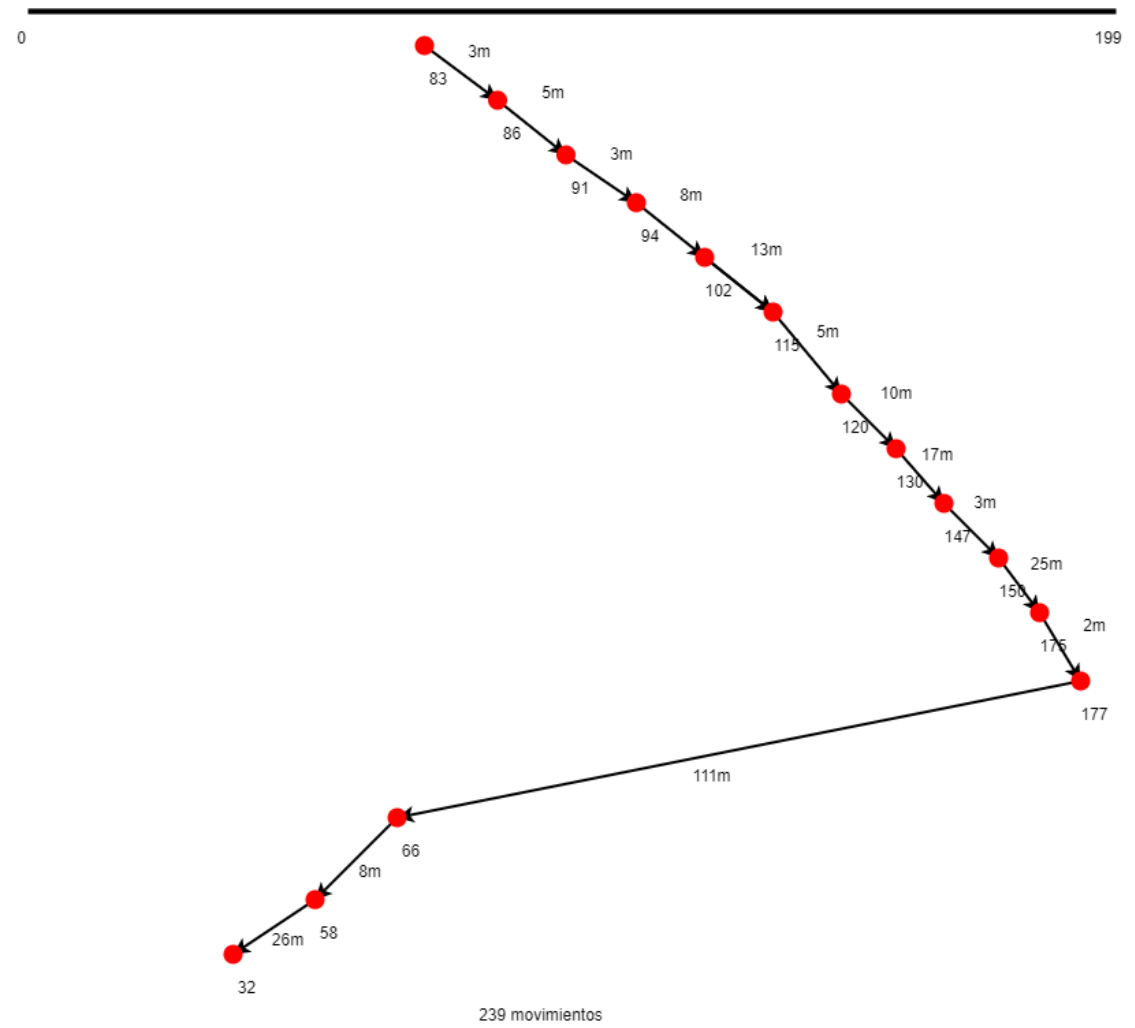
Analicemos las diferentes políticas de planificación de requerimientos a disco con un ejemplo: Supongamos un Head con movimiento en 200 tracks (numerados de 0 a 199), que está en el track 83 atendiendo un requerimiento y anteriormente atendió un requerimiento en el track 75.

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:

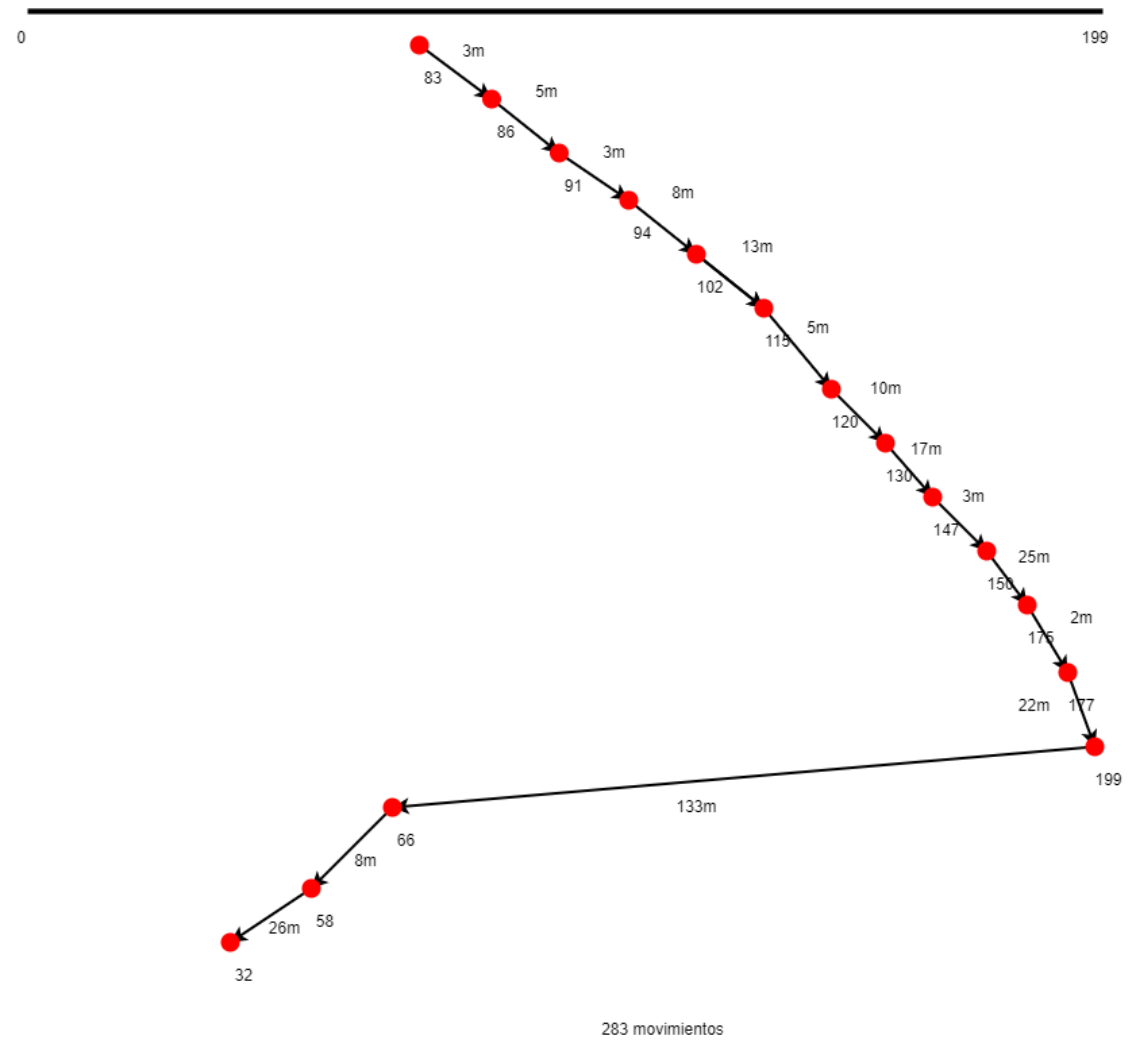
(a) FCFS (First Come, First Served)



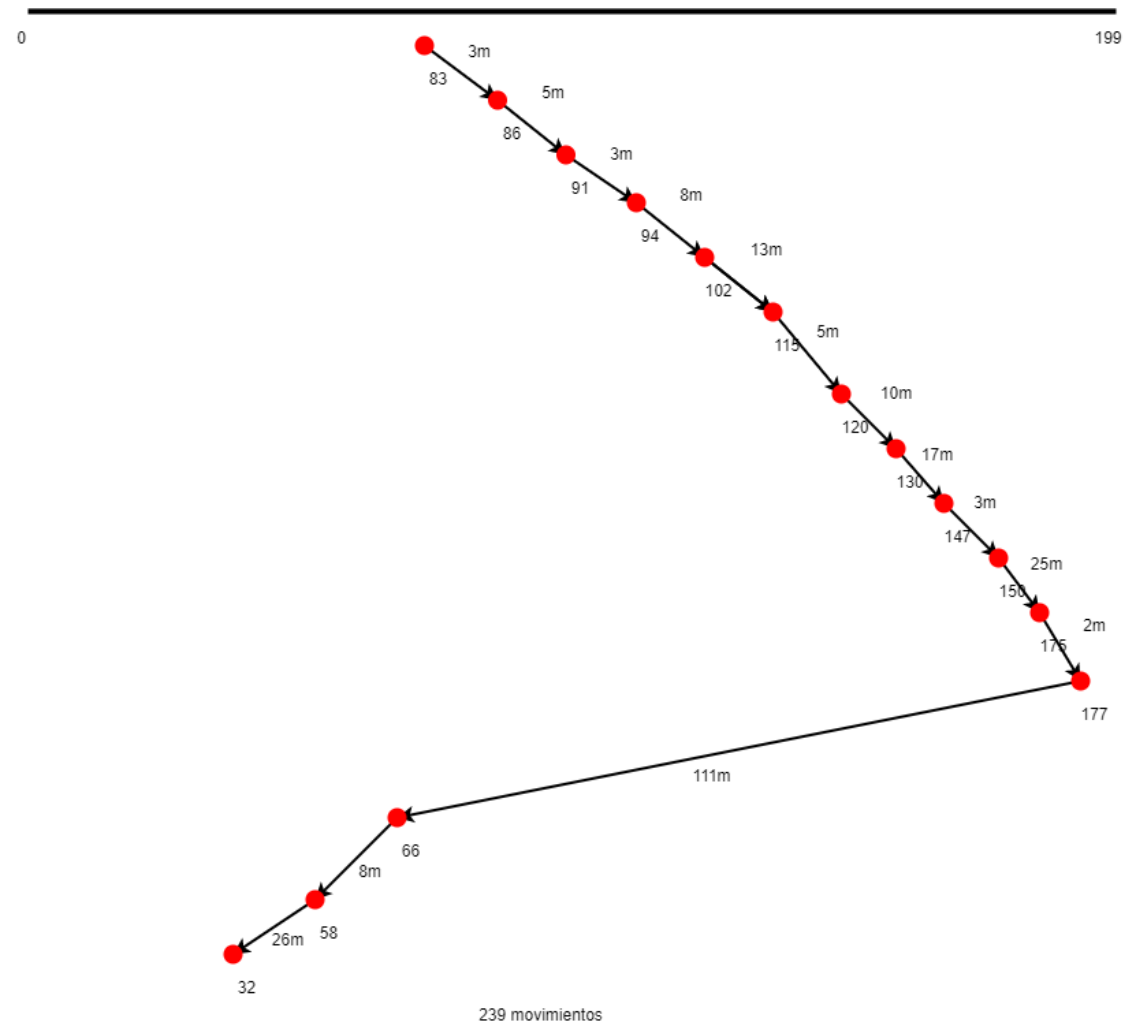
(b) SSTF (Shortest Seek Time First)



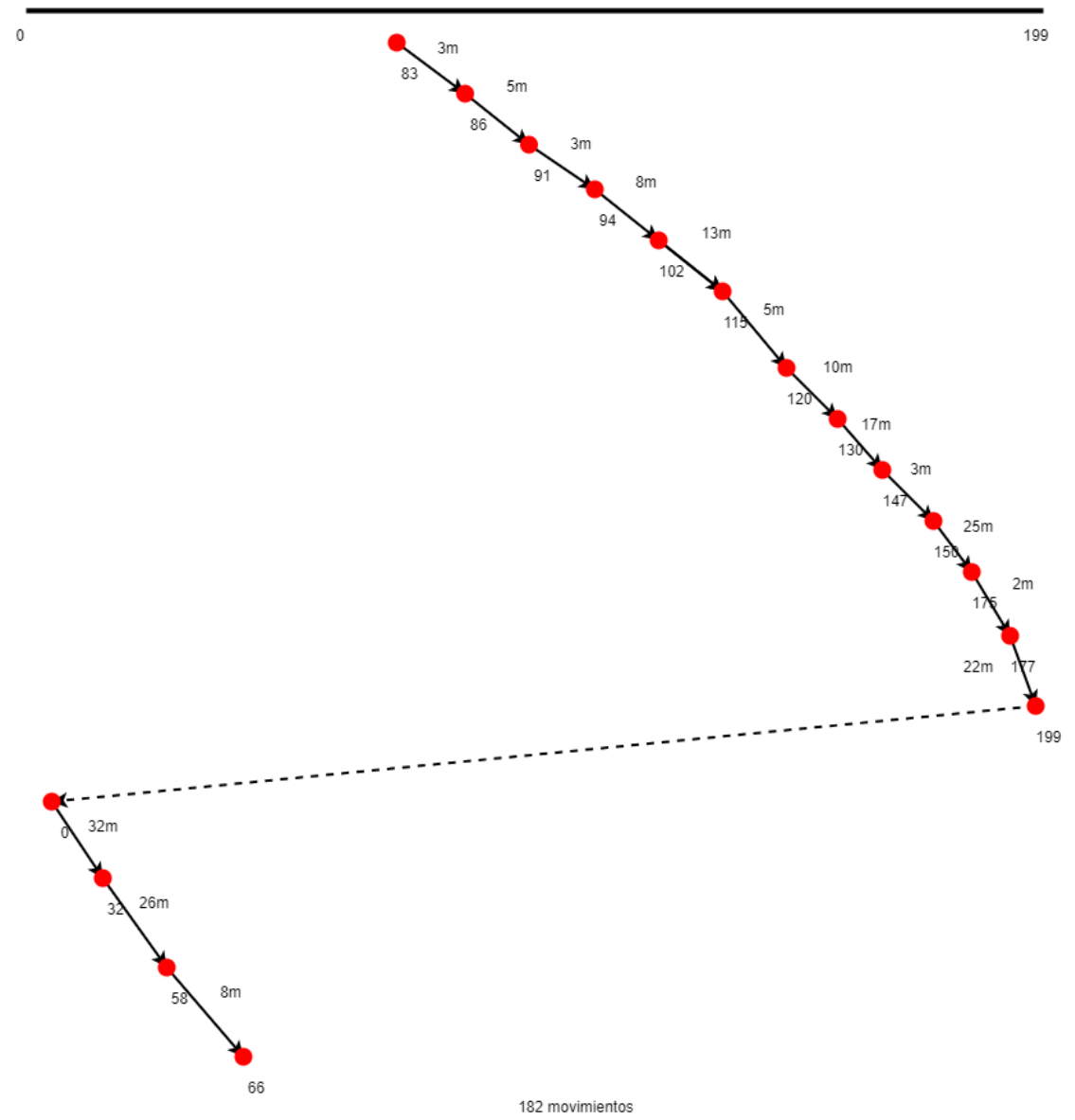
(c) Scan



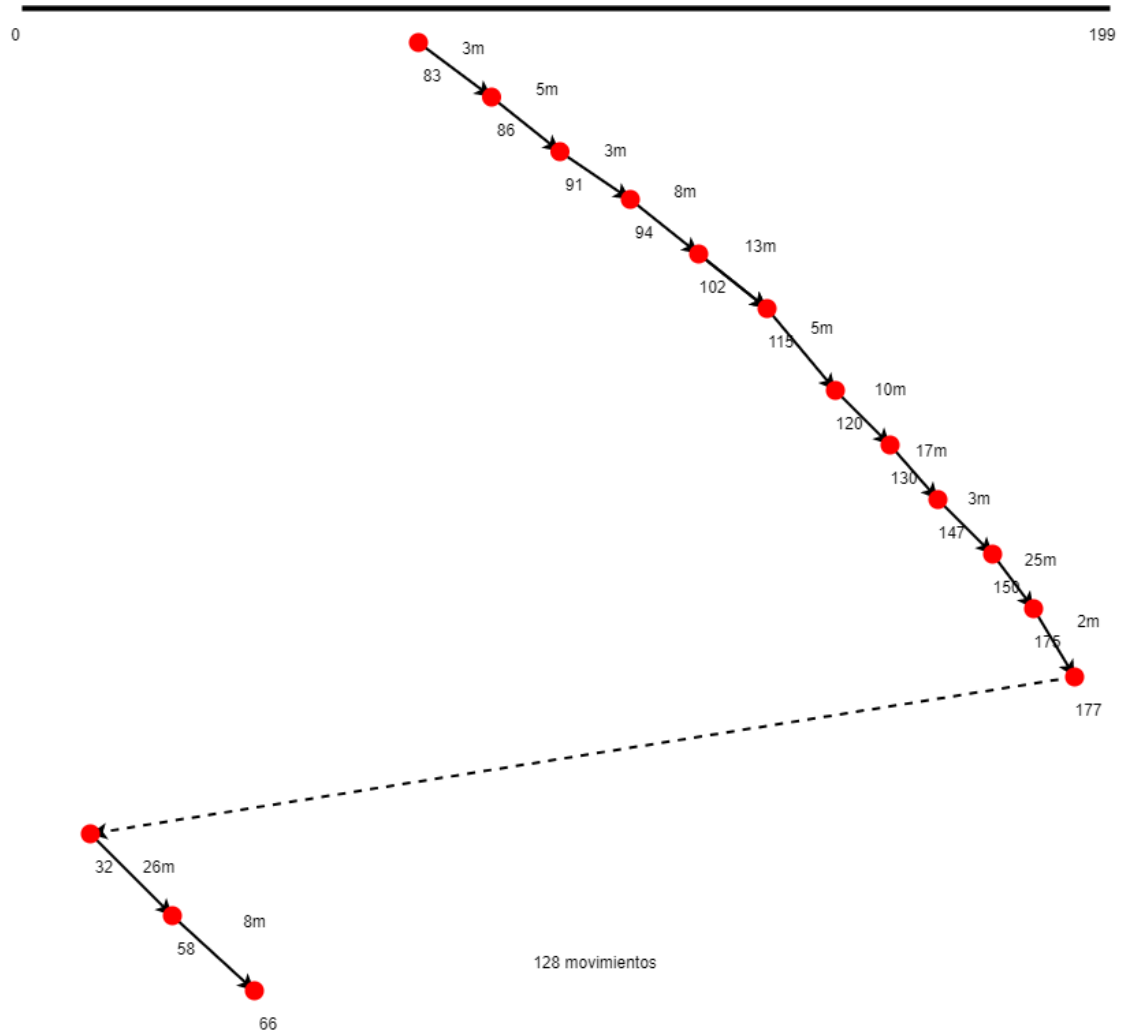
(d) Look



(e) C-Scan (Circular Scan)



(f) C-Look (Circular Look)



14. ¿Alguno de los algoritmos analizados en el ejercicio anterior pueden causar inanición de requerimientos?

Si, el SSTF

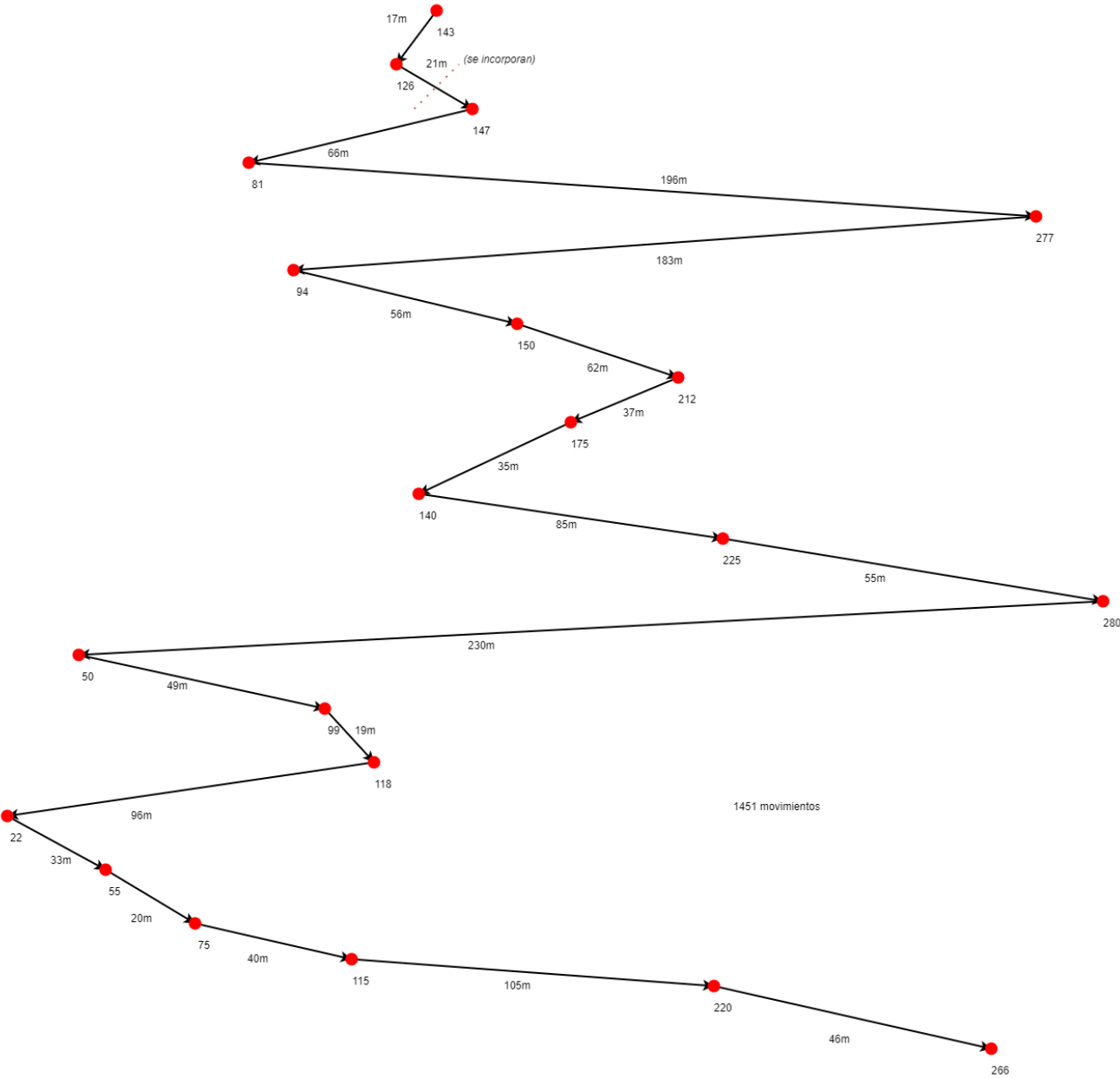
15. Supongamos un Head con movimiento en 300 pistas (numerados de 0 a 299), que esta en la pista 143 atendiendo un requerimiento y anteriormente atendió un requerimiento en la pista 125.

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:

(a) FCFS

0

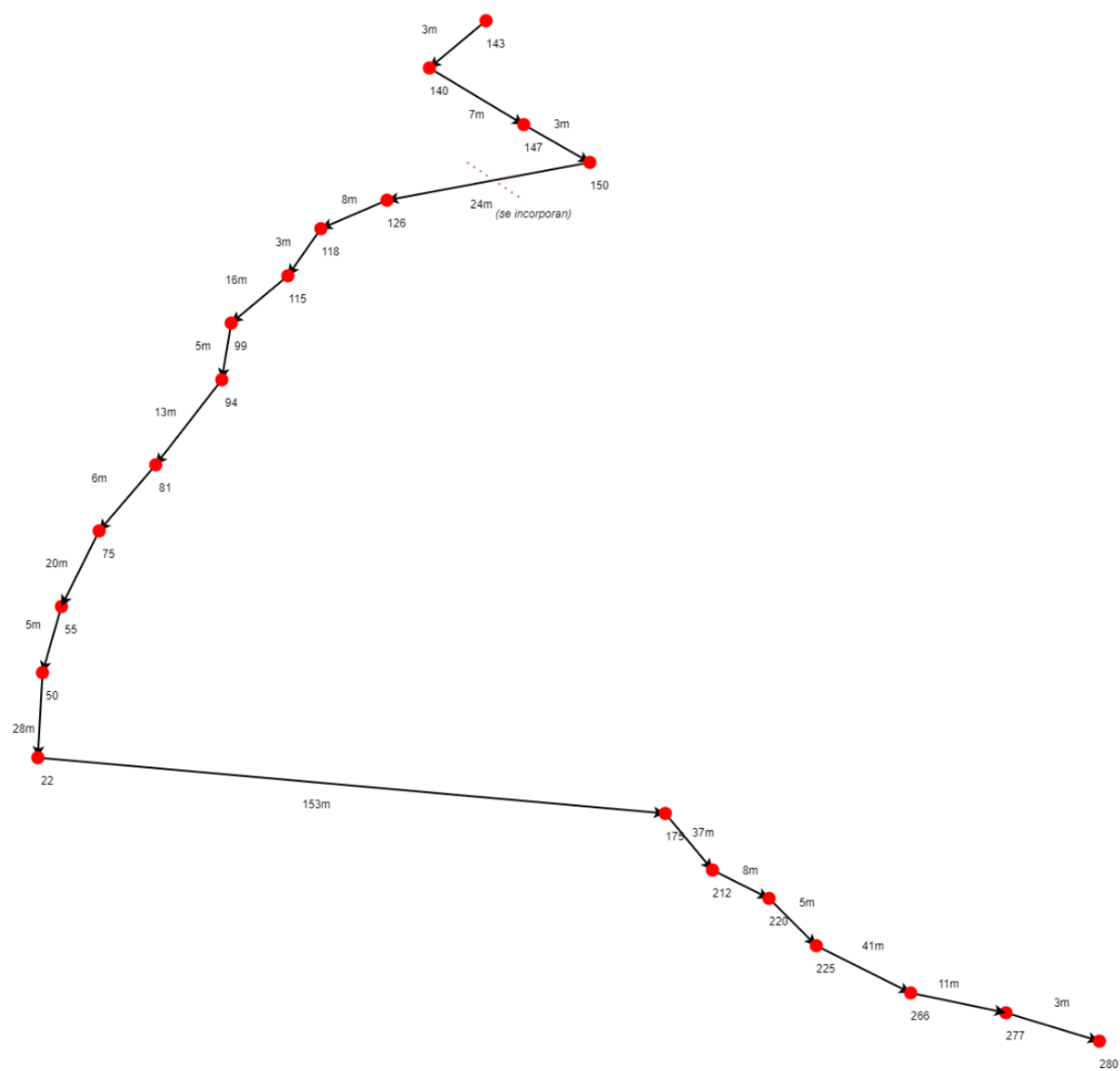
299



(b) SSTF

0

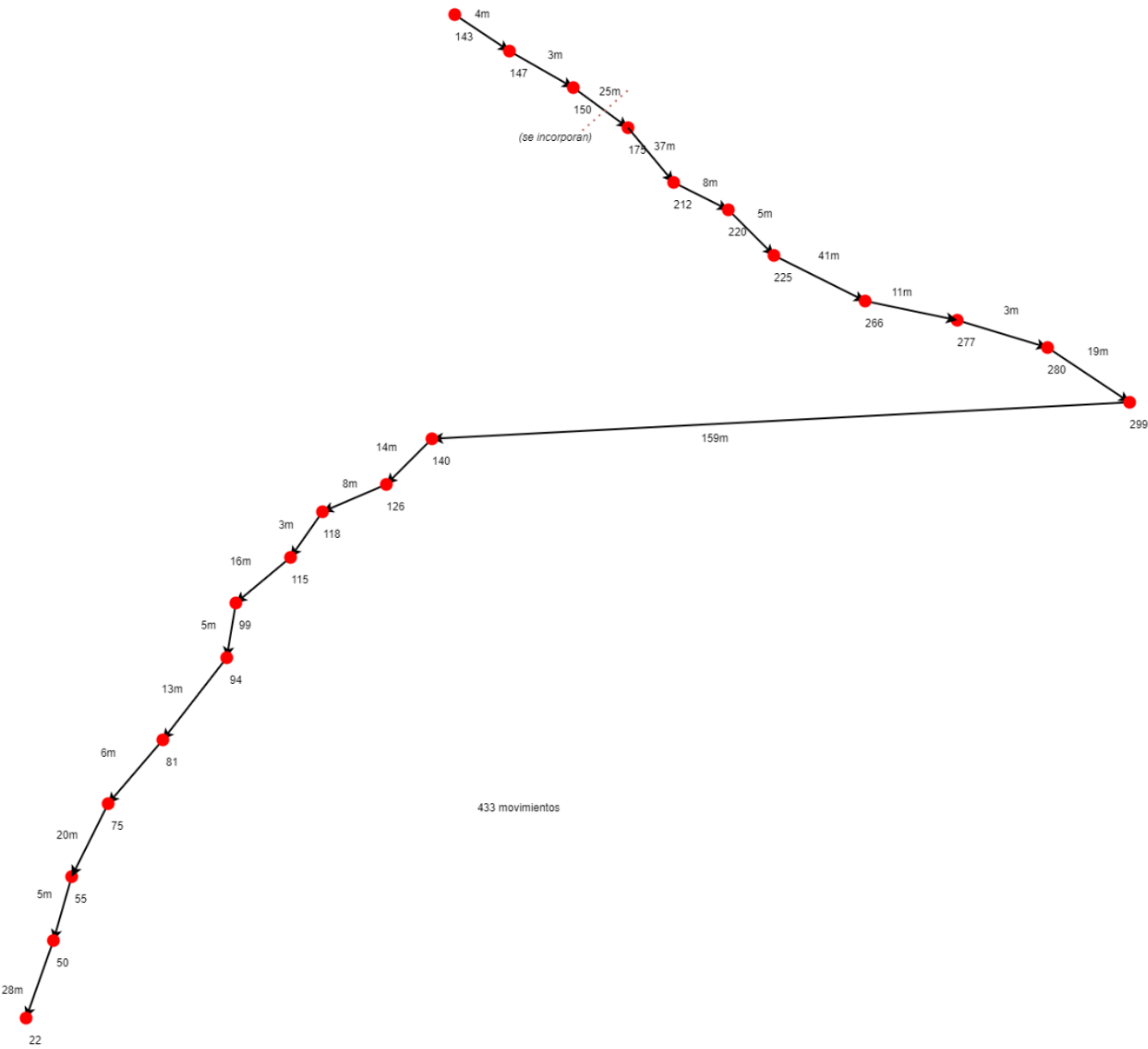
299



(c) Scan

0

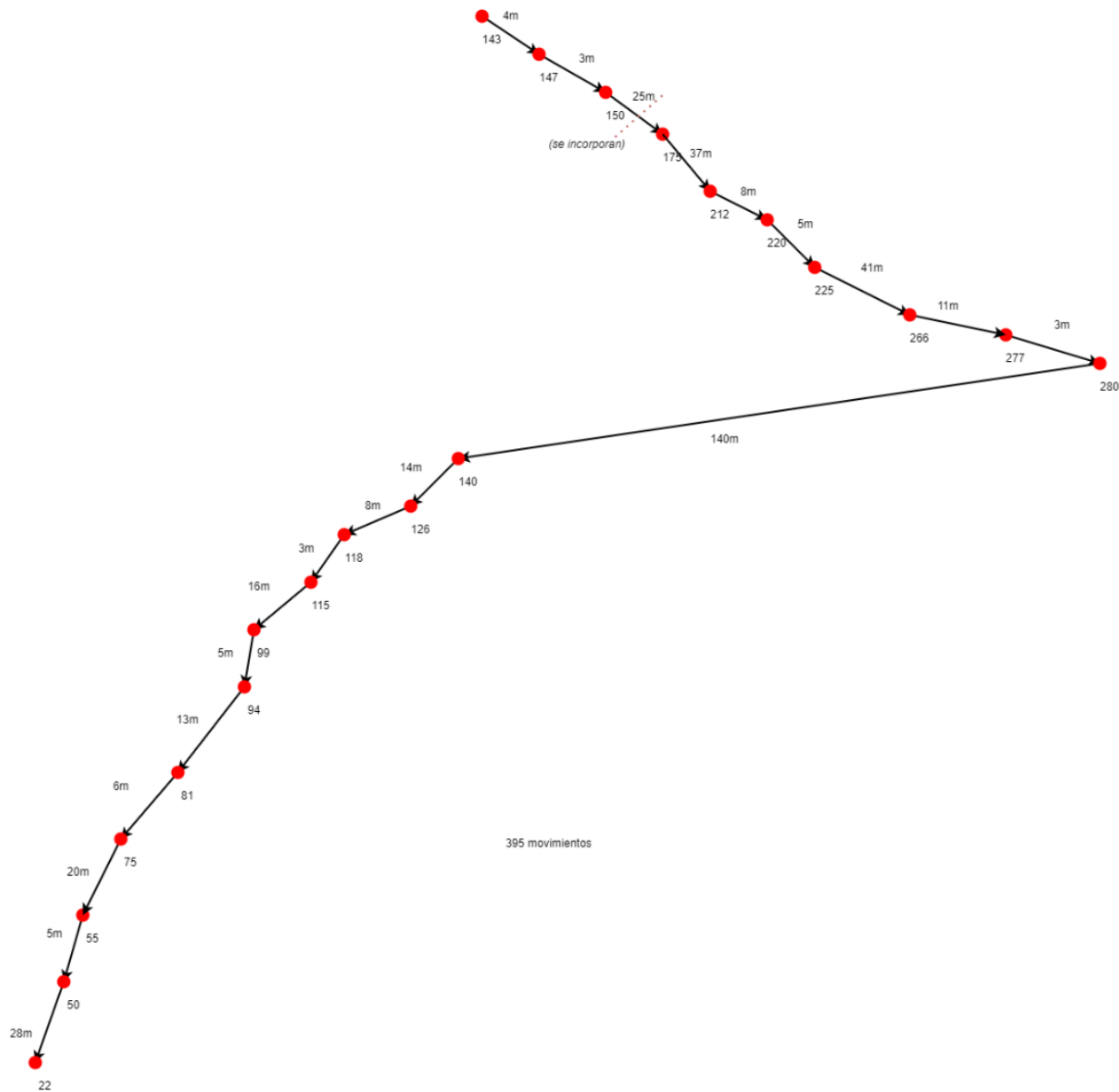
299



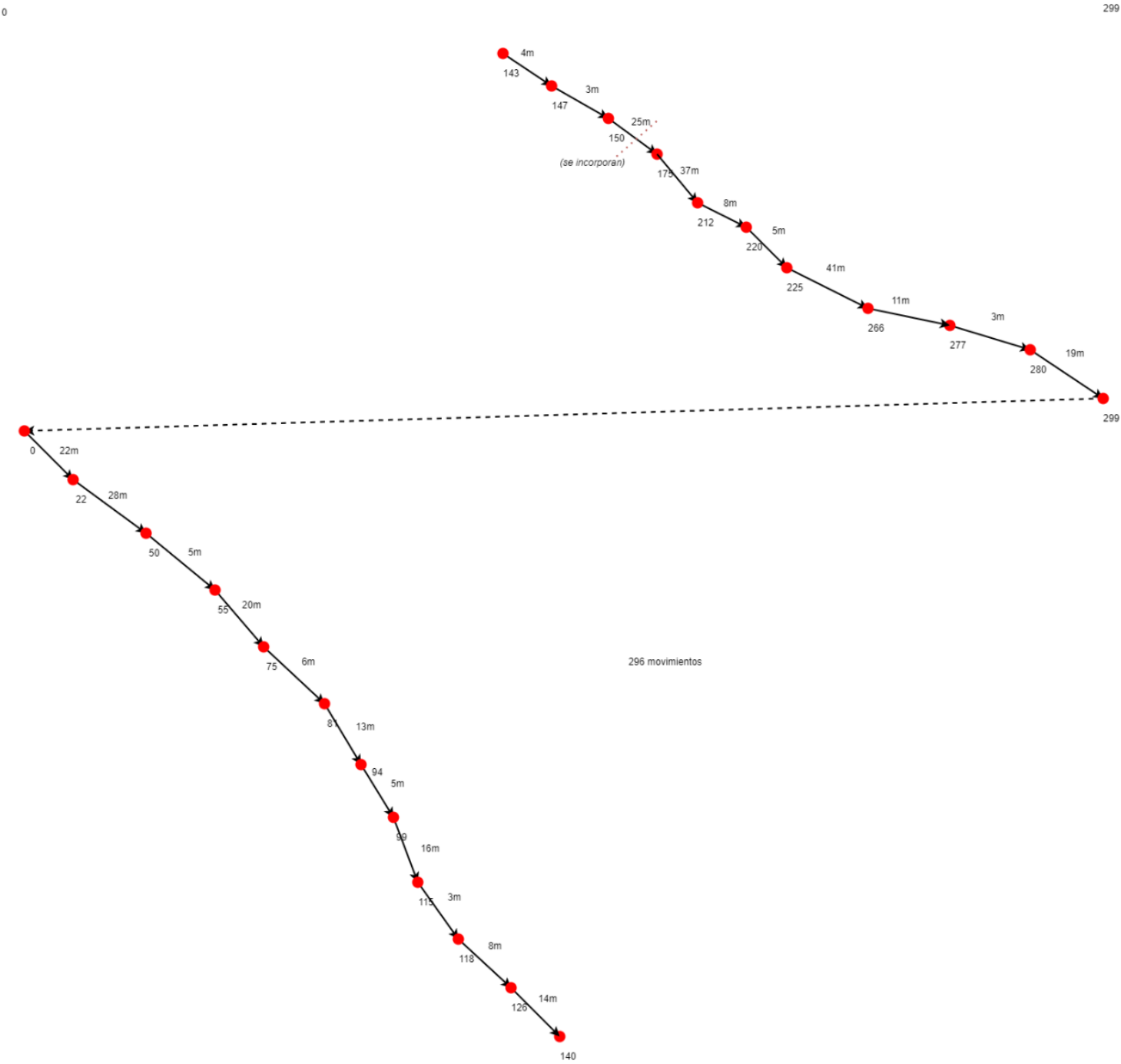
(d) Look

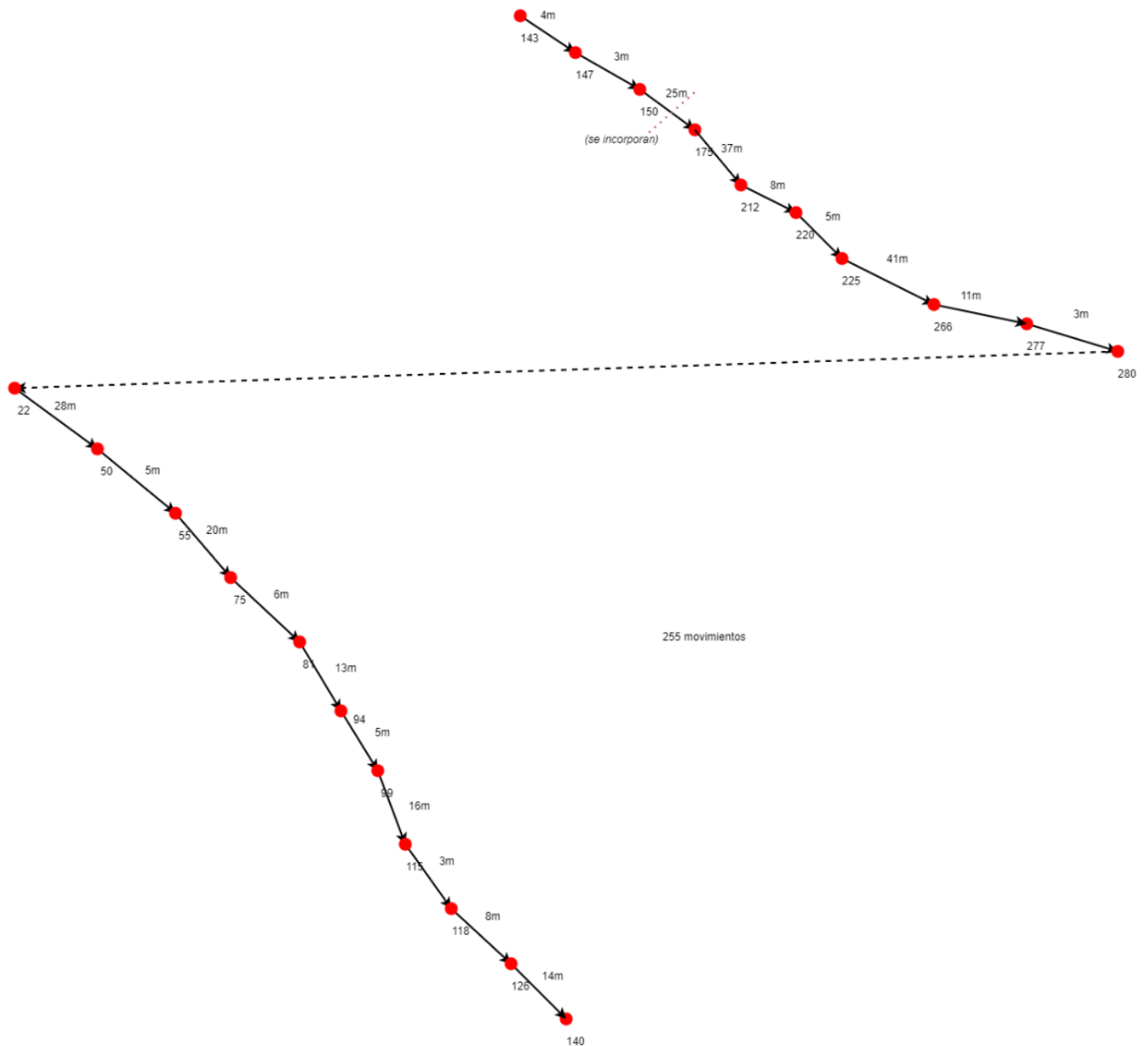
0

299



(e) C-Scan

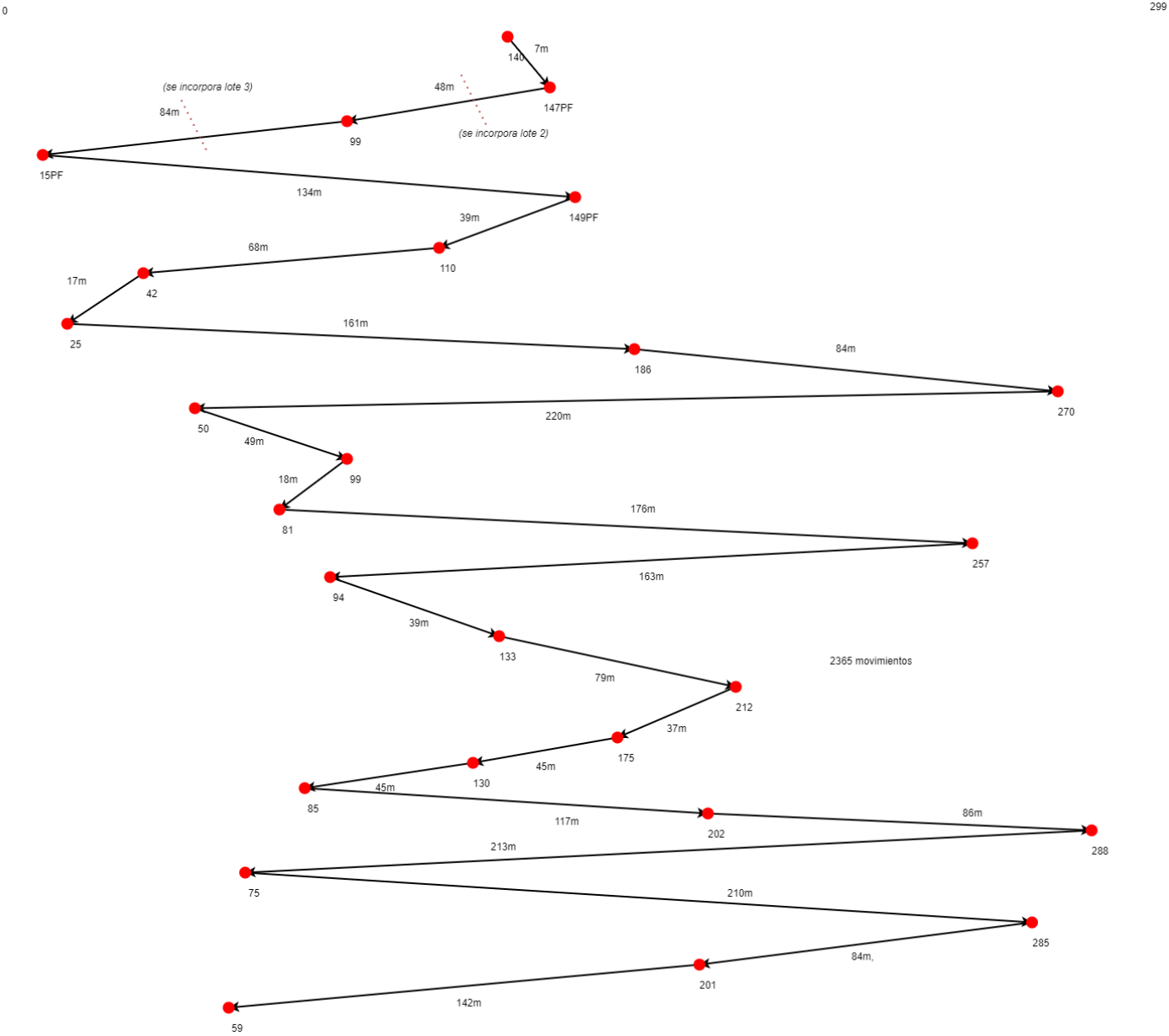




16. Supongamos un Head con movimiento en 300 pistas (numerados de 0 a 299), que esta en la pista 140 atendiendo un requerimiento y anteriormente atendió un requerimiento en la pista 135.

Si la cola de requerimientos es: 99, 110, 42, 25, 186, 270, 50, 99, 147PF, 81, 257, 94, 133, 212, 175, 130; y después de 30 movimientos se incorporan los requerimientos de las pistas 85, 15PF, 202 y 288; y después de otros 40 movimientos más se incorporan los requerimientos de las pistas 75, 149PF, 285, 201 y 59. 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:

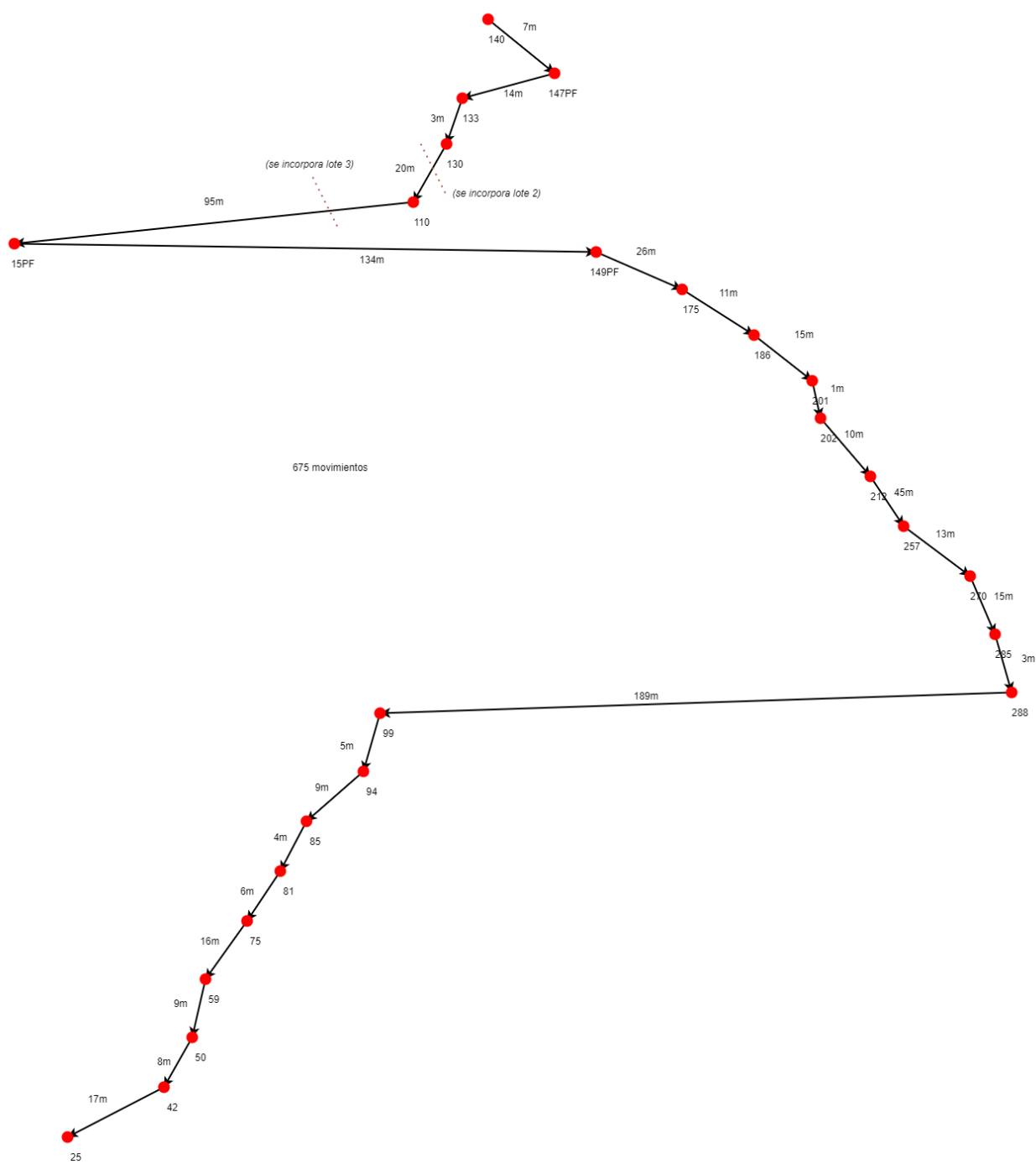
(a) FCFS



(b) SSTF

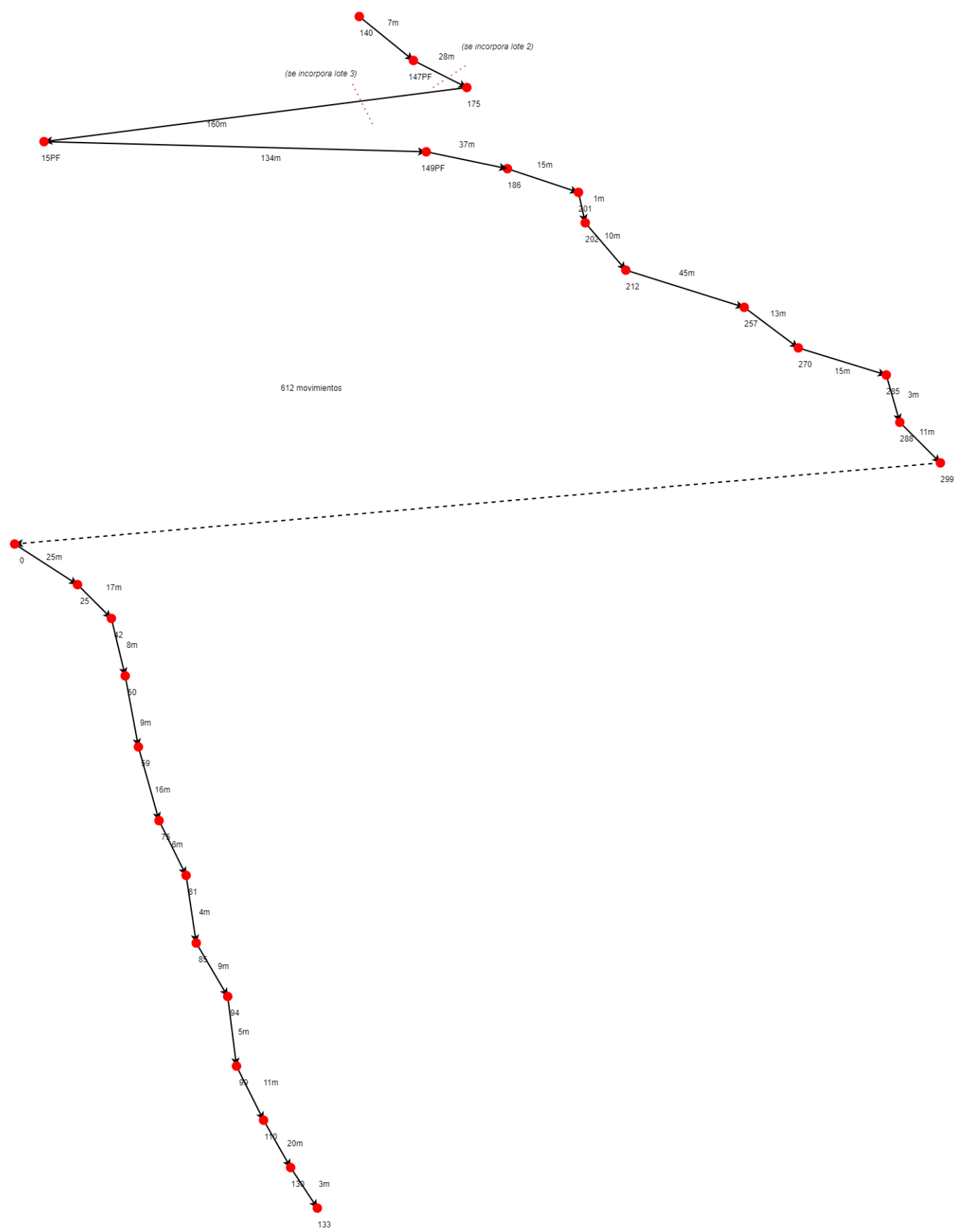
0

299

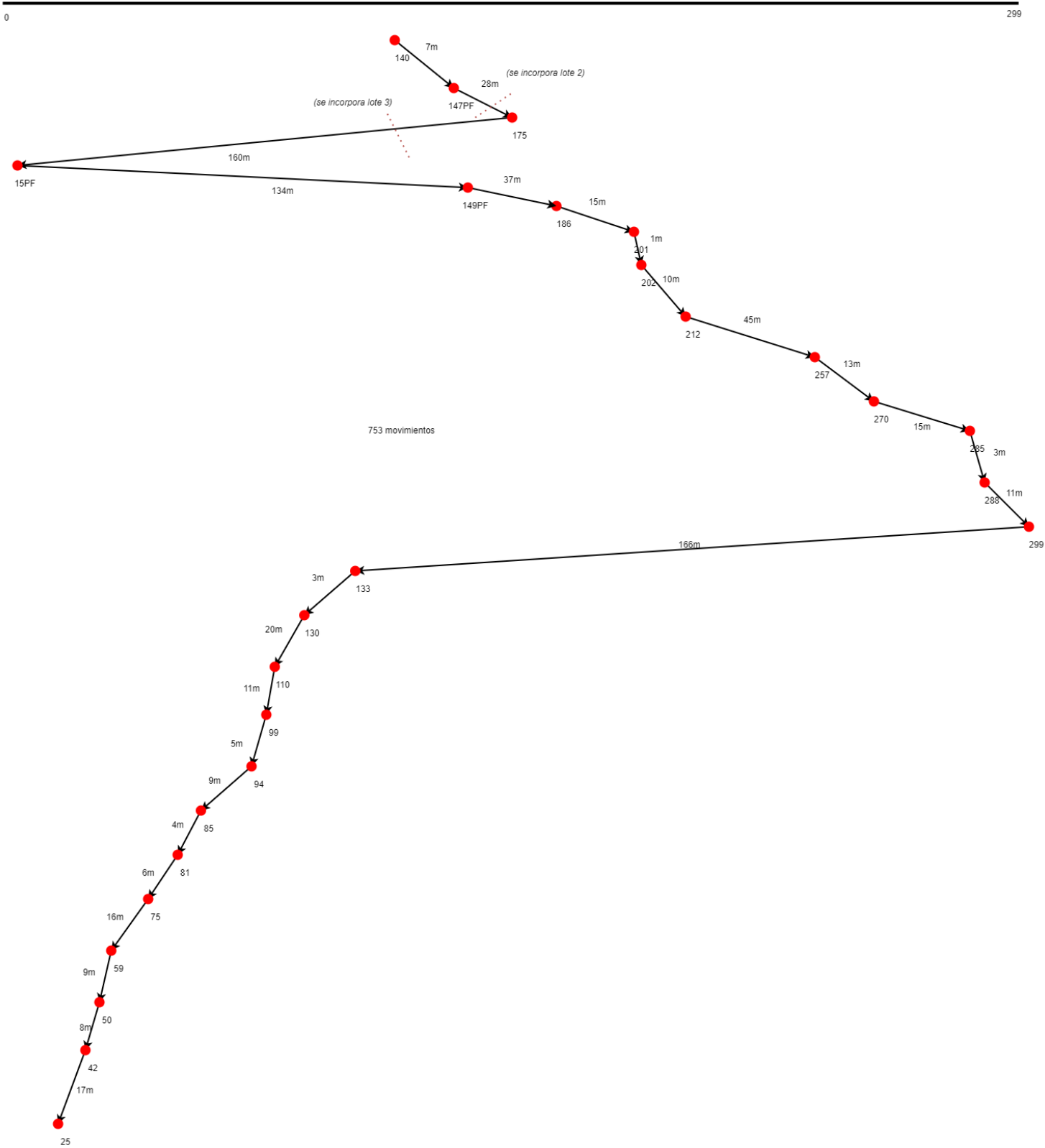


(c) C-Scan

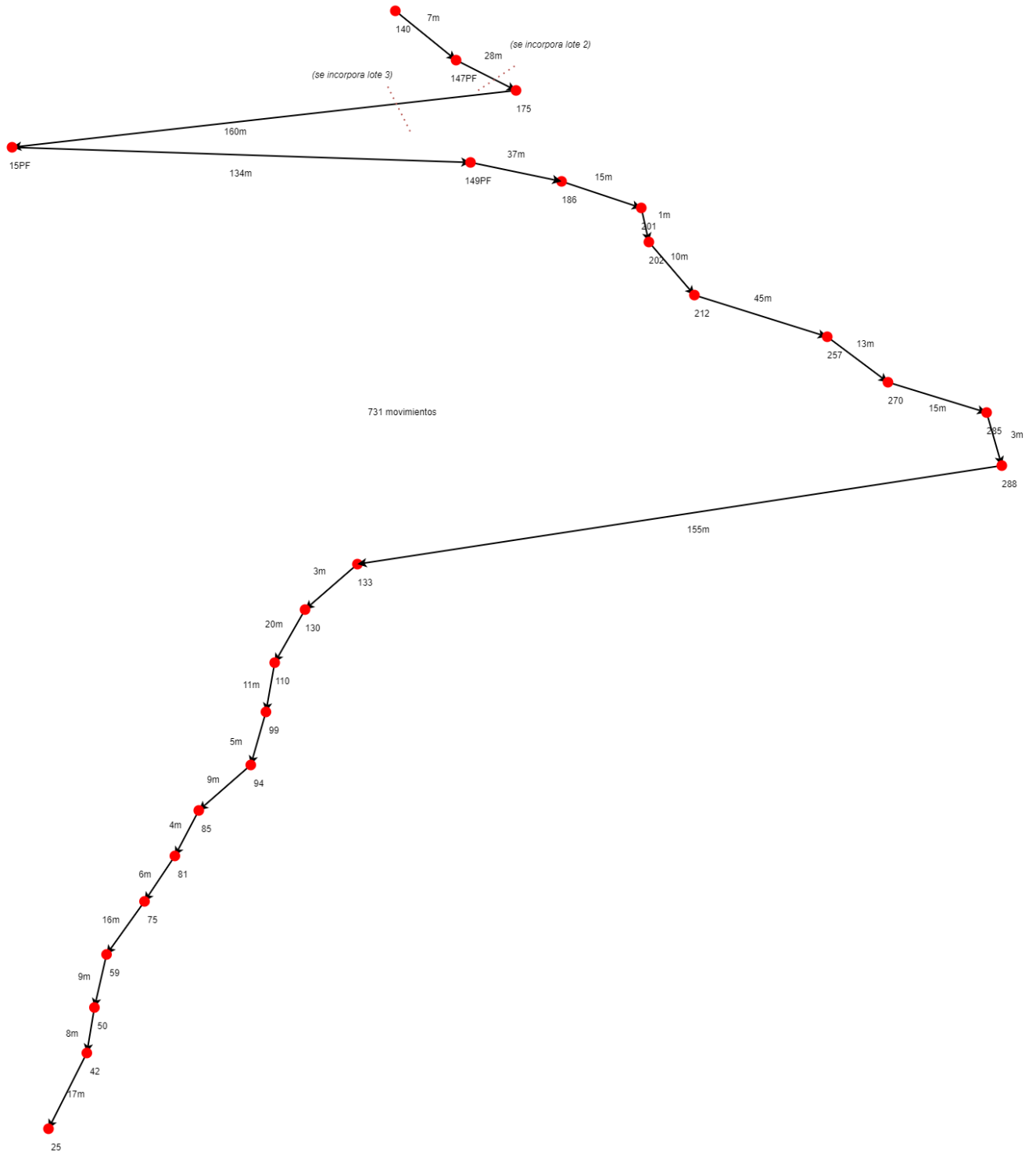
0 299



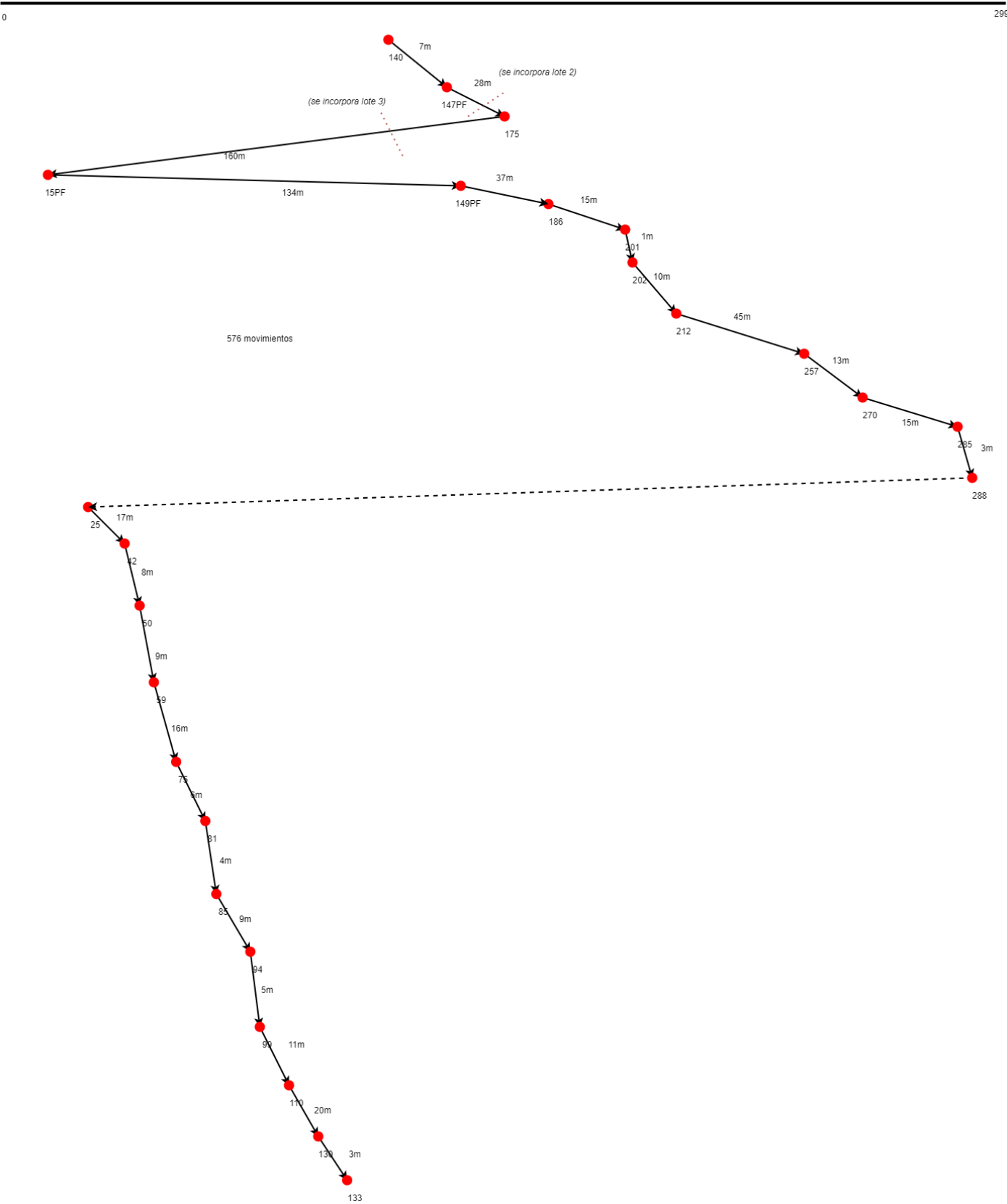
(d) Scan



(e) Look



(f) C-Look



Administración de Archivos

17. Dados los siguientes métodos de administración de espacio de un archivo:

Asignación contigua: Con asignación contigua, se asigna un único conjunto contiguo de bloques en tiempo de creación de los ficheros. Por tanto, hay una estrategia de preasignación que utiliza porciones de tamaño variable. La tabla de asignación de ficheros necesita sólo una entrada para cada fichero, mostrando el bloque inicial y la longitud del fichero. La asignación contigua es la mejor desde el punto de vista del fichero secuencial individual. Múltiples bloques se pueden leer de una vez para mejorar el rendimiento de E/S en procesamiento secuencial. Es también fácil obtener un único bloque. Por ejemplo, si un fichero comienza en el bloque b y se quiere acceder al bloque i -ésimo del fichero, su ubicación en almacenamiento secundario es simplemente $b + i - 1$. La asignación contigua presenta algunos problemas. Existirá fragmentación externa, haciendo difícil encontrar bloques contiguos de espacio de suficiente longitud. De vez en cuando, será necesario llevar a cabo un algoritmo de compactación para liberar espacio adicional en el disco. Además, con preasignación, es necesario declarar el tamaño del fichero en el tiempo de creación, con los problemas mencionados anteriormente.

Asignación enlazada: La asignación se realiza a nivel de bloques individuales. Cada bloque contiene un puntero al siguiente bloque en la cadena. De nuevo, la tabla de asignación de ficheros necesita sólo una entrada para cada fichero, mostrando el bloque inicial y la longitud del fichero. Aunque la preasignación es posible, es más común asignar bloques cuando se necesita. La selección de bloques es ahora una cuestión sencilla: cualquier bloque libre se puede añadir a una cadena. No hay fragmentación externa de la que preocuparse porque sólo se necesita un bloque cada vez. Este tipo de organización física se adapta mejor a ficheros secuenciales que se procesan secuencialmente. Seleccionar un bloque individual de un fichero requiere seguir la cadena hasta alcanzar el bloque deseado.

Una consecuencia del encadenamiento, tal como se describe, es que no existe principio de proximidad. Por tanto, si es necesario traer varios bloques de fichero a la vez, como en el procesamiento secuencial, se requiere una serie de accesos a diferentes partes del disco. Esto es tal vez un efecto más significativo en un sistema monousuario, pero también podría ser preocupante en el caso de un sistema compartido. Para resolver este problema, algunos sistemas consolidan ficheros periódicamente.

Asignación indexada: La asignación indexada resuelve muchos de los problemas de la asignación contigua y encadenada. En este caso, la tabla de asignación de ficheros contiene un índice separado de un nivel por cada fichero; el índice tiene una entrada por cada porción asignada al fichero. Típicamente, los índices de fichero no se almacenan físicamente como parte de la tabla de asignación de ficheros. Por el contrario, el índice de ficheros para un fichero se guarda en un bloque separado y la entrada para fichero en la tabla de asignación de ficheros apunta a dicho bloque. La asignación puede realizarse mediante bloques de tamaño fijo (Figura 12.11) o porciones de tamaño variable (Figura 12.12). La asignación por bloques elimina la fragmentación externa, mientras que la asignación por porciones de tamaño variable mejora la proximidad. En cualquier caso, la consolidación de ficheros se puede realizar de vez en cuando. La consolidación de ficheros reduce el tamaño del índice en el caso de porciones de tamaño variable, pero no en el caso de asignación de bloques. La asignación indexada da soporte tanto a acceso secuencial como directo a los ficheros y por tanto es la forma más popular de asignación de ficheros.

(a) Describa como trabaja cada uno.

(b) Cite ventajas y desventajas de cada uno.

18. Gestión de espacio libre. Dados los siguientes métodos de gestión de espacio libre en un disco:

Tabla de bits: Este método utiliza un vector que está formado por un bit por cada bloque en el disco. Cada entrada 0 corresponde a un bloque libre y cada 1 corresponde a un bloque en uso. Una tabla de bits tiene la ventaja de que es relativamente fácil encontrar un bloque libre o un grupo contiguo de bloques libres. Por tanto, una tabla de bits trabaja bien con cualquiera de los métodos de asignación de ficheros descrito. Esta estructura es tan pequeña como sea posible. La cantidad de memoria (en bytes) requerida para un mapa de bits de bloques es:

Tamaño de disco en bytes / 8 x tamaño de bloque del sistema de ficheros

Por tanto, para un disco de 16 Gbytes con bloques de 512 bits, la tabla de bits ocupa 4 Mbytes. Se puede almacenar la tabla de bits de 4 Mbytes en memoria principal? Si es así, entonces a la tabla de bits se puede acceder sin necesidad de acceder a disco. Pero incluso con los tamaños de memoria de hoy, 4 Mbytes es una cantidad considerable de memoria principal para dedicar a una única función (desventaja). La alternativa es poner la tabla de bits en disco. Pero una tabla de bits de 4 Mbytes requeriría alrededor de 8000 bloques de disco. No se puede hacer una búsqueda sobre esta cantidad de espacio de disco cada vez que se necesita un bloque, de tal forma que una tabla de bits residente en memoria es indicada.

Incluso cuando la tabla de bits está en memoria principal, una búsqueda exhaustiva de la tabla puede ralentizar el rendimiento del sistema de ficheros hasta un grado inaceptable. Esto es especialmente cierto cuando el disco está casi lleno y hay pocos bloques libres restantes. De forma análoga, la mayoría de los sistemas de ficheros que utilizan tablas de bits mantienen estructuras de datos auxiliares que resumen los contenidos de subrangos de la tabla de bits. Por ejemplo, la tabla se podía dividir lógicamente en varios subrangos de igual tamaño. Una tabla resumen podría incluir, para cada subrango, el número de bloques libres y el número de bloques libres contiguos de tamaño máximo. Cuando el sistema de ficheros necesita varios bloques contiguos, puede analizar la tabla resumen para encontrar un subrango apropiado y entonces buscar dentro de dicho subrango

Lista Ligada: Las porciones libres se pueden encadenar utilizando un puntero y valor de longitud en cada porción libre. Este método tiene una sobrecarga de espacio insignificante, porque no se necesita una tabla de asignación de disco, sino simplemente un puntero al comienzo de la cadena y la longitud de la primera porción. Este método es apropiado para todos los métodos de asignación de ficheros. Si se asigna un bloque cada vez, simplemente hay que escoger el bloque libre en la cabeza de la cadena y ajustar el primer puntero o el valor de la longitud. Si la asignación se hace de una porción de longitud variable, se debe utilizar un algoritmo de primer ajuste: se cargan las cabeceras de las porciones para determinar la siguiente porción libre apropiada en la cadena. De nuevo, se ajustan el puntero y los valores de longitud.

Este método tiene sus propios problemas. Después de cierto uso, el disco se quedará bastante fragmentado y muchas porciones serán de la longitud de un único bloque. También obsérvese que cada vez que se asigne un bloque, se necesita leer el bloque primero para recuperar el

puntero al nuevo primer bloque libre antes de escribir datos en dicho bloque. Si se necesitan asignar muchos bloques individuales de una vez para una operación sobre ficheros, esto ralentiza en gran medida la creación de ficheros. Similarmente, borrar ficheros altamente fragmentados es una tarea que consume mucho tiempo.

Agrupamiento: La técnica de indexación trata el espacio libre como un fichero y utiliza una tabla de índices tal y como se describió en la asignación de ficheros. Por motivos de eficiencia, el índice se debería utilizar en base a porciones de tamaño variable en lugar de bloques. Por tanto, hay una entrada en la tabla por cada porción libre en el disco. Esta técnica proporciona soporte eficiente a todos los métodos de asignación de ficheros.

Recuento: En este método, a cada bloque se le asigna un número secuencialmente y la lista de los números de todos los bloques libres se mantiene en una porción reservada del disco. Dependiendo del tamaño del disco, se necesitarán 24 o 32 bits para almacenar un único número de bloque, de tal forma que el tamaño de la lista de bloques libres es 24 o 32 veces el tamaño de la correspondiente tabla de bits y por tanto debe almacenarse en disco y no en memoria principal. Sin embargo, este es un método satisfactorio. Considérense los siguientes puntos:

1. El espacio en disco dedicado a la lista de bloques libres es menor que el 1% del espacio total de disco. Si se utiliza un número de bloque de 32 bits, entonces la penalización de espacio es de 4 bytes por cada bloque de 512 bytes.
2. Aunque la lista de bloques libres es demasiado grande para almacenarla en memoria principal, hay dos técnicas efectivas para almacenar una pequeña parte de la lista en memoria principal.
 - a. La lista se puede tratar como una pila (Apéndice 1B) con los primeros miles de elementos de la pila residentes en memoria principal. Cuando se asigna un nuevo bloque, se saca de la pila, que está en memoria principal. Similarmente, cuando se desasigna un bloque, se coloca en la pila. Cuando la porción de la pila en memoria se llena o se vacía, hay sólo una transferencia entre disco y memoria principal. Por tanto, esta técnica da acceso muy rápido en la mayoría de las ocasiones
 - b. La lista se puede tratar como una cola FIFO, con unos pocos miles de entradas desde la cabeza al final de la cola en memoria principal. Se asigna un bloque tomando la primera entrada de la cabeza de la cola y se desasigna añadiéndolo al final de la cola. Sólo hay una transferencia entre disco y memoria principal cuando la porción en memoria de la cabeza de la cola se vacía o la porción en memoria del final de la cola se llena.

En cualquiera de las estrategias listadas en el punto precedente (pila o cola FIFO), un hilo en segundo plano puede ordenar lentamente la lista en memoria o listas para facilitar la asignación contigua

(a) Describa como trabajan estos métodos.

(b) Cite ventajas y desventajas de cada uno