

RESEÑA DEL PROBLEMA

Problema	:	Fila
Complejidad	:	Baja
Tópico	:	Arreglos y Ciclos
Conocimientos	:	Básicos
Tamaño del Código	:	Pequeño
Codificación	:	Fácil
Estructuras de datos	:	Básicas
Propuesto por	:	Duvier Zuluaga, Colombia

ANTECEDENTES

El problema *Fila* busca realizar un seguimiento sencillo de una simulación de un proceso de la vida real como es el de hacer fila en un banco. No representa mayor dificultad, ya que es una actividad que todos han realizado alguna vez, y su implementación en computador es del mismo modo intuitiva.

APROXIMACIONES

Este problema es fácil y la primera aproximación que se visualiza es la de hacer un seguimiento por unidades de tiempo (es decir minuto a minuto, o segundo a segundo si se quiere).

De esta forma llevamos un reloj global que se incrementa en una unidad en cada paso, durante cada una de estas iteraciones simplemente se revisa si han llegado nuevas personas, si las que estaban en el cajero ya han terminado y se actualiza el tiempo de espera de las personas que están en la fila. Estas entidades se pueden representar fácilmente por medio de arreglos. Este proceso se realiza hasta que todas las personas hayan completado sus transacciones. A continuación se presenta un bosquejo del algoritmo:

```
Reloj <- 0
Iniciar la fila de personas como vacia
mientras haya personas sin terminar transaccion
    revisar que personas liberan los cajeros.
    Si hay personas en la fila mandarlas a los cajeros de
        ser posible.
    Computar la demora para las que logren servicio de
        cajero.
    Actualizar la demora de las personas que quedan en la
        fila.
    Revisar que personas llegan en este instante de tiempo
        y mandarlas a cajero si es posible o de lo
        contrario a hacer fila.
    Revisar las estadísticas solicitadas y actualizarlas
        si es necesario.
Reloj <- Reloj+1
```

Este algoritmo es bastante fácil de implementar. Su tiempo de ejecución está dominado por los tiempos de llegada y de transacción de las personas. Con las restricciones en la entrada del problema tenemos que basta para ejecutarse en el límite de tiempo de un segundo.

SOLUCION

La solución presentada toma otro enfoque del problema. Dado que sabemos que las personas en el archivo de entrada están ordenadas por su tiempo de llegada al banco, lo primero que podemos hacer es avanzar el reloj en saltos (lo que en simulación se conoce como el enfoque por eventos). En principio podríamos pensar en manejar dos eventos, el momento en que un cliente llega al banco, y el momento en que es atendido por el cajero. Entonces los avances de reloj estarán dados por el número de personas en vez de los tiempos de llegada y transacción (lo que es una gran ventaja porque las personas solo pueden ser 100, mientras que el tiempo puede llegar a ser 32765).

Si seguimos mirando condiciones del problema se puede ver que estamos interesados básicamente en estadísticas sobre la demora en la fila, entonces deberíamos mirar para cada persona la diferencia entre el tiempo en el que comienza a ser atendido y el tiempo en el que llegó. Como la persona comienza a ser atendida en el momento en que exista una ventanilla libre nos convendría tener una información sobre el tiempo en que se desocupará cada ventanilla (la cual podemos ir calculando a medida que las personas llegan al banco).

Teniendo toda esta información se puede delinear el algoritmo de la siguiente manera. Tenemos un arreglo en el que están los tiempos en que cada ventanilla estará libre, al inicio todas las ventanillas están libres en el tiempo cero. Para cada persona miramos si en el momento que llega existe algún cajero cuya marca de tiempo (el momento en que comienza a ser libre) sea menor, esto indica que el cajero va a estar libre en el momento en que llega esta persona. Si no hay cajeros libres entonces buscamos el que se desocupe más pronto para marcarlo como usado por esta persona, y marcamos el tiempo en que este cajero quede libre adecuadamente ($\text{horaInicioAtencion} + \text{demora}$). Con este proceso marcamos cuanto demora la persona esperando por el servicio de atención (cero si el cajero estaba libre antes de la llegada de la persona). En este punto miramos además si la persona se encuentra dentro del conjunto de los que tienen la demora máxima o si el mismo actualiza la demora máxima.

Ahora viene el problema de manejar las personas en la fila. Este proceso también se puede hacer persona a persona. En cada momento tenemos un arreglo que representa la hora en la que se comenzó (eventualmente comenzará si la ventanilla asignada se desocupa en el futuro) a atender a las personas anteriores, estas personas estarán haciendo fila cuando llegue esta persona únicamente si su hora de inicio de atención es mayor que la hora de llegada de esta persona, entonces revisando para cada persona la historia de tiempos de inicio de atención para las personas anteriores podemos calcular la longitud de la fila.

Este enfoque un poco más elaborado reduce aún más el tiempo de ejecución, pero debe tenerse cuidado en el manejo de los arreglos y los subíndices, en especial en el que representa la fila. El tiempo de ejecución está dominado por el número de personas en la entrada y el número de ventanillas, ya que para cada persona hacemos un recorrido lineal por ellas para ver cual está libre, como el número máximo de ventanillas es pequeño, este recorrido no representa mayor demora.

CODIGO

La solución propuesta esta compuesta de tres (3) funciones:

LeerParametros()

Simplemente lee el número de personas, ventanillas y los datos de los clientes.

Simular()

Realiza el procesamiento persona a persona, usando el arreglo *fila* para saber los momentos en los que se inicia la atención para cada cliente.

ImprimirSalida()

Imprime las estadísticas pedidas con el formato deseado.

MEJORAS AL ALGORITMO

Ya hemos visto que el algoritmo hace un proceso persona a persona, y para cada uno de ellos, lo realiza ventanilla a ventanilla. Una posible mejora para cuando el número de ventanillas sea mayor es usar otra estructura para el manejo de los tiempos en los que se desocupa cada ventanilla. Como en cada paso queremos averiguar cual es la ventanilla que se desocupa más pronto, podríamos usar una cola de prioridad para estas ventanillas. Si esta cola de prioridad se implementa por medio de un Heap, tenemos que para k ventanillas, obtener el tiempo de la ventanilla que se desocupa más pronto es una operación $O(1)$, la inserción de una nueva ventanilla (en este caso un nuevo tiempo en que se desocupa) y la remoción de una ventanilla son ambas de $O(\lg k)$. Este proceso aceleraría aún más el algoritmo, pero implica conocimiento de otras estructuras de datos, y una mayor complejidad en la codificación, lo cual para los tamaños dados en el problema no ameritaba el esfuerzo.

Cabe anotar que a pesar de que este problema fue el más fácil de todo el conjunto, y no representaba una gran complejidad, no fue resuelto correctamente por un gran porcentaje de los estudiantes participantes en la competencia.

Duvier Alexander Zuluaga Mora
dzuluaga@venus.uanarino.edu.co
duvier@hotmail.com