Enunciado de la Práctica de PRO2

Gestión de una terminal de contenedores

21 de octubre de 2020

Contenidos:

1. Enunciado de la práctica

Típicamente, una terminal de contenedores de carga es una gran explanada organizada en N hileras de contenedores, cada una de las cuales tiene M plazas. Las plazas son rectangulares y se sitúan una a continuación de otra formando una hilera. Entre las hileras hay pasillos para permitir la circulación de las grúas encargadas de colocar y retirar los contenedores. Las N hileras constituyen lo que se denomina *área de almacenaje*. Además suele existir un *área de espera* donde se pueden almacenar temporalmente un número pequeño de contenedores (aunque en esta práctica, y para simplificar, no se limitará la capacidad de este área).

Todos los contenedores tienen el mismo ancho y altura, pero difieren en su longitud: los hay de 1, de 2 y de 3 plazas[1]. Esto significa que un contenedor ocupará el equivalente a una, dos o tres plazas en función de su longitud. Por otro lado, los contenedores se pueden apilar unos sobre otros, siempre y cuando la totalidad de su base toque el suelo o contenedores en el "piso" inferior. Debido a las limitaciones de peso que los contenedores pueden soportar y a las limitaciones de las grúas de la terminal, sólo se pueden apilar hasta un máximo de H pisos. Habitualmente H será un valor pequeño, pero puede variar de una terminal a otra.

Las hileras de la terminal se numeran de 0 a $N-1$, y las plazas dentro de una hilera se numeran de 0 a $M-1$. Los niveles o pisos se numeran de 0 a $H-1$. Una *ubicación* es una terna de enteros $\langle i, j, k\rangle$ que designan, respectivamente, un número de hilera, un número de plaza y un piso. En una ubicación del área de almacenaje puede haber un contenedor (o parte de él) o un espacio libre. Se han de distinguir dos tipos diferentes de espacios libres: los *válidos*, aquellos que están sobre una superficie (que serán los espacios libres disponibles para insertar un contenedor), y los *inválidos*, aquellos que están por encima de otro espacio libre.

Definiremos como *hueco* a un grupo de plazas consecutivas libres válidas (en una hilera y piso determinados) que está delimitado a izquierda y derecha o por un contenedor o por los límites del área de almacenaje.

La ubicación de un contenedor es la terna $\langle i, j, k\rangle$ correspondiente a la plaza ocupada por el contenedor con menor j (recuérdese que hay contenedores que ocupan 2 y 3 plazas). Se usa el mismo convenio para la ubicación de un hueco. Notad entonces que la situación en el área de almacenaje de contenedores y huecos queda totalmente determinada por su ubicación y su longitud, formando lo que denominaremos *segmento*.

En una terminal tendremos dos operaciones básicas:

- `inserta_contenedor` y
- `retira_contenedor`.

Cuando un contenedor nuevo llega a la terminal (`inserta_contenedor`), el sistema debe encontrarle un lugar apropiado en el área de almacenaje, automáticamente. Si existen varios huecos en los que se puede poner el nuevo contenedor, necesitaremos una estrategia para decidir cuál de ellos elegir, intentado que:

[1] En realidad, la situación es más compleja y la variabilidad en las medidas es mayor, pero aquí trabajaremos con un modelo simplificado.

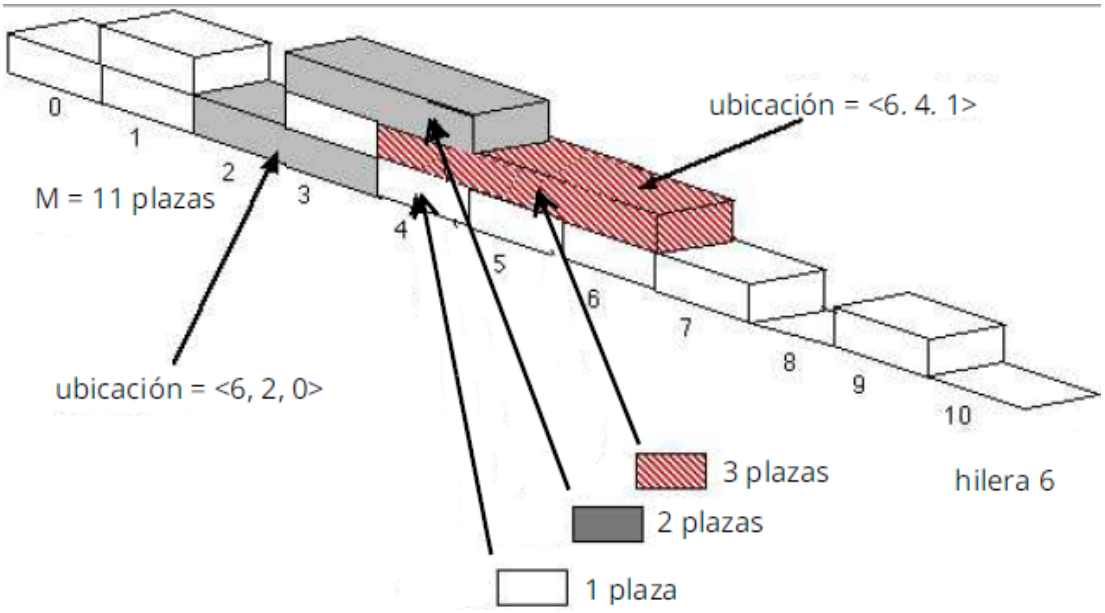

Figura 1: Esquema de una hilera de la terminal de contenedores

1. sea sencilla de implementar y el algoritmo correspondiente sea eficiente en tiempo de ejecución y en espacio de memoria;

2. aproveche lo mejor posible el espacio de la terminal, minimizando en la medida de lo posible la *fragmentación*[2].

Además de establecer qué espacio libre elegir para un nuevo contenedor, una estrategia dada podría intentar crear espacio libre para un nuevo contenedor reorganizando los ya existentes, pero en esta práctica no contemplaremos dicha posibilidad.

Si, por cualquier motivo, un contenedor no puede ser colocado en el área de almacenaje, entonces se coloca en el área de espera.

Por otro lado, cuando queremos quitar un contenedor `C` de la terminal (`retira_contenedor`) a menudo nos encontraremos con que `C` tiene otros contenedores encima y nos veremos obligados a moverlos al área de espera, para que no obstruyan la salida de `C`. A su vez, estos pueden tener otros encima, que también deberemos mover al área de espera y así sucesivamente.

Por ejemplo, consideremos una terminal con $H = 3$ pisos y $M = 5$ plazas por hilera y una hilera 0 representada así (además de un área de espera vacía):

```
espera:    -

piso 2     T T   U
piso 1     R Y Y Y Z
piso 0     W W W X X
           0 1 2 3 4
```

[2]Se habla de *fragmentación* cuando no se puede ubicar a un contenedor en el área de almacenaje habiendo sin embargo plazas no ocupadas suficientes.

El contenedor X (de 2 plazas) ocupa las plazas $\langle 0, 3, 0\rangle$ y $\langle 0, 4, 0\rangle$ y encima de él están los contenedores Y (de 3 plazas, su primera ubicación es la $\langle 0, 1, 1\rangle$) y Z (de 1 plaza, ocupando la ubicación $\langle 0, 4, 1\rangle$). Si queremos retirar el contenedor X tendremos que mover al área de espera los contenedores Y y Z, lo que a su vez obliga a mover al área de espera los contenedores que quedan encima de ellos, en este caso T y U.

En general, si hemos de retirar un contenedor C, se aplicará la siguiente política a los contenedores afectados, comenzando por C: un contenedor que no tenga ningún otro contenedor encima (o parte de alguno) debe retirarse o moverse inmediatamente al área de espera (según si es C o alguno de los que lo obstruyen). En caso contrario, debe retirarse o moverse al área de espera inmediatamente después de que se hayan movido al área de espera los contenedores del piso superior que lo obstruyen, en orden de menor a mayor plaza (gráficamente, de izquierda a derecha).

Así pues, en el ejemplo anterior, para retirar X tenemos que mover los contenedores T, U, Y y Z al área de espera, en ese orden. Una vez retirado X, tenemos esta situación:

```
espera: (en orden de salida, ver BEST_FIT)  Z, Y, U, T

piso 2
piso 1    R
piso 0    W W W
          0 1 2 3 4
```

Una vez se completa con éxito la inserción o retirada de un contenedor en el área del almacenaje, se recorrerán los contenedores del área de espera para recolocar en el área de almacenaje tantos como se pueda; es importante que en el área de espera haya el mínimo posible de contenedores. En el ejemplo de más arriba deberíamos intentar colocar de nuevo los contenedores Z, Y, U, T de nuevo en la terminal.

En esta práctica tendréis que implementar la estrategia BEST_FIT, que dicta qué hueco debe utilizarse para insertar un contenedor en el área de almacenaje y en qué orden se guardan los contenedores en el área de espera (y despúes se intentan recolocar en el área de almacenaje cuando hay oportunidad). Esta estrategia, que se describe con detalle en la subsección 1.1, nos acabaría dando, en el ejemplo anterior, la situación final siguiente después de retirar X y recolocar los contenedores del área de espera:

```
espera:   -

piso 2    Z
piso 1    R U Y Y Y
piso 0    W W W T T
          0 1 2 3 4
```

Finalmente, notad que después de insertar o retirar contenedores, los huecos del piso donde se ha realizado la inserción/retirada y los del del piso superior a éste quedan modificados, necesitando en algunos casos la *fusión* de huecos adyacentes en un único hueco (por la propia definición de hueco) y en otros casos la *subdivisión* de un hueco en otros huecos y en un segmento de espacio libre inválido.

1.1. La estrategia BEST_FIT

La estrategia BEST_FIT establece:

1. qué hueco elegir para un contenedor si hay varias posibilidades;
2. cómo gestionar el área de espera.

Durante una inserción, si existen varios lugares libres posibles para el nuevo contenedor, `BEST_FIT` elige el que mejor se ajuste al tamaño del contenedor. Por ejemplo, si el contenedor es de 2 plazas se intentará colocar en un hueco de exactamente 2 plazas; en su defecto, se intentará colocar en uno de 3 plazas (y quedará un hueco de 1 plaza), si tampoco hay huecos de 3 plazas entonces se intentará con uno de 4 plazas, y así sucesivamente. En caso de que existan varios huecos del mismo tamaño con el mejor ajuste posible para el contenedor a ubicar, entonces se escogerá el que tenga menor índice de hilera y en caso de empate el de menor índice de plaza (no puede haber dos huecos con igual número de hilera y de plaza).

Si no hubiese ningún hueco capaz de albergar al contenedor, `BEST_FIT` no pretende realizar ninguna reorganización de los contenedores del área de almacenaje, y se limita a poner al contenedor nuevo en el área de espera.

`BEST_FIT` gestiona el área de espera de manera que los últimos contenedores agregados al área de espera son los que se intentan recolocar en primer lugar. Cada vez que la operación `inserta_contenedor` tiene éxito al colocar el nuevo contenedor en el área de almacenaje es posible que se creen huecos válidos para algún contenedor del área de espera , por lo que se recorrerá el área de espera, empezando por el que se añadió en último lugar, en busca de un contenedor que pueda ser movido al área de almacenaje. Si la búsqueda tiene éxito se emplea la regla `BEST_FIT` para mover el contenedor en cuestión desde el área de espera a la de almacenaje. El proceso recién descrito se repite hasta que el área de espera se ha examinado en su totalidad sin que haya ningún contenedor susceptible de ser movido al área de almacenaje.

En cuanto a la operación de `retira_contenedor`, la estrategia `BEST_FIT` se limita a quitar los contenedores que haya por encima del contenedor a retirar (si hay), poniéndolos en la área de espera. Después se recorre ésta, intentando reubicar contenedores en el área de almacenaje y siguiendo un proceso idéntico al explicado anteriormente para la inserción.

Como ejemplo concreto de funcionamiento, supongamos una terminal (muy pequeñita) con $N = 1$, $M = 6$ y $H = 3$, inicialmente vacía. Usaremos la nomenclatura `i` m ℓ para indicar que se inserta un contenedor con matrícula m y ℓ plazas, y `r` m para indicar que se ha de retirar el contenedor de matrícula m. Supongamos que se han de procesar las siguientes "peticiones":

```
i A 3, i B 2, i C 3, i D 1, i E 1, i F 3, i G 1,
i H 2, r A, r E, r C, i I 1, i J 3, i K 2
```

Justo antes de retirar el contenedor `A` el estado de la terminal sería el siguiente:

```
espera: -
```

```
piso 2  D E H H
piso 1  B B F F F G
piso 0  A A A C C C
```

Para retirar el contenedor `A` se traspasan al área de espera los contenedores `D`, `E`, `B`, `H` y `F`, por este orden (recordad como lo describimos en la sección anterior: `B` ha de moverse *inmediatamente* después de haber movido `D` y `E`).

Una vez retirado `A` se recolocan todos los contenedores del área de espera en el área de almacenaje aplicando el criterio de BEST_FIT.

El estado final de la terminal al concluir la operación `retira_contenedor(A)` es

```
espera: -

piso 2  B B       E
piso 1  H H D     G
piso 0  F F F C C C
```

Si continuamos procesando peticiones de inserción y retirada (`r E`, `r C`, ...) el resultado final es:

```
espera: -

piso 2  B B G
piso 1  H H D J J J
piso 0  F F F I K K
```

Al intentar insertar `J` no hay lugar para colocarlo, de modo que queda en el área de espera; pero después se inserta `K` y entonces puede colocarse `J` encima.

2. Funcionalidades

2.1. Decisiones sobre los datos

1. Las matrículas de los contenedores son strings formadas exclusivamente por letras mayúsculas de la A a la Z y dígitos del 0 al 9; además comienzan siempre con una letra. Asumiremos que todas las matrículas especificadas para los contenedores de la terminal son correctas.

2. Todas las especificaciones de longitud de los contenedores son también correctas; se usará 1, 2 y 3 para indicar el número de plazas que ocupan.

3. Todos los parámetros de creación de una terminal (N, M y H) dados con cada comando `crea_terminal` son correctos. En particular siempre se cumplirá: (1) el número de hileras $N > 0$; (2) el número de plazas $M > 0$ y (3) el número de pisos $H > 0$

4. Las ubicaciones del área de almacenaje se especifican mediante una terna $\langle i, j, k\rangle$ donde i es la hilera ($0 \leq i < N$), j el número de plaza ($0 \leq j < M$) y k el número del piso ($0 \leq k < H$). Por convenio, la ubicación de todos los contenedores en el área de espera será $\langle -1, 0, 0\rangle$. También por convenio $\langle -1, -1, -1\rangle$ se utilizará para indicar que un contenedor no se encuentra en la terminal, ni en su área de almacenaje ni en su área de espera.

2.2. Programa principal: estructura y comandos

La entrada del programa principal está formada por una secuencia de rondas; cada ronda consiste en un conjunto de comandos que se aplican sobre la terminal creada con el primer comando de cada ronda, excepto la última ronda, que consiste en un único comando `fin`. Cuando comienza una nueva ronda, la terminal de la ronda previa se destruye. La estructura general del programa principal sería la siguiente:

```
string comando = "";
lee comando;
while (comando != "fin") {
   // comando == "crea_terminal N M H"
   procesa comando;
   lee comando;
   while (comando != "fin" and comando != "crea_terminal") {
      procesa comando;
      lee comando;
   }
}
```

Los comandos aceptados se describen a continuación. Todos ellos generan algún tipo de salida, pero en este documento solo se comentan las más importantes. La sintaxis exacta de la salida de cada comando se podrá derivar del juego de pruebas público del ejercicio creado en el Jutge para cada entrega.

1. `crea_terminal` N M H: inicia una nueva ronda, creando una terminal de contenedores con N hileras, M plazas por hilera y H pisos por plaza. Se asume que los parámetros son correctos. Tanto el área de almacenaje como el área de espera de la terminal están vacías.

2. `inserta_contenedor` m ℓ: inserta un contenedor con matrícula m que ocupará ℓ plazas. El comando admite la forma abreviada `i` m ℓ. Si ya existía un contenedor en la terminal con la misma matrícula se imprime un mensaje de error. En caso contrario se imprime la primera ubicación ocupada por el contenedor (la de menor número de plaza) si el contenedor se ha logrado ubicar en el área de almacenaje y $\langle -1, 0, 0\rangle$ si el nuevo contenedor se ha tenido que dejar en el área de espera.

3. `retira_contenedor` m: retira el contenedor con matrícula m del terminal. Admite la forma abreviada `r` m. Se imprime un mensaje de error si no hay ningún contenedor con matrícula m en la terminal.

4. `donde` m: imprime la ubicación del contenedor de matrícula m en la terminal; si está en el área de almacenaje se imprime la primera ubicación ocupada por el contenedor (la de menor número de plaza), si está en el área de espera se imprime $\langle -1, 0, 0\rangle$ y si no está entonces se imprime $\langle -1, -1, -1\rangle$.

5. `longitud` m: imprime la longitud, en número de plazas, del contenedor cuya matrícula es m o un error si no existe un contenedor con la matrícula dada en la terminal.

6. `contenedor_ocupa` i j k: imprime la matrícula del contenedor que ocupa la ubicación $\langle i, j, k\rangle$ del área de almacenaje; no se imprime nada si la ubicación no está ocupada por ningún contenedor y se imprime un error si $\langle i, j, k\rangle$ designa una ubicación fuera del área de almacenaje de la terminal. Nótese que si `contenedor_ocupa` i j k nos devuelve la matrícula m, entonces `donde` m no necesariamente imprime $\langle i, j, k\rangle$, puesto que es posible que $\langle i, j, k\rangle$ no sea la primera plaza ocupada por el contenedor con matrícula m.

7. `num_hileras`, `num_plazas`, `num_pisos`: imprimen el número de hileras, de plazas por hilera y de pisos por plaza, respectivamente.

8. `area_espera`: imprime la lista de contenedores en el área de espera, en el orden de salida de los contenedores: para cada contenedor se imprime $m(\ell)$ en una línea aparte, siendo m su matrícula y ℓ su longitud en plazas, por ejemplo

```
BAH0(1)
J19(2)
EXXO(1)
CZE(3)
```

9. `contenedores`: imprime la lista de todos los contenedores en la terminal en orden ascendente de matrícula. Para cada contenedor se imprime en una línea aparte primero su matrícula e inmediatamente, entre paréntesis y separados por coma, su ubicación y su longitud. Por ejemplo

```
ALTF5(<0,0,0>,1)
BAH0(<-1,0,0>,1)
CZE(<-1,0,0>,3)
EXXO(<-1,0,0>,1)
FAQS(<0,0,1>,3)
J19(<-1,0,0>,2)
KOPAK(<0,3,0>,1)
TRUST(<0,1,0>,2)
```

10. `area_almacenaje`: imprime el área de almacenaje de la terminal utilizando la letra inicial de la matrícula de cada contenedor en una representación bidimensional parecida a la que hemos empleado en los ejemplos del enunciado; por ejemplo, supongamos que hay tres contenedores: `AX1` de 1 plaza, `BHDE` de 2 plazas y `GTR201` de 3 plazas en una terminal definida con $N = 2$, $M = 4$ y $H = 3$, entonces el área de almacenaje se imprimiría así:

```
hilera 0
2
1    A
0   BB
  0123

hilera 1
2
1
0 GGG
  0123
```

Si el número de plazas por hilera es superior a 9, en la línea con números de plaza sólo se utiliza el último dígito. Por ejemplo, así se imprime una terminal con $N = 1$, $M = 15$ y $H = 2$:

```
hilera 0
1    A     C
0   BB    AAA  DD
  012345678901234
```

En este último ejemplo habría dos contenedores cuya matrícula comienza por A, uno de 1 plaza y otro de 3 plazas.

11. `huecos`: imprime una lista de los huecos de la terminal; cada hueco se especifica mediante la ubicación $\langle i, j, k\rangle$ en la que se inicia y su longitud o tamaño (el número de plazas que lo forman: 1, 2, 3, ..., M). La lista estará en el orden de menor a mayor tamaño, y a igual tamaño, por hilera y plaza. Recordad que hueco no es lo mismo que ubicación libre: todas las plazas que forman un hueco han de estar en el suelo de la terminal (piso 0) o han de tener contenedores debajo de ellas. Por ejemplo, para esta terminal

```
hilera 0
2
1    A
0   BB
  0123

hilera 1
2
1
0 GGG
  0123
```

la operación habrá de imprimir esta lista (sin los comentarios)

```
(<0,2,1>,1)  // el de encima de B
(<0,3,2>,1)  // el de encima de A
(<1,3,0>,1)  // el del suelo de la hilera 1
(<0,0,0>,2)  // el del suelo de la hilera 0
(<1,0,1>,3)  // el de encima de G
```

12. `fin`: termina la ejecución del programa