**INSTITUTO TECNOLÓGICO DE COSTA RICA I SEMESTRE 2018**

**INGENIERÍA EN COMPUTACIÓN**

**IC-4700 LENGUAJES DE PROGRAMACIÓN**

**TAREA PROGRAMADA 3 - EIFFEL**

El problema de empacado consiste en colocar objetos de diferentes tamaños dentro de cajas con una capacidad dada de almacenamiento, buscando minimizar el número de cajas requeridas. Por ejemplo si se tienen los siguientes objetos, representados por sus tamaños:  
 6, 7, 3, 2, 2, 2  
y se quieren almacenar en cajas de tamaño 11, se tiene la siguiente la siguiente distribución óptima en dos cajas:  
 [7,2,2] [6,3,2]

Resolver en forma óptima este problema es computacionalmente muy caro (NP-completo). Pero existen muchos algoritmos que aunque no son óptimos garantizan resultados muy buenos en la práctica. Se tienen los siguientes algoritmos:

**FFD (First Fit Decreasing)**: ordena los objetos por tamaño decreciente y busca colocar cada objeto en la primera caja disponible que tenga campo suficiente. Aplicando este algoritmo al caso anterior se tiene el siguiente proceso:

|  |  |  |
| --- | --- | --- |
| **Objeto** | **Cajas** | **Comentario** |
|  | [] | Secuencia: 6,7,3,2,2,2  Secuencia ordenada: 7,6,3,2,2,2 |
| 7 | [7] |  |
| 6 | [7] [6] | 6 no cabe en 1ª caja porque 7+6 > 11 |
| 3 | [7,3] [6] | 3 cabe en 1ª caja porque 7+3 <= 11 |
| 2 | [7,3] [6,2] | 2 no cabe en 1ª caja porque 7+3+2 > 11  sí cabe en la 2ª caja porque 6+2 <= 11 |
| 2 | [7,3] [6,2,2] | 2 no cabe en 1ª caja porque 7+3+2 > 11  sí cabe en la 2ª caja porque 6+2+2 <= 11 |
| 2 | [7,3] [6,2,2] [2] | 2 no cabe en las primeras dos cajas |

Como se aprecia, la solución no es óptima porque todo cabe en dos cajas. Sin embargo, se puede garantizar que las soluciones obtenidas por FFD no se desvían mucho de la solución óptima.

**FF (First Fit)**: no ordena los objetos por tamaño; simplemente los acomoda en el orden en que se presentan; busca la primera caja en donde pueda almacenarlos.

|  |  |  |
| --- | --- | --- |
| **Objeto** | **Cajas** | **Comentario** |
|  | [] | Secuencia: 6,7,3,2,2,2 |
| 6 | [6] |  |
| 7 | [6] [7] |  |
| 3 | [6,3] [7] | 3 cabe en 1ª caja porque 6+3 <= 11 |
| 2 | [6,3,2] [7] | 2 cabe en 1ª caja porque 7+3+2 <= 11 |
| 2 | [6,3,2] [7,2] | 2 no cabe en 1ª caja porque 6+3+2+2 > 11  sí cabe en la 2ª caja porque 7+2 <= 11 |
| 2 | [6,3,2] [7,2,2] | 2 cabe en la 2ª caja porque 7+2+2 <= 11 |

**BF (Best Fit)**: tampoco ordena los objetos por tamaño; para cada objeto escoge la caja que esté más llena y en la que aún queda campo para el objeto

|  |  |  |
| --- | --- | --- |
| **Objeto** | **Cajas** | **Comentario** |
|  | [] | Secuencia: 6,7,3,2,2,2 |
| 6 | [6] |  |
| 7 | [6] [7] |  |
| 3 | [6] [7,3] | se coloca 3 en la 2ª caja porque aunque cabe en ambas cajas, la 2ª está más llena |
| 2 | [6,2] [7,3] | 2 solo cabe en 1ª caja |
| 2 | [6,2,2] [7,3] | 2 solo cabe en 1ª caja |
| 2 | [6,2,2] [7,3] [2] | 2 no cabe en las primeras dos cajas |

La tarea consiste en desarrollar un programa en el lenguaje Eiffel para simular el desempeño de esos diferentes algoritmos. Se deben generar objetos cuyo tamaño esté dentro de un rango especificado, y colocarlos en cajas de un tamaño dado. A continuación se da una descripción más detallada de la simulación.

**Entradas**

El programa recibe como entradas los siguientes datos:

|  |  |
| --- | --- |
| **Parámetro** | **Significado** |
| tamaño de las cajas | * entero que especifica la capacidad de todas las cajas * valor default: 10 |
| tamaño máximo de los objetos | * los tamaños de los objetos tienen el rango: 1..tamaño\_max\_objetos * debe ser menor que el tamaño de las cajas * valor default: 7 |
| semilla | * semilla para la generación de una secuencia aleatoria de tamaños de objetos * valor default: 2018 |
| número de objetos | * número de objetos que serán generados para la simulación * valor default: 20 |

**Simulación**

Se debe generar una secuencia aleatoria de objetos cuyos tamaños cumplan con el rango de tamaños especificado. Luego se debe aplicar dicha secuencia a cada uno de los algoritmos descritos arriba: FFD, FF, BF; para así obtener la distribución de objetos en cajas realizada por cada algoritmo. Cada uno de los algoritmos debe modelarse como una clase aparte.

Tanto las cajas como los objetos generados deben modelarse como clases que incluyan al menos un identificador único y su tamaño. Para los objetos, el identificador es simplemente un entero consecutivo que se le asigna al generarlo. Dichos identificadores empiezan en el valor 1. Las cajas tienen también un identificador consecutivo que es independiente del de los objetos. Los identificadores de las cajas empiezan en el valor 1001.

**Salida**

La salida del programa consiste de los siguientes ítems.

* Datos generales (etiquetar los valores impresos)
  + tamaño de las cajas
  + tamaño máximo de los objetos
  + semilla
  + número de objetos generados
  + suma del tamaño de todos los objetos
* Listado de los objetos
  + Solo deben listarse un máximo de 40 objetos. Si hubieran más de 40 imprimir al final una indicación de que la lista está truncada ("...").
  + El listado debe venir separado en líneas con 10 objetos c/u.
  + Cada objeto debe mostrarse como un par ordenado (id,tamaño).
* Para cada uno de los algoritmos listar las cajas y su contenido.
  + Cada caja y su contenido debe mostrarse en una línea por aparte.
  + Mostrar un máximo de 10 objetos por caja; si hubiera más objetos en una caja, indicar usando "..." que no se mostraron todos sus objetos.
  + Se debe mostrar el id de cada caja y para cada objeto se debe mostrar el par (id, tamaño).
* Para cada uno de los algoritmos mostrar también las siguientes estadísticas:
  + número de cajas requeridas
  + promedio de ocupación =  
     suma\_tamaños\_todos\_objetos / (número\_cajas \* tamaño\_caja)

**Consideraciones adicionales**

Para la generación de números enteros aleatorios dentro de un rango de valores, consulte la clase [RANDOM](https://www.eiffel.org/article/random_numbers).

La implementación de BF se facilita mucho si se usan colas de prioridad. La clase [HEAP\_PRIORITY\_QUEUE](https://www.eiffel.org/doc/solutions/EiffelBase%2C%20Dispensers) ofrece la funcionalidad requerida.

El programa debe cumplir con los principios generales de la orientación a objetos:

* abstracción
* encapsulamiento
* modularidad
* jerarquía de herencia

Además, el programa debe incluir cláusulas **require** y **ensure** para cumplir con el diseño por contrato.

La tarea es individual y la fecha de entrega es el lunes 11 de junio a las 10pm.