Estrategias para la recolección de basura

(Garbage Collection strategies)

Extracto del material disponible en

http://www.site.uottawa.ca/ftppub/courses/Winter/seg3310/

Terminología

- **Heap:**
  - Es un área de memoria donde los datos pueden ser almacenados y removidos en cualquier orden.
  - Funciones como ‘malloc’ y ‘free’, o los operadores ‘new’ y ‘delete’ solicitan y liberan localidades de memoria en el heap
  - En la mayoría de los programas OO “buenos” todos los objetos son almacenados en el heap.

Terminología (cont.)

- **Conjunto raíz:**
  - Es un conjunto de objetos accesibles desde el programa
  - Ej: variables globales, o variables en el programa principal (almacenadas en la pila de ejecución del programa)

- ***Objeto heap* (también llamados celda o simplemente objeto):**
  - Es una pieza de datos asignada individualmente en el heap

- **Objeto *alcanzable*:**
  - Es un objetos que puede ser alcanzado transitivamente desde el conjunto raíz de objetos

Terminología(cont.)

- **Basura (Garbage):**
  - Son objectos *inalcanzables* desde los objetos del conjunto raíz pero que *no son libres*
- **Referencia colgada (Dangling reference):**
  - Es una referencia a un objeto que fue borrado
  - Puede causar el *crash* del sistema (con suerte!)
  - Puede causar *bugs* misteriosos
- **Aplicación:**
  - El programa usuario

Estrategias básicas para la recolección de basura

- **Cuenta de referencias**
- **Barrido de marcas**
- **Recogida de copias**

Cuenta de referencias

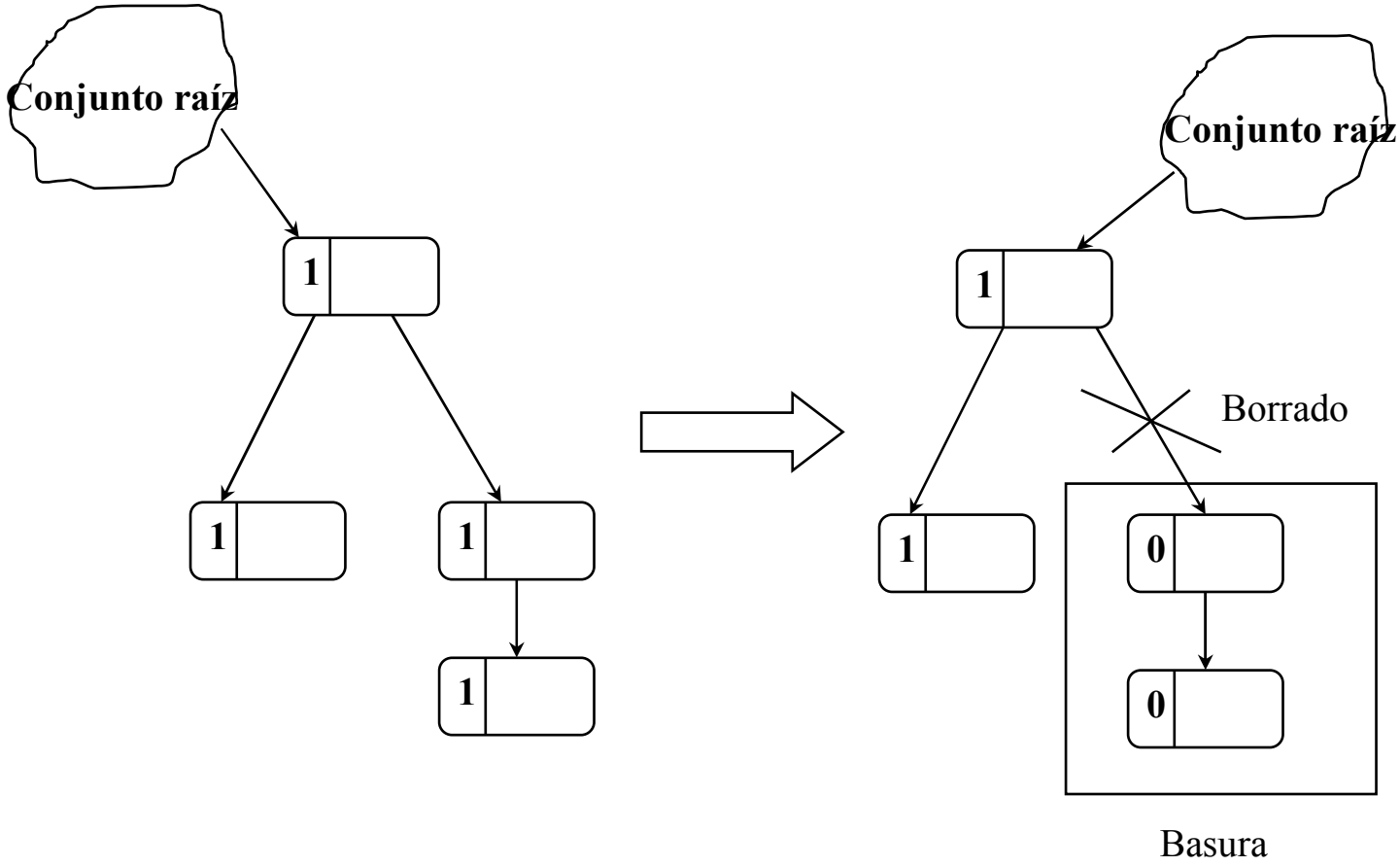
- **Cada objeto tienen una componente adicional llamada *contador de referencias (CR)***
- **Cuando un objeto O es creado, CR_O es inicializado en uno**
- **Cuando a cualquier objeto raíz le es asignada una referencia al objeto O, CR_O es incrementado**
- **Cuando una referencia al objeto O es borrada o se le asigna un nuevo valor, CR_O es decrementado**

Cuenta de referencias

- **Cualquier objeto con un CR igual a cero es considerado objeto basura y puede ser recolectado**

- **Cuando un objeto basura es recolectado,**
  - Se decrementa el CR de todos los objetos referenciados por él
  - La recolección de un objeto basura puede conducir a la recolección de otros objetos basura

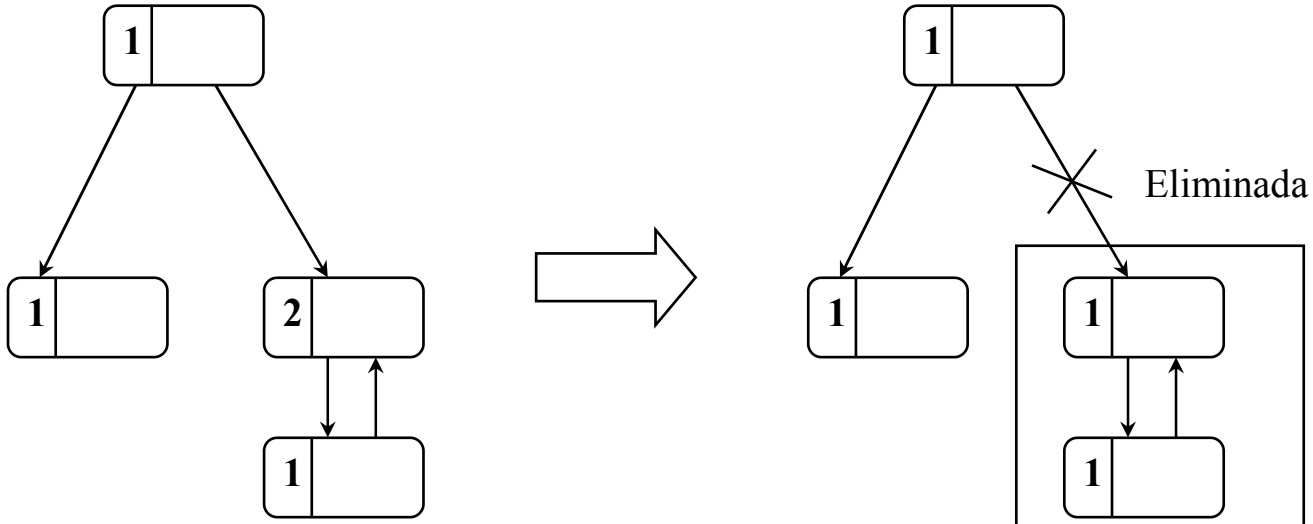
Ejemplo (cuenta de referencias)



Pros y Contras de la cuenta de referencias

- Pros:
  - El recolector de basura (GC) es ejecutado en paralelo con la aplicación (*on-the-fly* GC)
  - La memoria liberada es devuelta a la lista de celdas libres rápidamente
- Contras:
  - El CR de cada objeto debe ser actualizado cada vez que un puntero es cambiado
  - Se necesita una componente adicional para cada objeto (el CR)
  - Falla al devolver a la lista de celdas libres la basura con formato de cadena circular (próx. lámina)

Estructura de datos cíclica



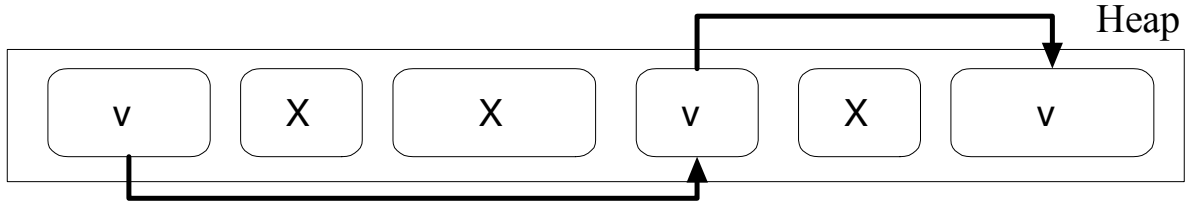
Basura pero no puede ser reclamada

Barrido de marcas (*Mark-Sweep algorithm*)

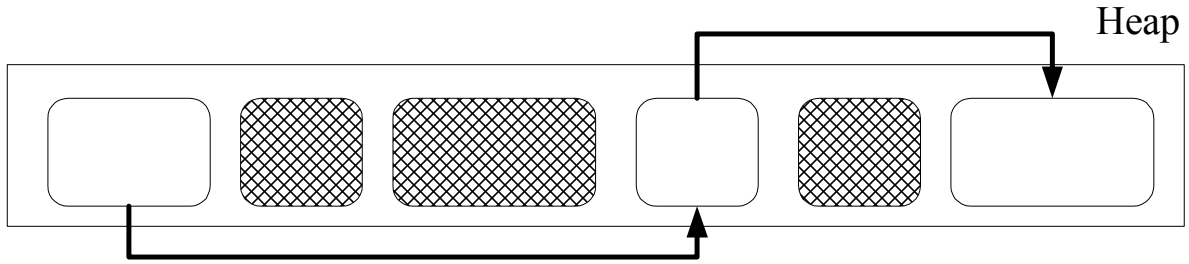
- **El algoritmo de barrido de marcas es invocado cuando la aplicación requiere memoria pero no hay suficiente espacio libre**
- **La aplicación es detenida mientras el barrido de marcas es ejecutado (*stop-and-collect* GC)**
  - Esto es impráctico para sistemas a tiempo real
- **Se realiza en dos fases:**
  - **Fase de marcación:** marca todos los objetos alcanzables
  - **Fase de barrido:** reclama los objetos basura

Ejemplo de barrido de marcas

- Fase de marcación



- Fase de barrido



Pros y Contras del barrido de marcas

- Pros:
  - Las estructuras de datos cíclicas pueden ser recuperadas
  - Tiende a ser más rápido que la cuenta de referencias

- Contras:
  - La aplicación debe detenerse mientras el algoritmo se está ejecutando (stop-and-collect GC)
  - Cada objeto alcanzable debe ser visitado en la fase de marcación y cada objeto en el heap debe ser visitado en la fase de barrido
  - Causa fragmentación de la memoria

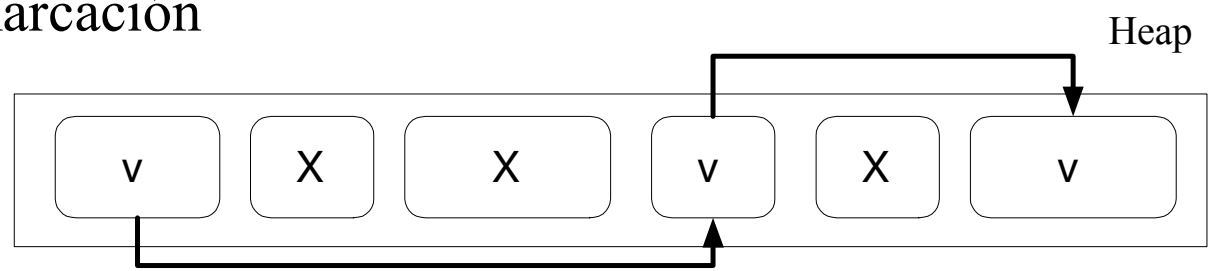
Barrido de marcas con compactación (*Mark-Compact algorithm*)

- **Similar al barrido de marcas pero no causa fragmentación de la memoria**

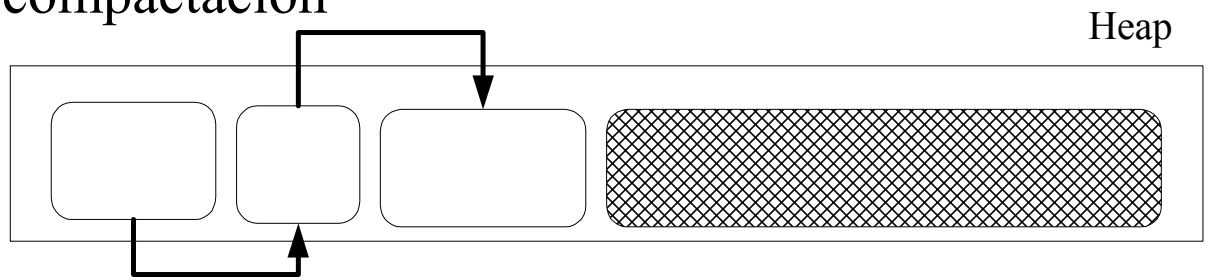
- **Dos fases:**
  - **Fase de marcación:** idéntica a la del barrido de marcas
  - **Fase de compactación:**
    - Los objetos marcados son compactados
    - Los objetos alcanzables son colocados en posiciones contiguas de memoria

Ejemplo

- Fase de marcación



- Fase de compactación



Pros y Contras del barrido con compactación

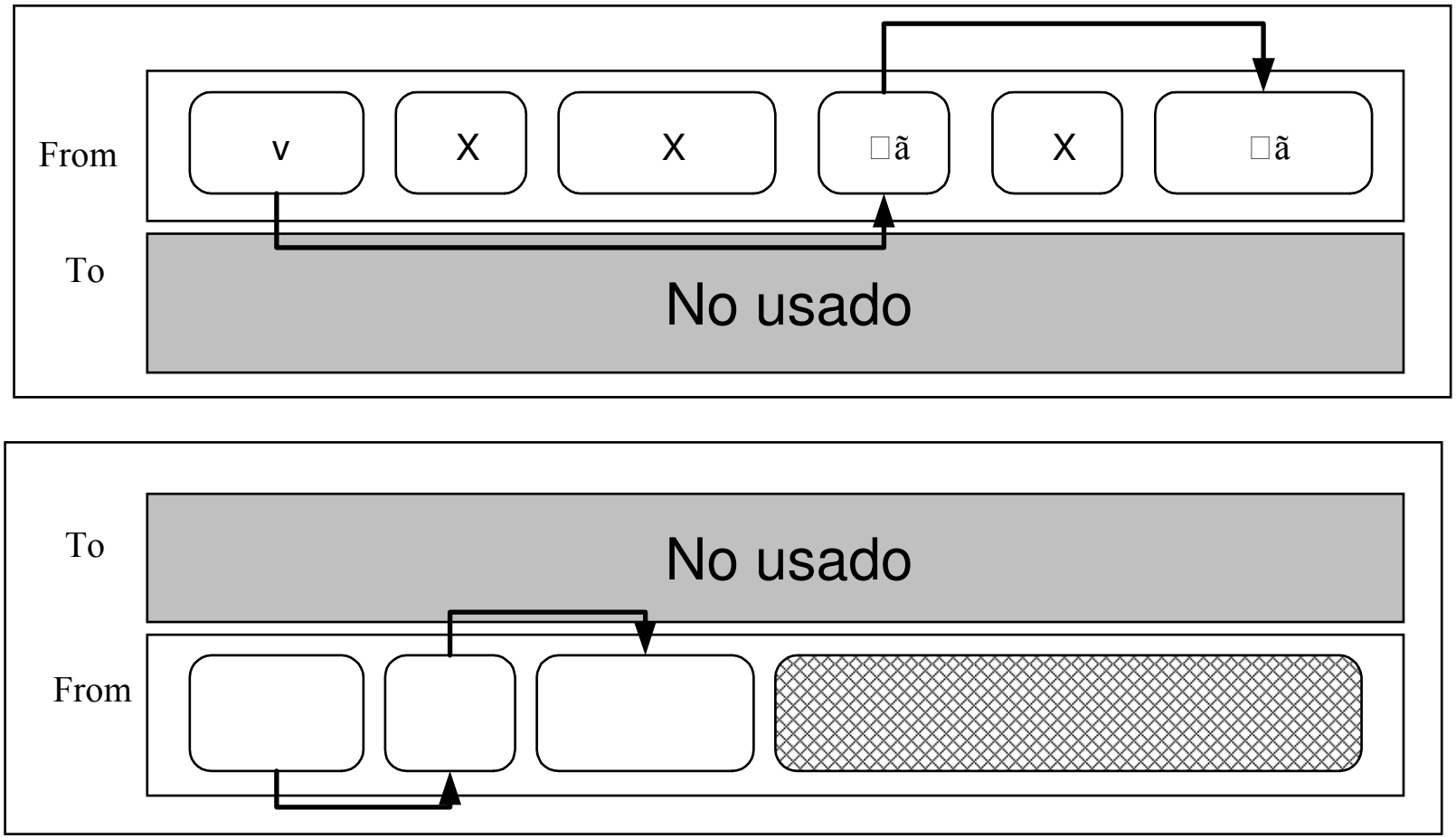
- Pros:
  - Elimina el problema de la fragmentación
- Contras:
  - Es un stop-and-collect GC
  - La implementación de la compactación requiere muchas pasadas sobre los objetos

Recogida de copias (Copying Algorithm)

- **La recogida de copias divide el heap en dos áreas iguales llamadas *from-space* y *to-space***
- **La aplicación trabaja en el área del *from-space***
- **El algoritmo visita los objetos alcanzables y los copia contiguamente en el área del *to-space***
  - Los objetos sólo necesitan ser recorridos una vez
- **Una vez que la copia es completada, el *to-space* y el *from space* invierten sus roles**

Ejemplo

-



Pros y Contras de la recogida de copias

- Pros:
  - Elimina la fragmentación
  - La copia es muy rápida
    - Siempre y cuando el porcentaje de objetos alcanzables es bajo
    - Sólo visita objetos alcanzables
- Contras:
  - Es un stop-and-collect GC
  - Sólo la mitad del heap está disponible de forma activa
  - Poco práctico para heaps muy grandes