Coletor de lixo (*garbage collector*)

Siang Wun Song - Universidade de São Paulo - IME/USP

MAC 5710 - Estruturas de Dados - 2008

Coletor de lixo

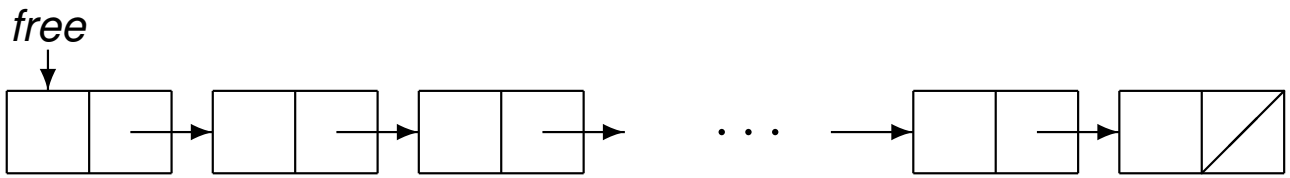
- Coletor de lixo é o nome de uma rotina básica de alguns sistemas (por exemplo, LISP, Java, sistemas operacionais baseados em listas de capacidades, etc.) para recuperar espaço de memória que havia sido alocado para uso por alguma aplicação mas posteriormente ficou em desuso pela aplicação, sem no entando ter sido retornado ao espaço disponível do sistema.
- É papel dessa rotina identificar e reaproveitar tais elementos de espaço.
- Para ilustrar melhor, vamos considerar um esquema de alocação dinâmica de espaço de memória, semelhante àquele visto em aula sobre a implementação de listas lineares.

Os campos de uma unidade ou elemento de espaço

Para facilitar a discussão vamos considerar que há uma unidade de espaco, ou elemento de espaço, assim definida:

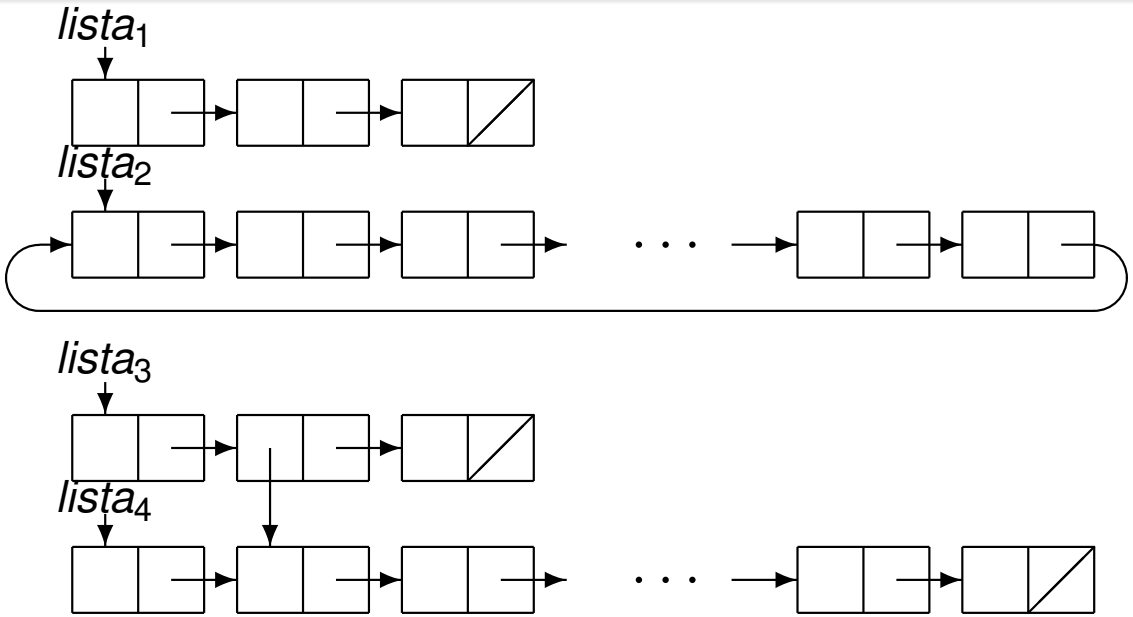
- Uma unidade ou elemento de espaço contém dois campos, cada um podendo conter ou uma informação literal ou um ponteiro ou referência.
- Em cada um dos dois campos, há um bit adicional para indicar se o campo contém informação literal ou se contém ponteiro.
- Além disso, para uso pelo coletor de lixo, cada unidade de espaço contém ainda ou um bit chamado *marca* ou um contador (inteiro), depdendo do método de coletor de lixo adotado (ver adiante).

Lista livre apontada por free



- O espaço das unidades ou dos elementos livres, num total de digamos *m* elementos, é organizado em uma lista ligada apontado por um ponteiro externo ao espaço chamado *free*.
- As aplicações podem solicitar elementos de espaço para seu uso e assim podem construir estruturas de dados com esses elementos retirados do espaço livre. As aplicações fazem acesso a essas estruturas de dados através de ponteiros externos conhecidos pela aplicação e também pelo sistema.

Ponteiros externos para acesso às estruturas criadas

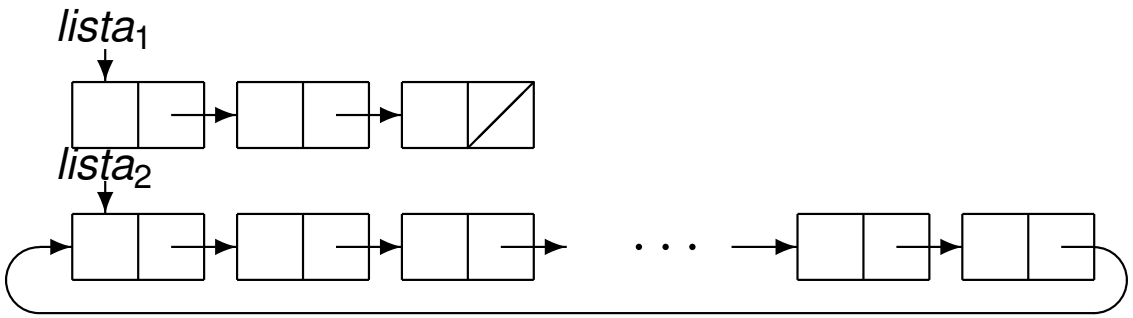


- No exemplo, os ponteiros externos $lista_1$, $lista_2$, $lista_3$ e $lista_4$ são usados para acesso às estruturas de dados construídas. Outro ponteiro externo conhecido é *free* que aponta para a lista dos elementos livres.
- Note na figura que um elemento dentro da estrutura de dado apontado por $lista_3$ pode apontar também para elemento da outra estrutura. Há assim compartilhamento de trechos da estrutura.

Unidade de espaço considerada lixo

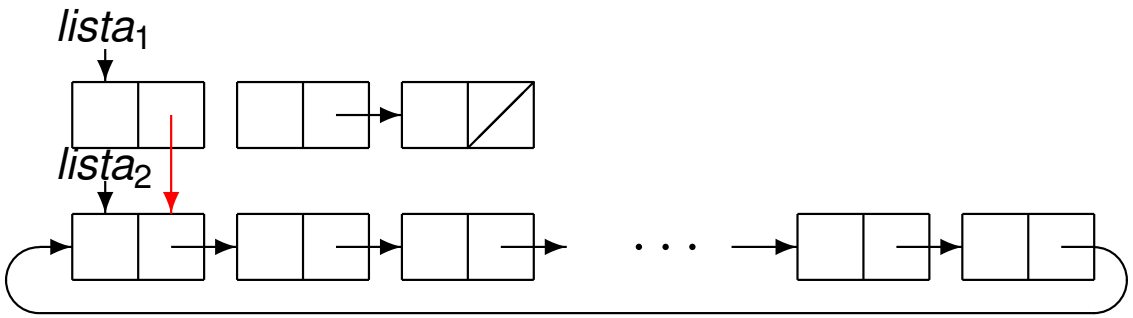
- Uma unidade de espaço é considerada ativa ou útil quando ela pode ser acessada ou atingida, partindo de qualquer ponteiro externo conhecido.
- Caso uma unidade de espaço não pode ser acessada dessa forma, então ela é definida como sendo inativa ou *lixo*.
- Unidades podem ser tornar lixo quando ponteiros são redirecionados de tal modo que certas unidades não são mais possíveis de serem acessadas a partir de nenhum ponteiro externo.

Exemplo do surgimento de lixo



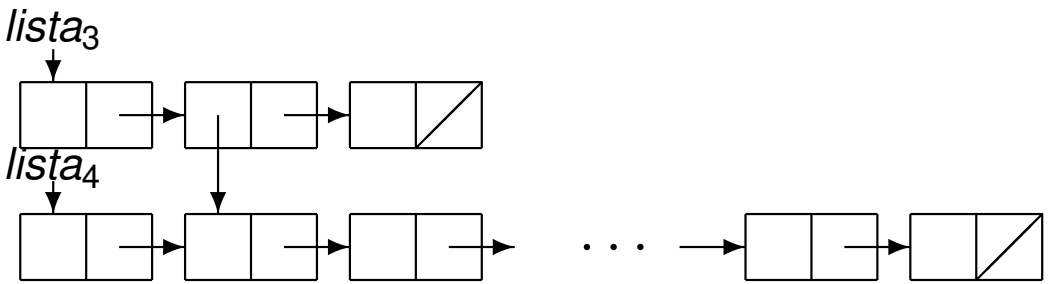
- Um ponteiro do primeiro elemento da $lista_1$ foi mudado de tal forma que a segunda unidade de espaço em diante não é mais acessível.

Exemplo do surgimento de lixo



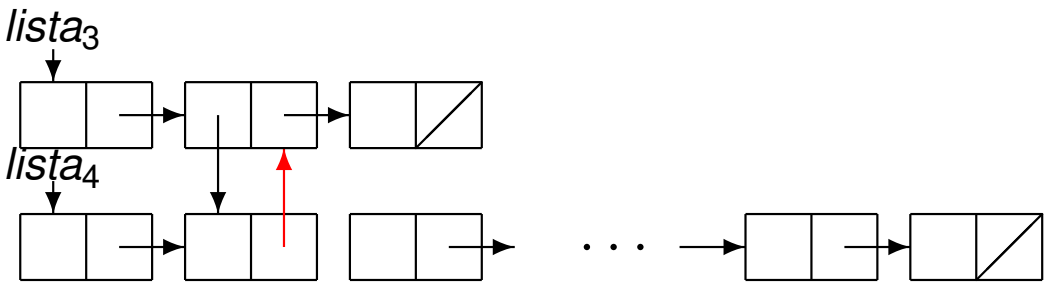
- Um ponteiro do primeiro elemento da $lista_1$ foi mudado de tal forma que a segunda unidade de espaço em diante não é mais acessível.

Exemplo do surgimento de lixo



- Do mesmo modo, um ponteiro na estrutura de dado da $lista_4$ foi alterado de tal forma que parte da sua estrutura orignal não mais possa ser acessado, gerando portanto várias unidades lixo.

Exemplo do surgimento de lixo



- Do mesmo modo, um ponteiro na estrutura de dado da $lista_4$ foi alterado de tal forma que parte da sua estrutura orignal não mais possa ser acessado, gerando portanto várias unidades lixo.

Coletor de lixo

É função do coletor de lixo identificar e recolher ou recuperar os elementos lixo, retornando-os para o espaço dos elementos livres ou disponíveis. Vermos dois esquemas de coletor de lixo.

1. Método marca-varre (*mark-scan*): esse método é geral e funciona sempre, isto é, todo lixo é sempre identificado e recolhido. Usa um bit adicional por elemento para servir de marca (a ser visto a seguir).
2. Método contador de referências: esse método pode não recolher certos tipos de lixo, mas tem a vantagem de identificar o lixo e procede seu recolhimento assim que o lixo surgir. Usa um contador (inteiro) por elemento para servir de contagem de referências (a ser visto a seguir).

Método marca-varre

O método marca-varre (*mark-scan*) é um método bem geral que funciona sempre, isto é, todo lixo é identificado e recolhido. Usa um bit adicional por elemento para servir de marca. O bit marca = 0 signfica elemento não marcado e o bit marca = 1 significa o elemento marcado.

O método consiste de duas fases: uma fase de marcação seguida de uma fase de varredura e recolhimento.

Fase de marcação (*mark*)

- A partir de ponteiros externos conhecidos, todos os elementos das estruturas de dados acessíveis são marcados.
- Para isso é necessário algum algoritmo para percorrer essas estruturas, com a utilização de uma pilha ou uma fila para ajudar o percurso e a marcação.
- É necessário distinguir em cada elemento o campo que contém ponteiro ou que simplesmente contém um valor literal.

Fase de varredura (*scan*)

O espaço total das m posições em que estão implementadas as estruturas de dados das aplicações, bem como a lista dos elementos livres, é varrido ou percorrido sequencialmente, começando com a posição 0 até a posição $m - 1$, examinando cada elemento uma e exatamente uma vez. Quando cada elemento é examinado, é tomada uma das duas ações:

- Se o elemento é marcado, simplesmente desmarca esse elemento e nada mais é feito com o elemento.
- Se o elemento não é marcado, então por definição ele é lixo e é recolhido para fazer parte do espaço disponível.

Esse método tem o custo de gastar um bit (para marca) por elemento. Todo lixo é sempre identificado e recolhido. Entretanto as duas fases podem levar tempo considerável durante o qual todas as aplicações ficam suspensas. Essa espera grande pode ser inadequada.

Método contador de referências

Esse método tem a vantagem de identificar o lixo e procede seu recolhimento assim que o lixo surgir.

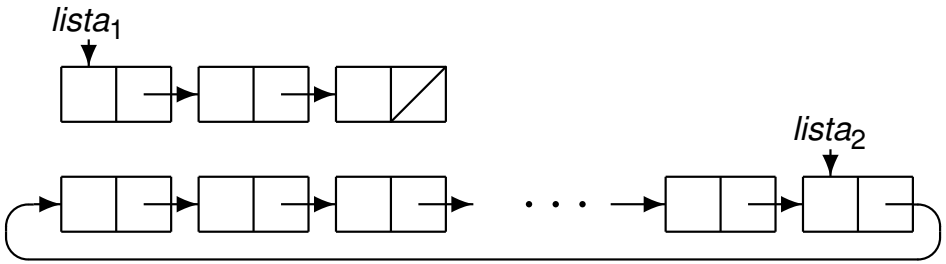
- Usa um contador (inteiro) por elemento para servir de contagem de referências.
- O contador de referências de um elemento é um número inteiro que indica a quantidade de ponteiros ou referências a esse elemento.
- Ele deve estar sempre mantido atualizado. Isto é, sempre que um ponteiro a um elemento *A* é retirado e passa a ser redirecionado para apontar para um outro elemento *B*, entäo o contador de referências de *A* deve ser diminuído de 1, ao passo que o contador de referências de *B* deve ser acrescido de 1.

Como identificar elementos que se tornam lixo

- Quando o contador de referências de um elemento atinge o valor zero, o elemento é lixo por definição, pois nenhum outro elemento estaria apontando a ele.
- O elemento é então recolhido ao espaço livre. Caso esse elemento aponte para outros elementos, esses outros elementos apontados devem ter seus contadores de referências atualizados, podendo resultar em lixo também caso esses contadores atingem zero com a atualização.
- Com isso o lixo pode ser identificado e recolhido assim que ele surgir.

Problemas do uso de contador de referências

- Listas circulares: mesmo quando toda uma lista circular se tornar inacessível, os contadores dos elementos dessa lista contêm no mínimo o valor 1 e assim os elementos não seriam identificados como lixo.

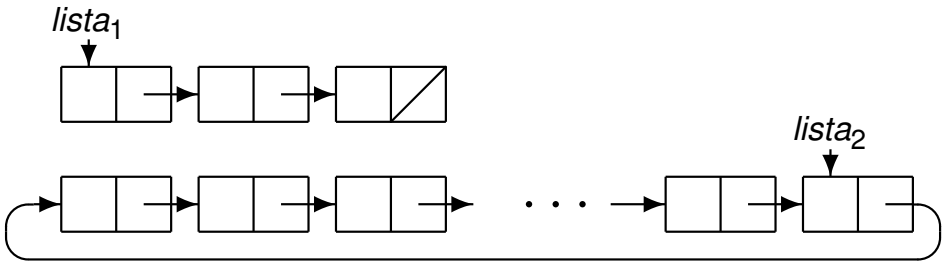


Sejam $lista_1$ e $lista_2$ apontadores externos.

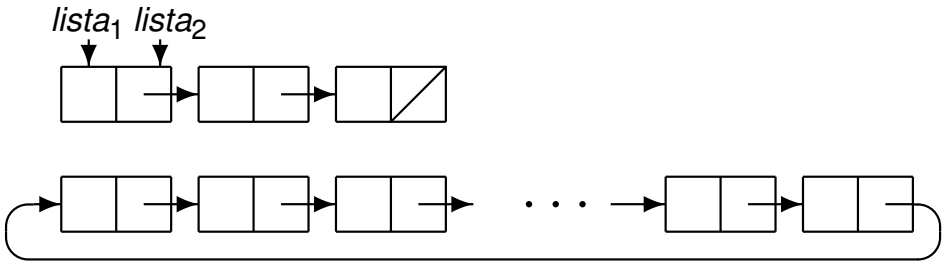
A execução do comando $lista_2 \leftarrow lista_1$ torna a lista circular (originalmente apontada por $lista_2$) inacessível. Mas o método de contador de referências não é capaz de detectar isso.

Problemas do uso de contador de referências

- Listas circulares: mesmo quando toda uma lista circular se tornar inacessível, os contadores dos elementos dessa lista contêm no mínimo o valor 1 e assim os elementos não seriam identificados como lixo.



Sejam $lista_1$ e $lista_2$ apontadores externos.



A execução do comando $lista_2 \leftarrow lista_1$ torna a lista circular (originalmente apontada por $lista_2$) inacessível. Mas o método de contador de referências não é capaz de detectar isso.

Problemas do uso de contador de referências

A outra situação problemática é a seguinte:

- Capacidade do contador excedida: o contador é um número inteiro e sua capacidade máxima pode ser atingida. A partir desse momento não é mais possível contabilizar o número real de referências ao elemento e o mesmo nunca mais poderá ser recolhido.

Um esquema que usa contador de referências pode ter seu espaço livre diminuído com o tempo, devido às duas situações acima.

Nesse caso, uma solução é utilizar, periodicamente, o método 1 (marca-varre), que será capaz de recuperar mesmo os elementos das duas situações acima mencionadas.