Disciplina Estruturas de Dados

Coletor de lixo (*garbage collector*)

Siang - 2004

Introdução

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, parecido com aquele visto em aula (seção 4 da apostila Notas de Aula - Listas Lineares).

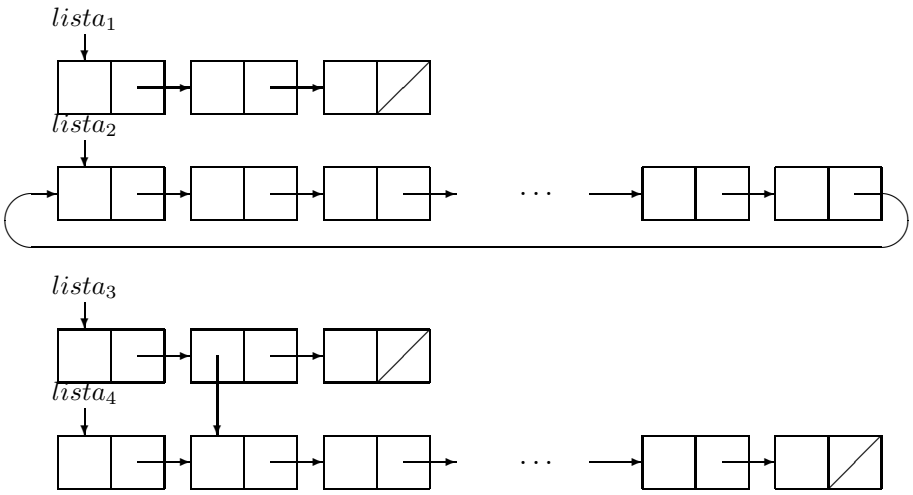

Figura 1: Ponteiros externos $lista_1$, $lista_2$, $lista_3$ e $lista_4$.

Seja um elemento de espaço constituído de dois campos, cada um podendo conter ou uma informação (literal) ou um ponteiro ou referência. Podemos usar um bit adicional em um dos 2 campos para indicar se o campo contém informação literal ou se contém ponteiro. Além disso, para uso pelo coletor de lixo, cada elemento poderá conter ainda um bit chamado *marca* ou um contador (inteiro), depdendo do método de coleção de lixo adotado (ver adiante).

O espaço dos elementos livres ou disponíveis, num total de digamos m elementos, é organizado em uma lista ligada apontado por um ponteiro externo ao espaço chamado *avail*. 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. Na figura 1, os ponteiros externos usados para acesso às estruturas de dados construídas são os ponteiros denominados $lista_1$, $lista_2$, $lista_3$ e $lista_4$. Note, como mostra a figura 1, que um elemento dentro da estrutura de dado apontado por $lista_3$ pode apontar também para elemento da outra estrutura.

Elemento lixo

Um elemento é considerado ativo ou útil quando ele pode ser acessado ou atingido, partindo de qualquer ponteiro externo conhecido. Caso um elemento não pode ser acessado dessa forma, então ele é definido como sendo inativo ou *lixo*. Elementos lixo podem surgir quando ponteiros são redirecionados de tal modo que certos elementos não são mais possíveis de serem acessados a partir de nenhum ponteiro externo.

Na figura 2, um ponteiro da $lista_1$ foi mudado de tal forma que o segundo elemento em diante não é mais acessível. 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.

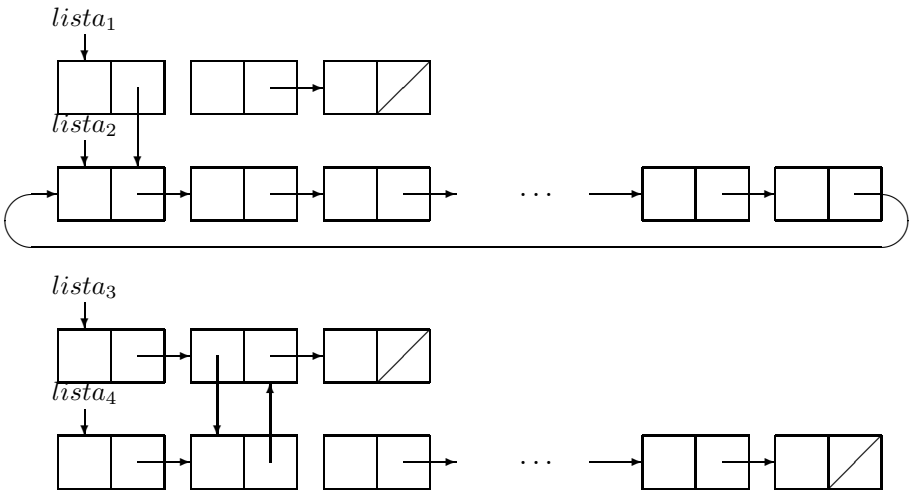

Figura 2: Certos elementos ficam inacessíveis e tornam-se lixo.

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 coleção de lixo.

1. Método marca-varre (*mark-scan*): esse método é geral e funciona sempre, isto é, todo lixo é sempre identificao 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 sempre identificao 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: 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 a marcação. É necessário distinguir em cada elemento o campo que contém ponteiro ou que simplesmente contém um valor literal (portanto não deve ser entendido como ponteiro).

Fase de varredura e recolhimento: O espaço total das m posições 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

O método contador de referências 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.

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.

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.

Infelizmente há duas situações em que o lixo pode não ser identificado.

- Listas circulares: mesmo quando toda uma lista circular se tornar inacessível (por nenhum ponteiro externo), seus contadores contém no mínimo o valor de 1 e assim não seria identificado como lixo.
- 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. Portanto uma vez atingida a capacidade máxima no contador, o elemento 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 o método 1 (marca-varre), que será capaz de recuperar mesmo os elementos das duas situações acima mencionadas.