Dominando Estruturas de Dados 1

Professor: Dr. Samuel Botter Martins

Lista Auto-organizável por Contagem

1. Descrição

Em uma lista encadeada, não há uma forma simples que facilite a recuperação dos registros armazenados nos nós: para recuperar o nó na posição **i** de uma lista é preciso percorrê-la a partir da cabeça, fazendo i acessos a nós. Em muitas aplicações, as **frequências** com que os registros são acessados **não são uniformes**. Faz sentido que os registros que são recuperados com maior frequência sejam colocados mais próximos da cabeça. Entretanto, tais frequências de acesso não são conhecidas previamente e mudam ao longo do tempo, à medida em que a lista é acessada.

Estratégias de permutação têm sido aplicadas para melhorar o número de acessos para recuperar registros em uma lista. Tais estratégias movem o registro que acabou de ser *buscado* um certo número de posições em direção ao início da lista, sem modificar a ordem relativa dos demais registros. Listas acompanhadas de alguma estratégia desse tipo foram chamadas de **listas auto-organizáveis.**

Uma das estratégias de permutação confia na **contagem** dos elementos acessados. Cada registro tem um contador do número de acessos. Quando um registro é recuperado, o contador é incrementado e ele é movido para *uma posição anterior* a todos os registros *com contador menor ou igual ao dele*.

Por exemplo, suponha que a lista L tenha registros com chaves únicas **(1,2,3,4,5)** nesta ordem e suponha que a sequência de requisições para recuperar registros seja *(4,2,2,4,3,1,3)*. Abaixo aparecem as **modificações** na lista e os **custos** para cada estratégia. O **custo** é medido como a soma do número de *nós visitados* para recuperar o registro com a chave requisitada, *sem contar as operações realizadas na reorganização da lista*. Se a chave não existir na lista, simplesmente ignore tal custo de busca.

- Lista inicial **L = (1, 2, 3, 4, 5)**.
- Contador **C = (0, 0, 0, 0, 0)**

- Requisição = 4**.**
  - Custo = 4.
  - Lista L = (4, 1, 2, 3, 5).
  - Contador C = (1, 0, 0, 0, 0)
- Requisição = 2.
  - Custo = 3.
  - Lista L = (2, 4, 1, 3, 5).
  - Contador C = (1, 1, 0, 0, 0)
- Requisição = 2.
  - Custo = 1.
  - Lista L = (2, 4, 1, 3, 5).
  - Contador C = (2, 1, 0, 0, 0)
- Requisição = 4.
  - Custo = 2.
  - Lista L = (4, 2, 1, 3, 5).

    - Contador C = (2, 2, 0, 0, 0)
- Requisição = 3.
    - Custo = 4.
    - Lista L = (4, 2, 3, 1, 5).
    - Contador C = (2, 2, 1, 0, 0)
- Requisição = 1.
    - Custo = 4.
    - Lista L = (4, 2, 1, 3, 5).
    - Contador C = (2, 2, 1, 1, 0)
- Requisição = 3.
    - Custo = 4.
    - Lista L = (3, 4, 2, 1, 5).
    - Contador C = (2, 2, 2, 1, 0)

- Custo total = 4 + 3 + 1 + 2 + 4 + 4 + 4 = 22.

Neste laboratório, seu objetivo é implementar essa **lista encadeada auto-organizável** por contagem. Cada nó da lista guardará um Produto, que possui número de série único (inteiro), nome e preço. O acesso aos produtos será feito pelo seu número de série.

2. Especificação da Entrada e Saída

Entrada

A entrada consiste de uma série de comandos terminados pelo comando **para.** Os possíveis comandos são:

- **add numero_serie nome preco**
    - Adiciona um produto no final da lista, com contagem zerada.
    - O produto possui:
        - **numero_serie**: inteiro;
        - **nome:** string sem espaços e acentos; e
        - **preco:** float.
- **acessa numero_serie**
    - Acessa o produto com número de série igual a **numero_serie,** computando seu custo de acesso e atualizando sua posição na lista ligada;
    - Os dados do produto acessado deverão ser impressos (veja a seção *Saída*);
    - Caso não exista um produto com o número de série informado, o programa deverá imprimir uma mensagem informando isso (veja a seção *Saída*).

- **para**

- Imprime os números de série da lista encadeada, na ordem que eles se encontram, e termina a execução do programa.

Saída

Ao acessar um produto que se encontra na lista encadeada, o programa deverá imprimir os dados de tal produto da seguinte forma. P. ex: *acessa 999:*

999, Playstation5, 4999.99

Caso o produto não exista, o programa deverá imprimir:

Produto 999 inexistente

Considere 2 casas decimais na impressão do preço do produto.

Por fim, assim que o comando **para** for lido, seu programa deverá imprimir duas linhas:

- Na primeira linha, os números de série dos produtos na ordem final da lista ligada; e
- Na segunda linha: o custo total de acessos (soma dos custos de todos os acessos feitos).

Veja os exemplos abaixo para saber como imprimir tais informações.

Exemplos

Entrada	**Saída**
add 1 Playstation5 4999.99 add 2 Lapiseira 5.00 add 3 BolaDeFutebol 89.90 add 4 MouseGamer 120.00 add 5 Chiclete 0.99 acessa 4 acessa 2 acessa 2 acessa 4 acessa 3 acessa 1 acessa 3 para	4, MouseGamer, 120.00 2, Lapiseira, 5.00 2, Lapiseira, 5.00 4, MouseGamer, 120.00 3, BolaDeFutebol, 89.90 1, Playstation5, 4999.99 3, BolaDeFutebol, 89.90 L = (3, 4, 2, 1, 5) Custo total = 22

Entrada	**Saída**

add 1 Playstation5 4999.99
add 2 Lapiseira 5.00
add 3 BolaDeFutebol 89.90
acessa 99
acessa 6
add 4 MouseGamer 120.00
add 5 Chiclete 0.99
acessa 4
acessa 2
add 6 Chupeta 31.47
acessa 100
acessa 6
acessa 1
acessa 3
add 7 Game_Fifa_2021 249.99
acessa 1
acessa 7
para

Produto 99 inexistente
Produto 6 inexistente
4, MouseGamer, 120.00
2, Lapiseira, 5.00
Produto 100 inexistente
6, Chupeta, 31.47
1, Playstation5, 4999.99
3, BolaDeFutebol, 89.90
1, Playstation5, 4999.99
7, Game_Fifa_2021, 249.99

L = (1, 7, 3, 6, 2, 4, 5)
Custo total = 31

3. Dicas

- Para **compilar** seu código no terminal:
  - ***gcc lab.c -o lab***
- **-o** significa *output*. Ele é responsável por gerar o binário do seu programa para execução. É **OBRIGATÓRIO** que o arquivo tenha a função **main;**
- Logo, o que você está dizendo é: "*compile o código **lab.c** com o compilador **gcc**, gerando o executável (saída) **lab**";*

- Para **executar** seu programa:
  - ***./lab***

- Você pode baixar os arquivos de casos de teste do run.codes e executá-los manualmente:
  - ***./lab < 01.in***
- A diretiva **<** redireciona o conteúdo do arquivo *01.in* para o terminal, cujas entradas/dados serão lidas pelo ***scanf***;
- Você pode ainda redirecionar a *saída* impressa no terminal para um arquivo:
  - ***./lab < 01.in > 01.res***
- Por fim, você poder comparar sua reposta com o gabarito (resultado do caso de teste), fazendo
  - ***diff 01.res 01.out***
    - onde *01.out* é a saída esperada para a entrada *01.in*