sqtpm
[256352]

voltar

Trabalho: 06-auto-organizavel
Linguagens: C
Data de abertura: 2019/09/19 10:00:00
Data limite para envio: 2019/09/25 18:00:00 (encerrado)

Último envio: 100% em 2019/09/25 14:44:29
Envios: 3

Lista auto-organizável

Em uma lista 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 freqüências com que os registros são recuperados não são uniformes. Faz sentido que os registros que são recuperados com maior freqüência sejam colocados mais próximos da cabeça, mas tipicamente tais freqüências não são conhecidas e mudam ao longo do tempo.

Estratégias de permutação podem ser aplicadas para reduzir o número de acessos a nós da lista durante a recuperação de registros em uma lista. Tais estratégias movem o registro que acabou de ser recuperado 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*.

Algumas estratégias de permutação foram propostas na literatura. As mais usadas incluem:

- Move-to-front (MTF): quando um registro é recuperado ele é movido para o início da lista, se ele ainda não estiver no início da lista.
- Transpose (TR): quando um registro é recuperado ele é trocado de posição com o registro que o precede, se ele ainda não estiver no início da lista.
- Count (C): 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 `(1,2,3,4,5)` nesta ordem e suponha que a seqüência 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.

sqtpm
[256352]

voltar

Move-to-front

- Lista inicial L=(1,2,3,4,5)
- Requisição = 4. Custo = 4. Lista L=(4,1,2,3,5)
- Requisição = 2. Custo = 3. Lista L=(2,4,1,3,5)
- Requisição = 2. Custo = 1. Lista L=(2,4,1,3,5)
- Requisição = 4. Custo = 2. Lista L=(4,2,1,3,5)
- Requisição = 3. Custo = 4. Lista L=(3,4,2,1,5)
- Requisição = 1. Custo = 4. Lista L=(1,3,4,2,5)
- Requisição = 3. Custo = 2. Lista L=(3,1,4,2,5)

Custo total = 4+3+1+2+4+4+2 = 20.

Transpose

- Lista inicial L=(1,2,3,4,5)
- Requisição = 4. Custo = 4. Lista L=(1,2,4,3,5)
- Requisição = 2. Custo = 2. Lista L=(2,1,4,3,5)
- Requisição = 2. Custo = 1. Lista L=(2,1,4,3,5)
- Requisição = 4. Custo = 3. Lista L=(2,4,1,3,5)
- Requisição = 3. Custo = 4. Lista L=(2,4,3,1,5)
- Requisição = 1. Custo = 4. Lista L=(2,4,1,3,5)
- Requisição = 3. Custo = 4. Lista L=(2,4,3,1,5)

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

Count

- 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.

Uma outra estratégia é a move-ahead-k, que move um registro k posições em direção à cabeça depois que ele é acessado. k pode ser definido como um número fixo, como um percentual da distância até a cabeça ou como outra função de distância. Algumas outras estratégias usam combinações dessas que foram listadas.

Neste trabalho as estratégias MTF, TR e C devem ser comparadas. Seu programa deve usar uma lista encadeada. (Em um vetor uma estratégia como essas seria muito custosa porque seriam necessárias muitas

movimentações dos dados.)

sqtpm
[256352]

voltar

Entrada

A entrada para o programa são o número inteiro N de chaves na lista, entre 1 e 1000, o número inteiro R de requisições e uma seqüência de R requisições inteiras no intervalo [1,N]. Cada um dos acessos em R deve ser realizado em ordem, para MTF, TR e C, sempre a partir da lista que é iniciada contendo as chaves na ordem 1,2,3,...,N.

Saída

A saída são três inteiros indicando os custos das estratégias MTF, TR e C, respectivamente.

Exemplo

Entrada:

```
5
7
4 2 2 4 3 1 3
```

Saída:

```
20 22 22
```

Observações

- Não deixe de liberar todos os nós da lista ao terminar o processamento.
- É interessante organizar o programa em três arquivos, um .h com as declarações de tipos e funções que manipulam a lista, um .c com as implementações das funções da lista e um outro .c com a função main e outras funções não relacionadas com a lista.
- A estratégia count vai fazer com que os registros fiquem em ordem não-crescente de contadores. Isso permite que a movimentação seja implementada fazendo apenas uma passada pela lista, ao invés de duas como pode parecer necessário à primeira vista. Depois de fazer seu programa funcionar com duas passadas, um exercício interessante é implementar com apenas uma.
