MC458A - Laboratório 1: Problema de Acesso à Lista

1 Introdução

O **Problema de Acesso à Lista** é considerado um problema clássico da área de **algoritmos online** e **análise competitiva**. Apesar de sua aparente simplicidade, este problema é conveniente para modelar (versões simplificadas) de várias situações práticas. Veja `https://en.wikipedia.org/wiki/List_update_problem`.

Neste problema temos uma lista ligada de itens e um algoritmo online que recebe uma sequência de **requisições** por itens e este deve **atender/servir** cada requisição. Para servir uma requisição x, o algoritmo deve percorrer a lista ligada a partir do seu início e procurar pelo item x para acessá-lo. O custo para acessar um item que está na i-ésima posição da lista é i; se o item não estiver na lista, o custo é $n + 1$ onde n é o número de itens da lista. O objetivo é **minimizar** o custo total das requisições. O algoritmo pode reordenar os itens na lista a qualquer momento, mas isto resulta num custo dependendo de quais itens forem deslocados. Os algoritmos que veremos usam **free transposition** que consiste em alterar a posição de um item que foi acessado ao procurar o item x **(sem custo adicional)**.

Do ponto de vista computacional, há dois tipos de modelos clássicos bem conhecidos: **offline** e **online**. No modelo offline o algoritmo recebe a sequência inteira de requisições de uma só vez e pode processá-la como quiser; sabe-se que o problema de minimizar o custo total é NP-difícil. No modelo online o algoritmo recebe cada requisição por vez e deve servi-la imediatamente sem conhecimento das requisições futuras. Neste trabalho trataremos de algoritmos para o modelo online.

2 Competitividade

Em **análise competitiva** costuma-se medir a qualidade de um algoritmo online comparando-o com uma "solução ótima" construída por um **adversário**. Um adversário também é um algoritmo online: ele recebe as requisições uma por vez e deve servi-las imediatamente; a diferença com um algoritmo online usual é que ele conhece as requisições futuras, o que o torna ridiculamente poderoso. Dizemos que um algoritmo online é **c-competitivo** se para qualquer instância (sequência de requisições) o custo da solução obtida pelo algoritmo é no máximo c vezes o custo de uma solução ótima construída pelo adversário. Este tipo de adversário é chamado **oblivious adversary** (há outros tipos).

3 Algoritmos para o problema de acesso à lista

Apresentamos quatro algoritmos descrevendo o que cada um faz em cada requisição:

MTF (Move To Front): após acessar o item requisitado, ele é movido para o início da lista sem alterar a posição relativa dos outros itens.

TRANS (Transposition): após acessar o item requisitado, ele é trocado de lugar com o item anterior sem alterar a posição relativa dos outros itens. Equivalentemente, ele é movido para a posição anterior.

FC (Frequency Count): inicialmente todos os itens são inicializados com frequência zero. Sempre que um item é requisitado, sua frequência é incrementada de uma unidade e a lista é **reordenada em ordem decrescente de frequência**. Ou seja, a frequência de um item é o número de vezes que ele foi requisitado/acessado. Para efeito deste trabalho a reordenação deve ser feita da seguinte forma: após atualizar a frequência do item x acessado, este é movido para a posição da lista em que antecede todos os itens que têm a mesma frequência dele sem alterar a posição relativa dos outros itens. Ou seja, se a frequência de x é k ele deve ser o primeiro item da lista reordenada com frequência igual a k.

BIT (versão aleatória de MTF): inicialmente cada item x é associado com um bit aleatório $b(x)$. Ao acessar um item x, se $b(x) = 1$ então o item é movido para o início da lista, senão a lista não é alterada. Em ambos os casos o bit $b(x)$ é complementado (1 vira 0 ou 0 vira 1).

Sabe-se que TRANS e FC não são c-competitivos para nenhum c. Pode-se mostrar que MTF é 2-competitivo (isto é o melhor possível para um algoritmo online determinístico) e BIT é 1.75-competitivo (algoritmos aleatórios em geral são melhores em modelos online).

Como curiosidade, há outros modelos do problema em que inserções e remoções são permitidas. Os resultados teóricos são similares.

4 Especificação de entrada e saída

Infelizmente não é muito prático comparar com um adversário, pois o problema offline é NP-dfícil. Assim, o trabalho consiste em simular os algoritmos MTF, TRANS, FC e BIT.

Observação: Note que em princípio não é necessário implementar uma lista ligada. Seu programa precisa apenas ser capaz de computar o custo de uma sequência de requisições.

A entrada é composta por 5 linhas:

- A primeira linha contém um inteiro N ($1 \leq N \leq 1000$) que representa o número total de itens.
- A segunda linha contém N inteiros no intervalo $[1 \ldots N]$, separados por um espaço em branco, que representam a ordem inicial da lista de itens.
- A terceira linha contém N bits (inteiro 0 ou 1), separados por um espaço em branco, que representam a inicialização do algoritmo BIT. O i-ésimo valor corresponde ao bit associado ao elemento da posição i da lista de itens.
- A quarta linha contém um inteiro K ($1 \leq K \leq 2500$) que representa o número total de requisições.
- A quinta linha contém K inteiros, separados por um espaço em branco. Cada inteiro x corresponde a uma requisição por um item de valor x.

A saída de seu programa deverá ser **quatro linhas** que correspondem ao custo total dos algoritmos MTF, TRANS, FC e BIT, respectivamente (há um `\n` após a última linha).

Exemplo:

Entrada	Saída
10 9 8 1 4 2 3 5 7 6 10 0 1 0 0 1 0 0 1 1 1 5 2 2 1 15 4	26 29 26 26

Entrada	Saída
8 7 8 1 4 2 3 5 6 0 1 0 0 0 0 0 1 5 1 2 1 2 10	21 23 21 25

5 Implementação e Submissão

- A solução deverá ser implementada em C, C++, Pascal ou Python 2/Python 3. Só é permitido o uso de bibliotecas padrão.
- O programa deve ser submetido no SuSy, com o nome principal **t1** (por exemplo, t1.c).
- O número máximo de submissões é 20.
- A tarefa contém 10 testes abertos e 10 testes fechados. A nota será proporcional ao número de acertos nos testes fechados.

A solução pode ser submetida até o dia 06/09/21.