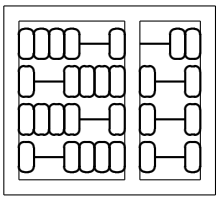

MC202 – Laboratório 03
TRANSFORMAÇÃO MTF (*Move To Front*)
INSTITUTO DE COMPUTAÇÃO — UNICAMP
Prof.: Anderson Rocha
anderson.rocha@ic.unicamp.br

Objetivos

Escrever um programa que calcule o resultado de uma uma variação da transformação MTF em uma cadeia de caracteres de entrada e a imprima como números decimais.

Introdução

MTF (*Move To Front*), também chamada de *Symbol ranking*, é uma transformação em que os caracteres do alfabeto de um texto são mantidos em uma fila constantemente ordenada pelo tempo desde sua última aparição, do mais recente ao menos recente, e na saída seu valor é substituído pela sua posição em tal fila. Essa transformação é usada principalmente para a compressão de dados. A idéia é que caracteres mais recentes são os mais prováveis de ocorrer novamente no futuro, e isso pode ser aproveitado de várias formas para comprimir melhor. É bastante usado após outra transformação, o BWT (*Burrows-Wheeler Transform*), cuja a saída tende a conter um número bem maior de longas repetições de caracteres.

Quando um símbolo se repete, ele é sempre mandado para o começo da fila. Assim, uma sequência de símbolos iguais quaisquer se transforma em uma sequência de zeros. Tais sequências de zeros podem ser substituídas por um símbolo indicando repetição seguido pelo o número de repetições, no que é chamado RLE (*Run Lenght Encoding*).

Limitar a aplicação do RLE aos zeros é vantajoso pois, caso contrário, se gastaria muitos bits para indicar que tal símbolo não se repete (zero repetições). É improvável qualquer outro número se repetir tanto após uma transformação MTF, e antes de tal transformação você não saberia quais caracteres tendem a se repetir.

Números menores tendem a ser mais frequentes e, assim, recomenda-se atribuir códigos menores para representar tais números. Como consequência, códigos maiores são usados para números maiores e menos frequentes. Assim, consegue-se representar a sequência com um número menor de bits, comprimindo-a.

Neste laboratório, vamos implementar uma variação dessa transformação para exercitar o uso de listas ligadas.

Especificação

1. A primeira linha conterá um número indicando a quantidade de caracteres a serem lidos seguido de um `\n`.

2. A entrada será uma cadeia contínua composta de caracteres imprimíveis do conjunto ASCII, incluindo o caractere espaço. No total, são 95 caracteres diferentes que poderão aparecer, com valor decimal de 32 até 126 (inclusive). Não há limite pré-definido para o tamanho máximo desta cadeia de caracteres. Você deve ler a quantidade de caracteres especificada pelo item acima.

3. Seu programa deve manter uma lista com os caracteres que já apareceram no texto. A lista deve começar vazia.

4. Quando um caractere aparecer pela primeira vez, seu programa deve imprimir como saída o valor do caractere, subtraído do valor base de 32 e somado ao tamanho atual da lista. Por exemplo, o caractere "X" tem valor 88 de acordo com a tabela ASCII. Se o programa já tiver recebido 14 caracteres diferentes antes desse, então a saída vai ser o número decimal 70, que é $88 - 32 + 14$.

5. Novos caracteres devem ser inseridos logo após o primeiro elemento da lista, ou seja, na posição 1 e não na posição 0. A excessão é para o caso de a lista estar vazia. Neste caso, o caractere deve ser inserido na frente da lista (posição 0).

6. Quando aparecer um caractere que já foi visto antes, seu programa deverá imprimir a posição dele na lista e o mover de onde ele estiver para a frente da lista. Assim, por exemplo, se o caractere se repetir imediatamente, seu programa irá imprimir zero.

7. Após cada número que seu programa imprimir, ele deve também imprimir um espaço.

8. Ao final do programa, imprimir uma nova linha (`\n`).

Exemplo 1

Entrada:

```
8
bananaaa
```

Saída:

```
66 66 80 2 2 1 0 0
```

Exemplo 2

Entrada:

```
...ptrnntbtchhhhhhhhhhhhhhh rtr...
```

Saída:

```
34
14 0 0 81 86 85 82 1 3 71 0 73 79 1 0 0 0 0 0 0 0 0 0 0 0 0 0 8 7 3 1 7 0 0
```

Observações

1. Deve ser utilizada uma estrutura de **lista ligada** para a resolução do lab.
2. Os laboratórios devem ser compiláveis usando o GCC 4, com a seguinte linha de execução: `gcc -lm -std=c99 -Wall -pedantic -Werror -o main arquivo.c`
3. É obrigatório liberar toda a memória alocada. Os programas que não liberarem a memória alocada serão considerados errados, independentemente da saída correta nos testes.
4. Códigos ilegíveis serão considerados errados. A legibilidade é obtida com identação correta e coerente, bons nomes de variáveis e funções, bem como boa subdivisão do código em funções auxiliares.
5. Dica: utilize `%*c` no `scanf` para consumir apenas o `\n` após o número.