Trabalho Prático 0: Anagramas

Manoel da Rocha Miranda Júnior

September 22, 2015

1 Introdução

Este trabalho prático tem como objetivo agrupar conjuntos de palavras com base em anagramas e retornar em ordem decrescente a quantidade de palavras anagramas existente em cada grupo. Segundo a definição da wikipédia[1], um anagrama consiste em "*uma espécie de jogo de palavras, resultando do rearranjo das letras de uma palavra ou frase para produzir outras palavras, utilizando todas as letras originais exatamente uma vez*". Por exemplo, partindo da palavra "ATO", é possível formar os anagramas TAO, TOA, AOT, OTA, OAT.

A solução deste problema passa pela utilização de uma lista encadeada de anagramas existentes no texto que são determinados a partir da ordenação das letras de cada palavra em ordem alfabética.

2 Solução

A entrada do problema consiste em N listas ($1 \leq N \leq 10$) de até 10^6 palavras com no máximo 50 caracteres. Em razão do possível consumo de memória no pior caso, uma lista encadeada foi utilizada como estrutura de dados para armazenar as palavras, permitindo que se use posições não contíguas na memória primária.

Ao ler cada palavra da entrada, os caracteres que a compõem são ordenados alfabeticamente objetivando a identificação do anagrama formado por ela. Por exemplo, as palavras AOL e OLA são interpretadas como sendo o 'anagrama' ALO. Cada nó da lista encadeada representa um anagrama e é composto por uma string que armazena o anagrama em si, um contador que é incrementado a cada vez que o anagrama em questão é econtrado e também pelos apontadores para as células adjacentes. Foi utilizado também uma estrutura denominada Set que armazena um ponteiro para o último nó da lista, bem como seu tamanho.

O diagrama abaixo ilustra a estrutura de dados utilizada na modelagem do problema:

[1] Acesso em 19/09/2015

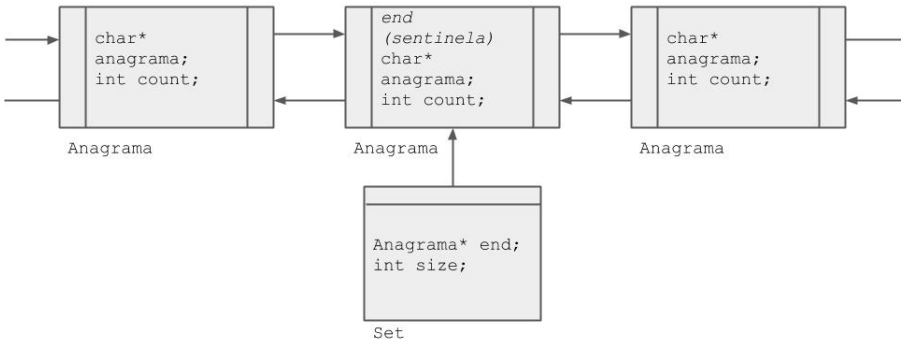

Figure 1: Diagrama do tipo abstrato de dados utilizado

O algoritmo usado na ordenação dos caracteres de cada palavra foi o quicksort por meio do método *qsort*, presente na biblioteca padrão da linguagem C. Este método foi escolhido em função de sua complexidade assintótica em caso médio de $O(n * logn)$. A cada palavra lida da entrada é executado um procedimento de busca na lista pelo seu anagrama. Em caso de uma busca com sucesso, ou seja, caso uma palavra anagrama à nova palavra já tenha sido inserida, é executado apenas um incremento do contador presente na estrutura que a representa. Caso a palavra represente um anagrama não existente na lista, é inserido um nó ao final desta.

Ao final de cada lista presente na entrada, depois de lida todas as palavras e após todos os anagramas serem devidamente identificados e contados, é feita uma iteração sobre toda a lista e os contadores de cada anagrama são inseridos em um arranjo de inteiros. Esse aranjo é em seguida ordenado de forma decrescente usando novamente a função qsort. A saída do programa consiste nos elementos deste arranjo após sua ordenação.

Os procedimentos acima descritos são realizados para cada lista e os dados são gravados na saída padrão.

3 Análise de custo

3.1 Análise Teórica do Custo Assintótico de Tempo

O processo de leitura da entrada consiste primeiramente na ordenação dos caracteres da palavra em O(c log c), com o uso do algorítmo Quick Sort, onde c representa a quantidade de caracteres existentes na palavra. Depois de ordenada, é feita uma busca sequencial pelo anagrama da palavra a ser inserida cujo custo no pior caso pode ser O(n), o que ocorre quando a célula correspondente ao anagrama se encontra no final da lista ou se a busca não tiver sucesso e o anagrama ainda deve ser inserido. Neste caso, n representa o tamanho do conjunto de palavras da lista.

Depois do processo de leitura, a lista de anagramas é percorrida novamente em busca dos contadores de cada grupo, sendo o custo desta operação O(n). A cada iteração, o contador de cada grupo é lido e armazenado no arranjo de contadores para futura ordenação.

O processo de ordenação dos contadores é realizado por meio do algorítmo Quick Sort, cujo custo assintótico no caso médio é O(n log n).

Logo, a complexidade de tempo da solução apresentada é influenciada pelos seguintes fatores:

- Tamanho médio das palavras na lista, que implica no custo de ordenação dos caracteres que a compõem
- Tamanho das listas de palavras existentes na entrada bem como a quantidade de anagramas em cada lista. A quantidade de anagramas de cada lista é relevante devido ao custo de iteração do set e posteriormente da ordenação dos contadores. No pior caso, a quantidade de anagramas é igual a quantidade de palavras.

O custo assintótico no algorítmo como um todo é de:

$$O(n) * [O(p) * (O(p) + O(c * logc)) + O(p) + O(p * logp) + O(p) + O(p)]$$

$$= O(n) * [O(p^2) + O(p * c * logc) + O(p * logp) + O(p)]$$

$$= O(n * p^2) + O(n * p * logc) + O(n * p * logp) + O(n * p) = O(n * p^2) = O(p^2)$$

onde:

- n é igual à quantidade de listas na entrada a serem processadas;
- p representa a quantidade de palavras em uma lista;
- c é a quantidade de caracteres de uma palavra.

Desta forma, podemos concluir que o custo assintótico do algorítmo apresentado é $O(p^2)$, sendo a quantidade de palavras presentes na lista da entrada. Este comportamento poderá ser verificado na seção 4 desta documentação.

3.2 Análise Teórica do Custo Assintótico de Espaço

A análise de custo de espaço pode ser feita levando em consideração a estrutura de dados utilizada. Cada célula da lista encadeada é composta por:

- uma string de, no máximo, 51 caracteres (considerando o caracter terminador de string), totalizando 204 bytes;
- um inteiro de 4 bytes;

- apontadores para os nós adjacentes, totalizando 16 bytes.

Dessa forma, podemos chegar à uma complexidade de espaço final de O(224 * n) = O(n), sendo n a quantidade de palavras em uma dada lista na entrada. Portanto, conclui-se que o consumo de memória cresce linearmente com o tamanho da entrada.

4 Análise experimental

Para a análise expermental desta solução foi implementado um gerador de entradas que leva em consideração o tamanho das palavras que compẽm as listas, e o tamanho da lista em si.

O tempo gasto pelo algorítmo foi medido pelo comando time do Linux. Esta ferramenta retorna o tempo gasto na execução de um comando. O experimento foi feito em uma máquina com processador Intel Core i3 2.3 GHz e 4 GB de memória RAM.

Para verificar a complexidade assintótica de tempo, variou-se o tamanho das listas presentes na entrada. A quantidade de listas se manteve em 10 e o tamanho das palavras em 40 caracteres. O tamanho da lista foi variado de 2000 até 28000, incrementando de 2000 em 2000 a cada amostragem. Os resultados foram plotados no gráfico disposto a seguir:

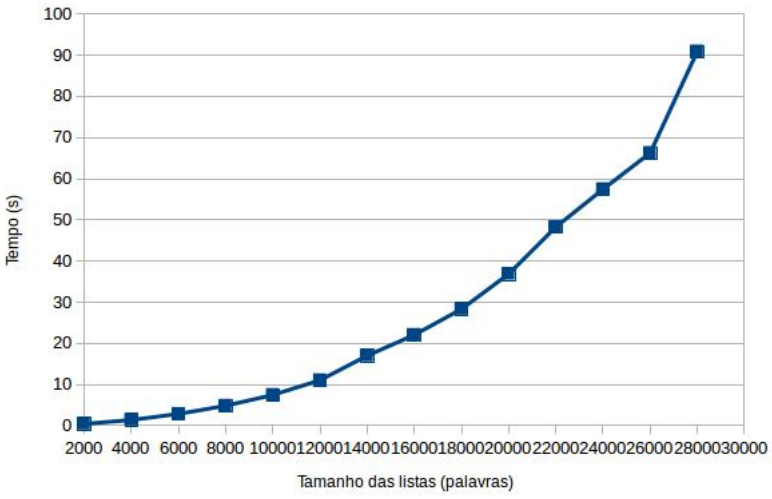

Figure 2: Análise experimental variando o tamanho das palavras entre 2000 e 28000

Conforme podemos aferir pelo formato da curva, a complexidade se mosta, de fato, $O(p^2)$, ou seja, o tempo de execução cresce quadraticamente em relação ao tamanho da entrada, conforme aferido na seção 3.1.

5 Conclusão

A partir deste trabalho prático foi possível solucionar o problema de agrupamento de palavras anagramas bem como a ordenação dos grupos formados para cada lista de palavras presente na entrada. Por meio do TAD lista usando apontadores foi possível modelar os dados da entrada e realizar a computação desses dados. O algorítmo de ordenação e a estrutura de dados utilizados na solução do problema tem um grande impacto na complexidade assintótica do mesmo, que pode ser verificada e comprovada através de experimentos realizados com entradas de tamanho adequado para este tipo de análise.