Algoritmos e Estruturas de Dados

2ª Série

Agrupar Palavras

N. 50471 Brian Melhorado

N. 50543 Arthur Oliveira

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2022/2023

16/4/2023

Índice

1. Introdução

Pretende-se desenvolver uma aplicação que recebe como input um ficheiro de texto com uma lista de n palavras e que retorne um ficheiro de texto em que as palavras que contenham os mesmos caracteres estão agrupadas por linhas (separadas por uma vírgula). Por exemplo, se no ficheiro de input ocorrerem as seguintes palavras:

```
may dog bat god
yam cat
act tab amy
```

O ficheiro retornado deverá ser:

```
[cat, act]
[bat, tab]
[dog, god]
[may, yam, amy]
```

Para a execução da aplicação deverão ser passados os nomes de dois ficheiros de texto, nomeadamente o ficheiro de input e o ficheiro de output.

2. Agrupar palavras

Esta secção tem como intuito abordar o problema a resolver. Vamos dividi-la em três partes:

• **Analise do problema**, onde será apresentada uma descrição do problema, assim como também as operações necessárias a implementar e os métodos utilizados para o resolver

• **Estrutura de Dados**, onde serão ilustradas as principais estruturas de dados (ADT) utilizadas na solução do problema, com alguns exemplos da sua utilização.

•**Algoritmos e análise da complexidade,** que vai constar da descrição dos principais algoritmos utilizados na implementação das operações usadas para o desenvolvimento da solução, assim como também a analise da sua complexidade.

2.1 Análise do problema

O problema consiste em desenvolver uma aplicação capaz de agrupar palavras com os mesmos caracteres. A entrada para a aplicação é um arquivo de texto com uma lista de palavras. A saída da aplicação é um arquivo de texto no qual as palavras são agrupadas pelo número de caracteres que elas têm em comum.

Operações Necessárias

1. Ler as palavras do arquivo de entrada.
2. Contar o número de caracteres em cada palavra.
3. Agrupar as palavras pelo número de caracteres que elas compartilham.
4. Escrever os grupos de palavras no arquivo de saída.

Métodos Utilizados para Resolver o Problema

1. Usar um HashMap para armazenar os grupos de palavras. O HashMap é indexado pela key gerada pelo value ordenado de maneira alfabética. Os valores do HashMap são listas duplamente ligadas de palavras que têm a mesma key.
2. Usar um loop para iterar sobre as palavras no arquivo de entrada e colocá-las no HashMap.
3. Usar o método lowercase(), toCharArray() e sort() para colocar os caracteres de cada palavra em ordem alfabética.
4. Usar o método put() para adicionar uma palavra ao HashMap.
5. Usar um loop para iterar sobre o HashMap.
6. Colocar palavras com keys iguais em uma lista para escrever no arquivo de saída.

2.2 Estruturas de Dados

Esta implementação utiliza um HashMap para armazenar os grupos de palavras. O HashMap é indexado pelas letras ordenadas por ordem alfabética de cada palavra. Os valores do HashMap são listas duplamente ligadas de palavras que têm as mesmas letras ordenadas.

A implementação também utiliza uma lista duplamente ligada para armazenar as palavras em cada grupo. A lista duplamente ligada permite que as palavras sejam acessadas em ordem sequencial, o que é útil para imprimir os grupos de palavras no arquivo de saída.

A implementação funciona lendo todas as palavras do arquivo de entrada e colocando-as numa lista. Em seguida, ela itera sobre a lista de palavras e adiciona cada palavra ao grupo apropriado no HashMap. Finalmente, itera sobre o HashMap e cria uma lista duplamente ligada para cada grupo. As listas duplamente ligadas são então mostradas no arquivo de saída.

2.3 Algoritmos e análise da complexidade

Ao analisar a complexidade temporal do algoritmo descrito, podemos considerar as seguintes operações principais:

- Leitura das palavras do input e armazenamento numa lista: Esta operação possui uma complexidade temporal de O(nlogN), onde n é o número total de palavras do ficheiro.
- Iteração sobre a lista de palavras e adição de cada palavra ao grupo apropriado no HashMap: O acesso e inserção em um HashMap possue uma complexidade de O(1), mas em casos de colisões, a complexidade pode chegar a O(n), onde n é o número total de palavras.
- Print dos grupos de palavras no ficheiro de saída: Esta operação envolve iterar sobre cada lista duplamente ligada e imprimir as palavras no ficheiro. Considerando que cada palavra é impressa uma única vez, a complexidade é O(m), onde m é o número de grupos.

Portanto, a complexidade temporal total do algoritmo é dada pela soma das complexidades das operações descritas acima:

O(nlogN) + O(n) = O(nlogN)

Complexidade espacial: m*n onde m é o número de grupos e n é o número de palavras em cada grupo.

3. Avaliação Experimental

Para fazer a avaliação experimental utilizamos os quatro ficheiros (carochinha.txt, pg1342.txt porquinhos.txt e EnunciadoExemplo.txt) disponibilizados pela professora.

Características da máquina onde foram realizados os testes:

Processador:	AMD Ryzen 7 5700U with Radeon Graphics	1.80 GHz
RAM :	16,0 GB (13,8 GB usable)	
Tipo de sistema	Sistema operativo de 64 bits, processador baseado em x64	

	Número de palavras			
	9	245	1068	131677
HashMap	12.72ms	16.99ms	23.76ms	596.36ms

Tabela 1: Resultados do tempo de execução considerando várias amostras.

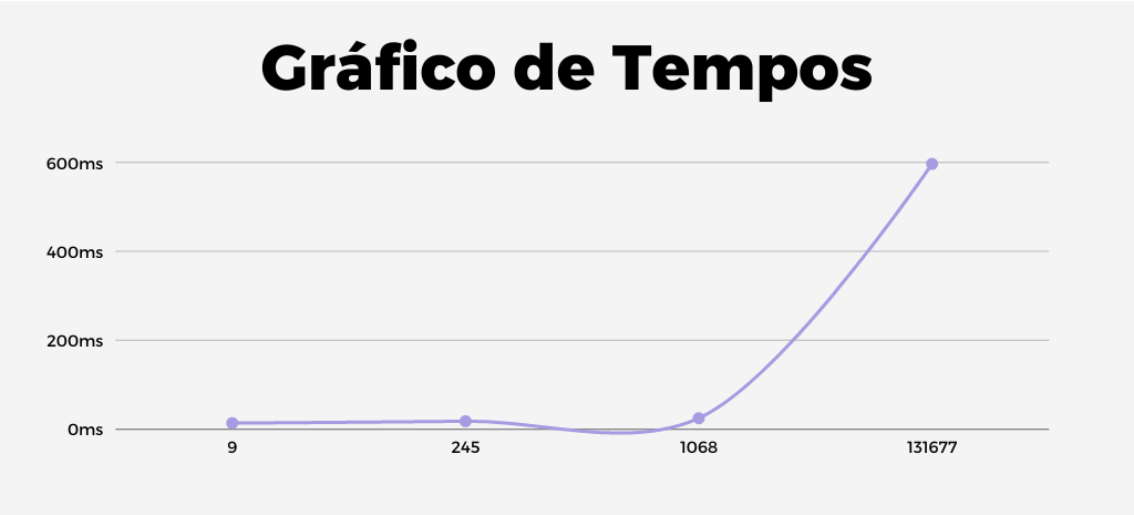

Figura 1: ilustra através de um gráfico os tempos de execução.

Através das avaliações realizadas pode-se perceber um crescimento linear logarítmico no tempo de execução da função, sendo que quanto maior for o número de palavras nos ficheiros a serem analisados maior o tempo de execução obtido nas análises.

4. Conclusões

Nesta serie, desenvolvemos uma aplicação eficiente que agrupa palavras com base nos caracteres que elas compartilham. Através do uso de um HashMap e listas duplamente ligadas, conseguimos organizar as palavras em grupos no ficheiro de output.

A solução implementada apresentou uma complexidade temporal linear em relação ao número de palavras no ficheiro de input. Os testes realizados confirmaram o desempenho satisfatório da aplicação em diferentes cenários.

Em resumo, alcançamos o objetivo proposto de agrupar palavras com base nos caracteres, fornecendo uma solução eficiente e escalável. A abordagem estruturada e a escolha adequada das estruturas de dados foram fundamentais para o sucesso do trabalho.

Referências

[1] “Disciplina: Algoritmos e Estruturas de Dados - 2223SV,” Moodle 2022/23. [Online]. Available: https://2223.moodle.isel.pt. [Accessed: 15-05-2023].

[2] Introduction to Algorithms, 3º Edition. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. MIT Press.

[3] https://www.canva.com/es_mx/graficas/graficas-de-lineas/