Trabalho Prático II: O problema da agenda de viagens de Rick Sanchez

Eduardo de Paiva Dias
2018126657

Belo Horizonte – MG – Brasil

Introdução

O objetivo deste trabalho é praticar algumas algoritmos de ordenação clássicas para implementar um programa que fornece uma agenda de visitação de planetas para Rick visitar da melhor forma possível. É dividido em algumas partes como:

- Ler a quantidade de tempo máxima para visitação de planetas em 1 mês.
- Ler a quantidade de planetas a ser visitados.
- Ler o tamanho do nome de um planeta.
- Ler todos os tempos para visitar um planeta e o nome dele.
- Ordenar os planetas por mês e logo depois ordenar pelo nome.
- Fornecer a agenda para Rick visitar os planetas.

Implementação

Funcionamento geral do programa:

Primeiramente, após ser fornecido os dados iniciais é iniciado uma ordenação de tempo usando o algoritmo de ordenação MergeSort, pois devido a especificação ele é um algoritmo de complexidade O(n*log(n)) e é estável. Logo após isso é definido o mês em que se vai visitar cada planeta. Com isso, a última parte é ordenar pelo nome de cada planeta usando o algoritmo de ordenação RadixSort, devido a especificação ele é um algoritmo de complexidade O(n*k).

As possíveis entradas para o funcionamento do programa seria:

A entrada do programa consiste em um número e um char:

- (Int) Tempo Máximo : para inserir a quantidade de tempo máxima para visitar em um mês uma quantidade de planetas.
- (Int) Quantidade Planetas : para inserir a quantidade de planetas máximo a se visitar.
- (Int) Quantidade Caracteres: para inserir o tamanho de caracteres que se tem o nome de um planeta.
- (Int) Tempo Planeta: Tempo para visitar exatamente esse planeta, contudo não pode ultrapassar o tempo máximo colocado anteriormente.
- (Char) Nome: Nome dado a um planeta, contudo esse nome não pode ultrapassar a quantidade de caracteres máxima colocada anteriormente.

Funcionamento das funções:

TipoItem: Estruturas usada para a criação de um vetor de Planetas.
merge(), mergeSort() : Função que contém o algoritmo clássico de ordenação mergesort.
alterar_indice() : Fornece o mês para visitação, logo após a ordenação por tempo do mergesort.
ContagemChar(), Radixsort() : Função que contém o algoritmo de ordenação radixsort/countingsort.
totalElementosporIndice() : Função auxiliar que fornece a quantidade de planetas que contém em um mês.
ordenaPorIndice() : Coloca os planetas que vão ser visitados no mesmo mês em um vetor auxiliar, logo após isso ordena com o radix sort e retorna para o vetor original.
Imprime() : Imprime a agenda para Rick.
main() : Contém a chamada das funções de mergeSort(), alterarIndice(), ordenaPorIndice() e Imprime() e ainda insere os Planetas em um vetor .

Instruções de compilação e execução

O programa foi desenvolvido no sistema operacional windows na linguagem C e compilado pelo ambiente de desenvolvimento integrado de código aberto e multiplataforma livre Code::Blocks. Contudo para compilar o programa no linux basta escrever no terminal o seguinte comando:

```
gcc trabalhopratico2.c -o trabalhopratico2
```

E para executar o programa podemos escrever no terminal:

```
./trabalhopratico2
```

Ou usar um dos casos de teste:

```
./trabalhopratico2 < 00.in > 00.out
```

sendo 00.in o arquivo de entrada e 00.out o arquivo gerado de saída.

Análise de complexidade

A **complexidade de tempo** para a inserção de um recipiente é O(1), pois se insere diretamente na ultima posição vazia.

A **complexidade de tempo** para mergeSort() é O(n*log(n)) em qualquer caso. Contudo, em **complexidade de espaço** é O(2*n), pois se utiliza um vetor auxiliar para ordenação.

A **complexidade de tempo** para RadixSort() e ContagemChar() é O(k*n) em qualquer caso, sendo k a quantidade de caracteres no nome de um Planeta. A **complexidade de espaço** nesse caso é de **O((2*n)+256)** sendo o vetor original e o auxiliar, além disso contém o vetor de contador.

A **complexidade de tempo** em alterar_Indice() é O(n), pois só se passa uma vez no vetor inteiro colocando o mês, em ordenarPorIndice() a sua **complexidade de tempo** é de O(M) sendo M a quantidade de mês máximo para se visitar todos planetas.

Análise do algoritmo

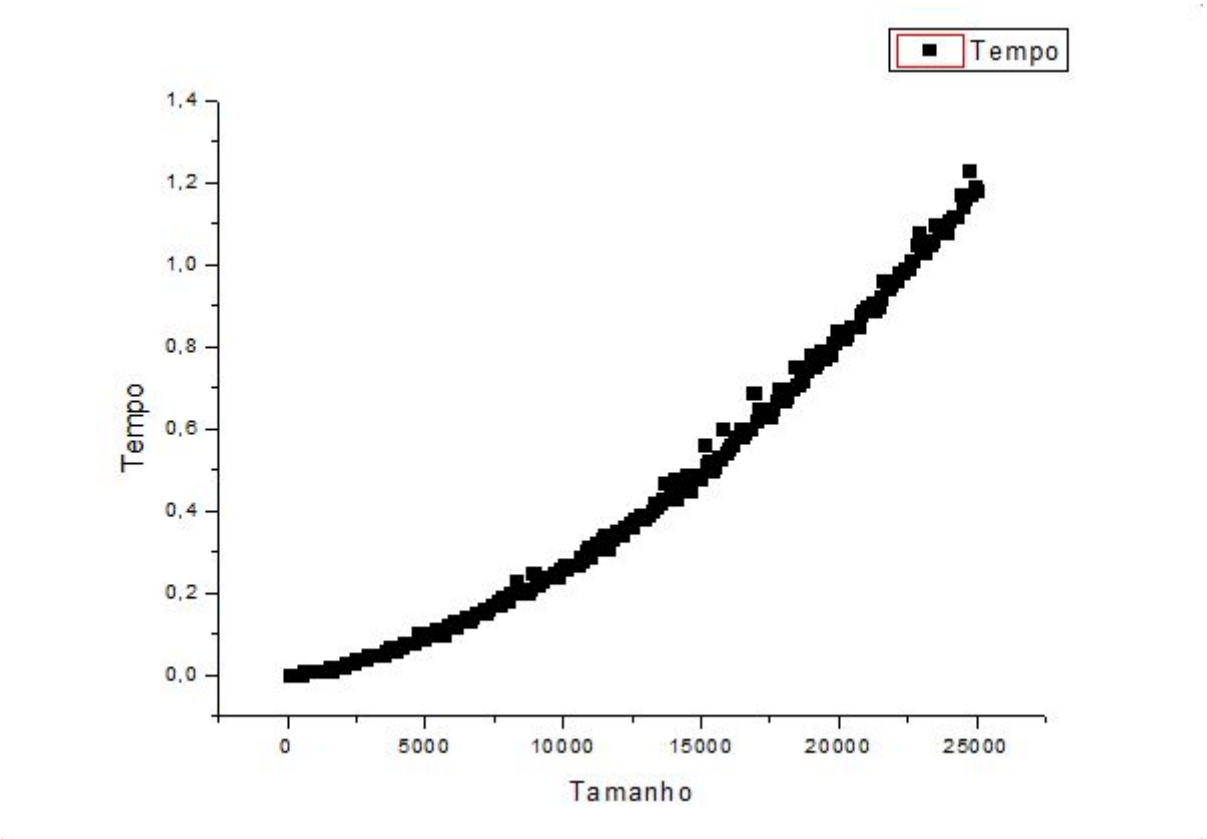

Gráfico gerado no software ORIGIN com 25000 entradas e seu tempo nos testes.

O algoritmo aumenta bastante o seu tempo com o aumento da entrada, sendo exponencial o aumento, contudo seu tempo poderia ser bem menor implementando um algoritmo de radix sort que seu contador não olhasse toda tabela ascii, mas sim a quantidade de letras do alfabeto minusculas.

Conclusão

O trabalho foi concluído com êxito, tanto em questão da utilização dos algoritmos clássicos de ordenação, como na saída correta da agenda, entretanto em alguns caso de teste o algoritmo pode não ser muito eficiente devido a sua longa entrada e a implementação do radix sort não ser muito eficiente.

Bibliografia

CHAIMOWICZ, Luiz; PRATES, Raquel. MergeSort. Belo Horizonte, 2019/1.

CHAIMOWICZ, Luiz; PRATES, Raquel. RadixSort. Belo Horizonte, 2019/1.

ZIVIANI, Nivio. Projeto de algoritmos com implementação em C. Belo Horizonte, 2010.