Tp2 - Estrutura de dados 2019-2
Luis Henrique Soares Monteiro 2018054583

1. Introdução

Dado um número específico de planetas, o tempo total de viagem por mês, um tempo de permanência referente à cada um dos mesmos , além do número de caracteres e o nome de cada dos planetas, é exigido que o programa calcule e retorne uma lista em ordem crescente em relação ao tempo de visita e alfabética em relação aos planetas visitados num mês, de maneira que nos primeiros meses sejam feitas quantas visitas forem possíveis.

1.1. Implementação

Para implementar o programa, foram utilizados dois algoritmos principais, escolhidos em razão de suas complexidades de tempo e estabilidade. Como a ordenação dos tempos de estadia em cada planeta deveria ser feita utilizando um algoritmo de complexidade da grandeza de $O(n\log_2 n)$ que também fosse estável, foi escolhido o Mergesort, por garantir tanto a complexidade de execução como a estabilidade.

Já no caso da ordenação dos nomes da lista de planetas (feita ao final do programa com uma lista já ordenada em relação aos tempos individuais de cada planeta) foi escolhido o algoritmo RadixSort por possuir complexidade O(n*k) assumindo k = log(n).

Considerando a implementação como um todo, o programa inicialmente lê do usuário 3 valores de entrada, referentes respectivamente ao tempo total para visita, o número total de planetas e o número de caracteres para o nome de cada planeta. É instanciado então, uma estrutura Vector e uma célula, utilizados para iterar sobre os planetas que serão adicionados contiguamente nas entradas seguintes.

Após a adição de todos os planetas, o método referente ao algoritmo de ordenação MergeSort é chamado, passando um ponteiro para o vetor de planetas como referência, além de dois inteiros, referentes às posições de início de fim do vetor. Após ordenar o vetor recursivamente (em relação aos tempos de estadia em cada planeta), o programa chama o método OrdenaMes, criado para iterar sobre a lista atribuindo o mês em que cada planeta será visitado.

Em seguida, é executado o algoritmo de ordenação RadixSort sobre o vetor atualizado, com o objetivo de ordená-lo alfabeticamente. Assim que o último algoritmo de ordenação é executado, o programa chama o método Print(), instanciado para imprimir as saídas ordenadas como pedido anteriormente.

Como meu programa atribui o mês em que cada planeta será visitado anterior á ordenação alfabética, encontrei dificuldades para implementar a função de impressão das saídas, já que meu vetor final não termina ordenado como deve ser a saída, mas ordenado alfabeticamente, com cada planeta possuindo um mês a ser visitado. Para imprimir a saída, é executado um loop que compara o mês atual e o mês de visita de cada planeta específico, imprimindo somente os que forem iguais e excluindo itens a medida que são impressos.

1.2. Instruções de compilação e execução

Para compilar o programa utilizando Linux, basta possuir um compilador c++ instalado, acessar a pasta “luismonteiro\src” e digitar make.

Para executar o programa basta digitar ./tp2 pelo terminal.

2. Análise de complexidade

Considerando o programa como um todo, e incluindo a parte mais custosa(imprimir as saídas), a complexidade estaria em torno de 2k + nlog(n) + pk + k(k-i). Logo, seria da ordem de $O(k^2+pk+nlog(n))$ sendo k o número de planetas, p o tamanho do nome dos planetas e i o número de planetas referente a cada mês.

Desconsiderando a função de imprimir a saída a complexidade seria da ordem de O(nlog(n)+pk)).

Abaixo segue o gráfico relacionando tamanho de entradas aleatórias com tempo de execução:

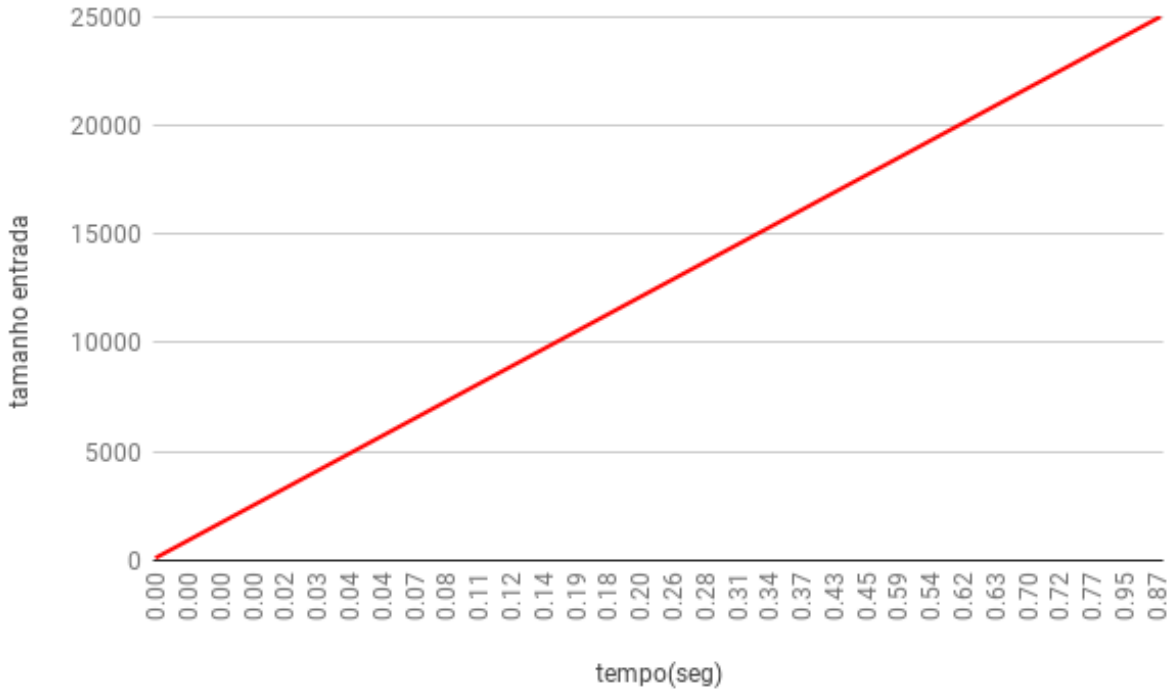

Figura 1. Legenda

3. Conclusão

Em geral, o trabalho foi satisfatório, encontrei dificuldades, como citado acima, para diminuir o tempo de execução da função de imprimir saídas e para implementar o algoritmo RadixSort.

Também encontrei dificuldades na implementação do Radix pois inicialmente estava utilizando estruturas de listas, o que dificulta o acesso e alteração de posições dos planetas, mas o desenvolvimento como um todo foi realizado sem muitos problemas.

4. Bibliografia

Para implementação do MergeSort foram consultados os slides passados em sala de aula, assim como a página GeeksforGeeks e Wikipédia.

Já para a implementação do RadixSort, consultei o livro Projetos de Algoritmos (3^aedição, Nívio Ziviani), mais especificamente a implementação do radixsort com caracteres.