O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

Introdução

O trabalho prático consiste no ordenamento de uma lista de planetas para definir a ordem de vistas aos planetas e os meses das mesmas, respeitando certas condições.Desse modo, o programa usa um array de uma classe Planet, contendo os planetas a serem visitados.

Existe um tempo máximo que se pode gastar em um mês de visitas e caso exceda, os planetas restantes ficam para o próximo mês e assim por diante. É necessário que o tempo de permanência em cada planeta seja cada vez maior com o passar dos meses, ordenando então a visita em ordem crescente por tempo de permanência e após isso, em ordem alfabética na lista de cada mês.

Implementação

No programa existem quatro métodos principais, sendo eles ***mergeSort***, ***decideMonth***, ***reorder*** e ***radixSort***. O compilador é o padrão do linux, ***g++***.

mergeSort

O método ***decideMonth*** é do tipo ***void*** e recebe como parâmetro um array da classe ***Planet*** e os inteiros ***numberOfPlanets*** e ***totalTime***, que correspondem respectivamente, ao números de planetas da lista e o tempo máximo que se pode ser feito visitações em planetas em um mês.

Com isso, é percorrido o array do início até o final, de modo que a cada acesso, é armazenado em uma variável - ***maxTimeSpent*** - a soma do tempo dos restantes com o atual, caso o valor seja maior que o ***totalTime,*** a variável que representa os mês atual - ***month*** - é somada em 1 e o valor de ***maxTimeSpent*** se torna o tempo do planeta atual, feito isso, é definido o mês do planeta.

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

decideMonth

O método ***decideMonth*** é do tipo ***void*** e recebe como parâmetro um array da classe ***Planet*** e os inteiros ***numberOfPlanets*** e ***totalTime***, que correspondem respectivamente, ao números de planetas da lista e o tempo máximo que se pode ser feito visitações em planetas em um mês.

Com isso, é percorrido o array do início até o final, de modo que a cada acesso, é armazenado em uma variável - ***maxTimeSpent*** - a soma do tempo dos restantes com o atual, caso o valor seja maior que o ***totalTime,*** a variável que representa os mês atual - ***month*** - é somada em 1 e o valor de ***maxTimeSpent*** se torna o tempo do planeta atual, feito isso, é definido o mês do planeta.

reorder

O método ***reorder*** é do tipo ***void*** e recebe como parâmetro recebe como parâmetro um array da classe ***Planet***, os inteiros ***numberOfPlanets*** e ***nameSize***, que correspondem respectivamente, ao números de planetas da lista e o tamanho do nome dos planetas.

Essa função, salva o valor do mês do último elemento do array de planetas e então começa um ***while*** com condição de parada apenas quando esse valor, ***currentMonth***, se tornar 0. Feito isso, é feito uma contagem de quantos planetas estão no mês inicial, ***month***, após isso, o valor de ***month*** é acrescido em um e ***currentMonth*** é subtraído em um.

Após isso, é criado um array auxiliar de planetas com o tamanho igual ao número de planetas no mês atual e, então, feito uma cópia desses planetas para o auxiliar, para então utilizar o ***radixSort***, passando como parâmetro o auxiliar, o número de planetas no mês atual - ***planetsInCurrentMonth*** - e o tamanho do nome de cada planeta - ***nameSize*** -, depois, é substituído - por meio de um for - a lista de planetas do mês atual pela lista ordenada dos planetas no mesmo mês.

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

radixSort

O método ***radixSort*** é do tipo ***void*** e recebe como parâmetro recebe como parâmetro um array da classe ***Planet***, os inteiros ***numberOfPlanets*** e ***nameSize***, que correspondem respectivamente, ao números de planetas da lista e o tamanho do nome dos planetas.

É utilizado uma implementação próxima ao ***count sort***, em que dentro de um ***for*** principal - executado do valor final do array de nomes para o ínicio -, é feito a contagem de quantos de cada caracteres existem e a partir disso, é determinado em que posição termina a sequência de cada caractere, feito isso, é então armazenado em um auxiliar a posição correta de cada planeta, de acordo com o seu caractere acessado, enfim, é transcrito para o array original as novas posições dos planetas.

O processo se repete até que o array esteja completamente ordenado em ordem alfabética.

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

Instruções de compilação e execução

Compilação

Acesse pelo terminal a pasta lucas_marinho e digite o comando ***make*** para compilar os arquivos, será gerado um arquivo com o nome ***tp2***.

Execução

Acesse pelo terminal a pasta lucas_marinho e digite o comando ***make test*** para executar os testes e ***./tp2*** para executar o trabalho.

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

Análise de Complexidade

Seja ***n*** o número de planetas e ***k*** o número total de meses.

A complexidade de tempo do ***mergeSort***, no pior e melhor caso, é :

f(n) = O( ), em que ***n ≥ 1*** *n og n l*

Complexidade espacial do ***mergeSort*** , no pior e melhor caso, é :

f(n) = O( ), em que ***n ≥ 1*** *n*

A complexidade de tempo do ***decideMonth,*** no pior e melhor caso, é :

f(n) = O( ), em que ***n ≥ 1*** *n*

A complexidade espacial do ***decideMonth,*** no pior e melhor caso, é :

f(n) = O( ), em que ***n ≥ 1*** 1

A complexidade de tempo do ***reorder,*** no pior e melhor caso, é :

f(n) = O(x), em que ***n ≥ 1*** *n k*

A complexidade espacial do ***reorder,*** no pior e melhor caso, é :

f(n) = O( ), em que ***n ≥ 1*** *n*

A complexidade de tempo do ***radixSort***, no pior e melhor caso, é :

f(n) = O(x), em que ***n ≥ 1*** *n k*

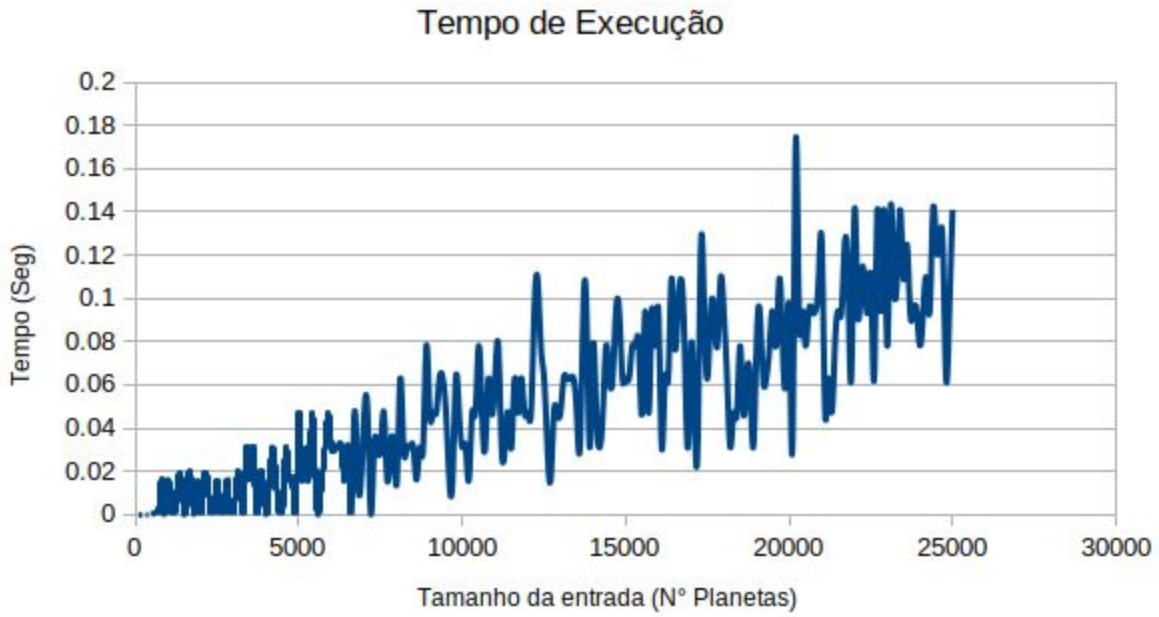
Complexidade espacial do ***mergeSort*** , no pior e melhor caso, é :

f(n) = O(+), em que ***n ≥ 1*** *n k*

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100



O tempo de execução tende a aumentar com o aumento do número de planetas, porém por ser aleatório os valores, alguns casos em que a o número de planetas é grande, porém o total de meses é pequeno, pode gerar tempos melhores que a mesma situação, porém com mais meses e menos planetas, por isso, essa disparidade de tempo de execução em entradas com tempos parecidos.

Devido a métodos como reorder e radix sort que dependem de ambas as variáveis.

O problema da agenda de viagens de Rick Sanchez

Lucas Nascimento Marinho

2018088100

Conclusão

Durante o trabalho foi necessário fazer uma rápida revisão sobre como implementar os ordenadores, principalmente o ***radix sort***, que foi necessário diversas consultas no ***stackoverflow***[2], para poder encontrar casos semelhantes de erros e dificuldades de implementação do mesmo para ***char***.

O foco principal foi nos métodos do ***merge sort***, ***reorder*** e ***radix sort***, sendo os que levaram mais esforços.

O primeiro devido a sua implementação, que de primeira havia feito uma que não era estável, de modo a falhar em alguns dos testes, foi necessário então revisar o código e corrigir tal questão.

O segundo, foi requisitado um certo tempo para decidir com implementar a divisão do vetor de modo satisfatório, para depois então utilizar o ***radix sort***.

O terceiro, a dificuldade por si só de implementar o método para ordenar utilizando char ao invés de ***int***.

Resolvido essas questões, o restante da implementação foi feita sem demais dificuldades, apenas foi necessário remover o ***while*** do main fornecido, pois o mesmo atrapalhava a execução dos testes.

Bibliografia

Durante a programação foi utilizado os sites ***geeks for geeks***[1] , ***stackoverflow***[2] e os materiais dados em sala de aula.

---

[1] https://www.geeksforgeeks.org/

[2] https://stackoverflow.com/