**TRABALHO DE ESTRUTURA DE DADOS**

**SIMULADOR DE ELVADOR**

**Lucas Cavalcante Corrêa**

**14/0070320**

**1/2018**

**Descrição sucinta sobre o desenvolvimento do trabalho.**

O objetivo do programa é criar um simulador de elevador em que duas estratégias pudessem ser escolhidas: First to Come Firts Served (FCFS) e Shortest Job First (SJF).

O programa possui uma função que gera o ambiente do elevador e a lista de eventos. A partir dela, o programa principal recebe a escolha da estratégia pelo usuário, o arquivo .txt do ambiente do elevador e o arquivo .txt dos eventos. Em seguida, dada a estratégia escolhida, o programa principal vai chamar a função da FCFS ou SJF. Cada uma dessas funções chama outras quatro funções que executam os principais procedimentos do elevador: receber um novo pedido da lista de eventos (Adiciona Eventos), verificar se têm alguém para descer (SaidaElevador), verificar se têm alguém para entrar (EntraElevador) e decide em qual sentido ir (Sentido). Cada uma dessas 4 funções chama um conjunto de Tipos Abstratos de Dados (TAD’s). O diagrama a seguir mostra essa cadeia de funções:

Funções dos TAD’s

Em relação aos resultados do programa, foram testadas quantidades de eventos de 10 mil, 100 mil, 1 milhão e 10 milhões; para prédios de 20 e 40 andares e elevadores de capacidade de 10 e 20 pessoas. De maneira geral, se o número de andares ou a capacidade aumentar, o programa acaba levando aproximadamente, 1,5 e 1 vezes, respectivamente, mais tempo para executar o programa. Entretanto, o tempo continua crescendo na mesma proporção da entrada.

Além disso, percebeu-se que a estratégia SJF é mais eficiente do que a FCFS. Ainda mais, para um número de andares e capacidade maior, a estratégia SJF fica ainda mais eficiente em relação a FCFS.

**Descrição dos módulos e sua inter-dependência.**

O programa conta com 4 módulos:

1. Módulo da função principal (main);
2. Módulo das estratégias;
3. Módulo das funções que cada estratégia chama dos TAD’s;
4. Módulo do gerador de eventos e ambiente do elevador.

O módulo que gera o ambiente do elevador e a lista de eventos é independente. O da função principal puxa a biblioteca do módulo das estratégias, que por sua vez puxa a biblioteca do módulo das funções.

**Descrição dos TADs e as estruturas de dados utilizadas.**

Nesse programa foi utilizado as estruturas por listas dinamicamente encadeadas e filas. Existem:

1. Uma lista dinamicamente encadeada que representa a **lista de pessoas dentro do elevador**. Cada elemento da lista é uma struct com os três dados de cada evento mais dois campos para o tempo de espera dentro e fora do elevador.
2. Uma **lista** dinamicamente encadeada que representa os **andares**. Cada elemento dessa lista é uma fila.
3. Uma fila que representa a **fila de pessoas esperando em cada andar**. Cada Elemento da fila possui uma struct com os três dados de cada evento mais dois campos para o tempo de espera dentro e fora do elevador.

Foram utilizadas as funções de:

1. Inserção (cada lista teve um específico);
2. Desinflarar fila;
3. Enfileira fila;
4. Esvaziar Lista (cada lista teve um específico);
5. Esvaziar Fila.

Além dessas funções, o programa realiza determinados comandos para retirar determinados elementos em cada uma das listas. Por questões de facilidade na programação, optou-se por não criar uma função separada para essa operação.

Da forma como os dados estão organizados, acredita-se que a busca por eventos entre os andares é mais eficiente, uma vez que eles estariam organizados por andar e momento. Quando se diz organizado por andar, quer dizer que todos os eventos que são de determinado andar estarão juntos. Não necessariamente a lista de andares vai estar com os elementos organizados por ordem, os elementos dessa lista aparecem conforme a necessidade, ou seja, quando existe a demanda por uma fila naquele andar, quando existe uma pessoa esperando no andar. Quando a fila/elemento do andar se esvazia, esse elemento sai da lista de andares e o espaço na memória alocado para ele é liberado.

Dessa forma, essa estrutura pode ser utilizada nas duas estratégias. Na FCFS, o programa vai escolher o sentido do elevador procurando o evento com menor momento no primeiro elemento de cada fila presente na lisa de andares (uma vez o primeiro item de cada fila é o primeiro a pedir dessa fila). O desenho abaixo mostra o comportamento do elevador quando este está vazio em casos diferentes. Não se considera o pedido da pessoa dentro do elevador.

Elevador

Lista de andares

Fila de cada andar

Pessoa de momento 1

Pessoa de momento 4

Pessoa de momento 2

Pessoa de momento 3

Andar 3

Andar 2

Andar 4

Andar 1

Pessoa de momento 5

Pessoa de momento 5

Lista de andares

Fila de cada andar

Pessoa de momento 4

Pessoa de momento 2

Pessoa de momento 3

Andar 3

Andar 2

Andar 4

Andar 1

Se não tem ninguém na fila do andar 3, ele é retirado

Pessoa de momento 5

Lista de andares

Fila de cada andar

Pessoa de momento 4

Pessoa de momento 3

Andar 2

Andar 4

Andar 1

Pessoa de momento 5

Lista de andares

Fila de cada andar

Pessoa de momento 4

Andar 4

Andar 1

Pessoa de momento 5

Lista de andares

Fila de cada andar

Andar 1

No caso da SJF, o programa ao invés de procurar pelo menor momento no primeiro elemento da fila de cada andar, ele verificaria o andar de cada primeira pessoa nas fila de andares existente e procuraria a que tem a menor distancia. Ou seja, o menor módulo de (andar atual – andar do passageiro).

Para ambos os casos, o programa sempre vai dar preferência na lista de pessoas dentro do elevador. Nessa lista, o primeiro elemento sempre é o que têm preferência.

**Descrição do formato de entrada dos dados.**

O arquivo .txt com a lista de eventos possui uma matriz nX3 em que cada linha é um evento, ou uma pessoa, com um momento em que esta pediu o elevador, o andar de origem e para qual andar ela deseja ir, respectivamente. Esse programa considera que nem sempre no momento seguinte aparecerá uma pessoa nova pedindo o elevador, ou seja, um momento de determinado evento é x e não necessariamente o momento do próximo evento é x+1.

O segundo arquivo de entrada é do ambiente do elevador, contendo apenas dois valores: o número de andares do prédio e a capacidade do elevador, respectivamente.

**Descrição do formato de saída dos dados.**

O programa vai gerar uma sequência de “prints” no console que falaram de qual andar para qual andar o elevador foi, se alguém entrou ou saiu do elevador, informando para onde a pessoa queria/quer ir, qual é a sua origem, o momento que pediu o elevador, tempo de espera fora e dentro do elevador.

**Explicação sobre como utilizar o programa.**

Para executar o programa, escreva: “./Simulador -i Ambiente\_Elevador.txt -e Lista\_Eventos.txt -s <estratégia>”. A opção “–i” vai pegar o arquivo com as informações do ambiente do elevador, “-e” vai pegar a lista com os eventos e “–s” escolherá a estratégia que só podem ser “fcfs” ou “sjf”.

Se você quiser utilizar outros arquivos do ambiente e da lista de eventos, certifique-se de que os seus arquivos estão como especificados no item “Descrição do formato de entrada dos dados” e que você coloque o nome com .txt na linha do comando.

**Estudo da complexidade do programa.**

Quando comparados os tempos de execução do programa para entradas de 10 mil, 100 mil, 1 milhão e 10 milhões, percebeu-se que eles também eram multiplicados por 10. Ou seja, a ordem de complexidade do programa é de O(n).

**Listagem do programa fonte. Listagem dos testes executados.**

Estratégia fcfs para 20 andares e capacidade para 10 pessoas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** |
| 10.000.000 | 250724886 | 266221377 | 1104363167 | 57.604.678 |
| 1.000.000 | 23980100 | 25923562 | 100172876 | 5.765.570 |
| 100.000 | 2431909 | 2582462 | 9597858 | 576.146 |
| 10.000 | 239901 | 253742 | 922722 | 57.819 |

Estratégia sjf para 20 andares e capacidade para 10 pessoas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** |
| 10.000.000 | 212342687 | 228.875.458 | 921125708 | 27.580.030 |
| 1.000.000 | 21584836 | 23184333 | 91288805 | 2.755.156 |
| 100.000 | 2013456 | 2188750 | 7785682 | 276.406 |
| 10.000 | 206944 | 220978 | 785057 | 27.874 |

Estratégia fcfs para 40 andares e capacidade de 10 pessoas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** | **Memoria** |
| 1.000.000 | 39296238 | 41501141 | 150750542 | 12.489.839 | 29982720 |
| 100.000 | 3777514 | 4006162 | 14415044 | 1.249.027 | 3850240 |
| 10.000 | 405272 | 416994 | 1504104 | 125818 | 1232896 |

O computador passou de 30 minutos de execução para 10 milhões, por isso não se colocou essa entrada

Estratégia sjf para 40 andares e capacidade de 10 pessoas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** | **Memoria** |
| 1.000.000 | 19909103 | 22872059 | 80052758 | 4.333.293 | 23441408 |
| 100.000 | 1976908 | 2235072 | 7940463 | 433.363 | 3170304 |
| 10.000 | 200735 | 223654 | 778309 | 43.787 | 1.167.360 |

O computador passou de 30 minutos de execução para 10 milhões, por isso não se colocou essa entrada

Estratégia fcfs para 20 andares e capacidade de 20 pessoas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** | **Memoria** |
| 10.000.000 | 227540159 | 235241049 | 964735154 | 38.053.968 | 108232704 |
| 1.000.000 | 21939417 | 23571637 | 90803791 | 3.803.988 | 22052864 |
| 100.000 | 2162395 | 2303720 | 8678755 | 380.024 | 3055616 |
| 10.000 | 214315 | 222521 | 932064 | 37877 | 1159168 |

Estratégia sjf para 20 andares e capacidade de 20 pessoas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **eventos** | **tempo Sistema** | **tempo usuário** | **tempo relógio** | **Zepslon** | **Memoria** |
| 10.000.000 | 161787903 | 181002095 | 717474046 | 17.926.073 | 84246528 |
| 1.000.000 | 16630315 | 18225455 | 67697387 | 1.794.946 | 9146368 |
| 100.000 | 1681152 | 1853499 | 6772081 | 179.732 | 1802240 |
| 10.000 | 154615 | 170341 | 624415 | 17911 | 1040384 |