Relatório de Estudo de Estrutura de Dados

Estudo da Estrutura de Dados Fila

Alunos: Gustavo Damiani Melo de Souza Duarte
Gustavo Lemos Almeida Andrade
Vinícius Damião

Turma: ADS módulo 3 – 1º semestre de 2023

Introdução

Estrutura de dados é um modo particular de armazenar e organizar os dados em um computador de modo que possam ser usados eficientemente. Existem diferentes formas de organização, que são adequadas a diferentes tipos de aplicação e algumas são altamente especializadas, destinando-se a algumas tarefas específicas. Destacamos neste trabalho a estrutura de dados fila (*queue*), esta estrutura tem como principal objetivo registrar a ordem de chegadas dos elementos que são inseridos nela. Ela possui a lógica FIFO (First in First Out), ou seja, o primeiro elemento que é inserido dentro da fila é também o primeiro a sair, levando isso em consideração, podemos dizer que o primeiro elemento que entra dentro da fila deve ficar no final da fila, já que o primeiro elemento inserido sempre ficará no início para sair primeiro, vale lembrar também que, seguindo essa lógica, o elemento que está posicionado no início é sempre o mais antigo.

A eficiência da estrutura pode ser mensurada por meio do tempo de execução (por exemplo, em milissegundos ou por *clocks* do processador) para a inserção ou remoção de um novo elemento na estrutura. Entretanto, essas variáveis variam conforme a solução de implementação usada, podendo utilizar memória sequencial ou memória dinâmica. Cabe ao programador conhecer as diferenças e eficiência de cada implementação para poder escolher a melhor solução para o seu programa.

Objetivo

Este trabalho visa apresentar dados relacionados a tempo de execuções realizadas pela estrutura de dados fila considerando a implementação que utiliza memória sequencial e que utiliza memória dinâmica, com o objetivo de se compreender a eficiência das duas implementações.

Metodologia

Para obter os dados de eficiência das duas versões da estrutura de dados proposta, definiu-se como variável para ser coletada o tempo de execução em unidades de *clock* de processador. Como essa unidade varia conforme os recursos dos processadores modernos, optou-se por executar cada teste três vezes e calcular a média das mensurações realizadas, que será utiliza durante a discussão dos dados.

Este trabalho optou por implementar as duas soluções em linguagem C, cujo código das operações inserir, remover e listar estão disponíveis a seguir:

```
void enqueue(Queue *queue, int number) {
  if (queue->position + 1 == queue->size) {
    return;
```

```
  }

  ++queue->position;

  queue->numbers[queue->position] = number;
}
```

Código de inserção de elemento na estrutura de dados fila.

```
int dequeue(Queue *queue) {
  if (queue->position == -1) {
    return 0;
  }

  int out = queue->numbers[0];

  for (int i = 0; i < queue->position; ++i) {
    queue->numbers[i] = queue->numbers[i + 1];
  }

  --queue->position;

  return out;
}
```

Código de remoção de elemento na estrutura de dados fila.

```
void print_queue(Queue queue) {
  for (int i = 0; i <= queue.position; ++i) {
    printf("[%d] ", queue.numbers[i]);
  }

  puts("");
}
```

Código de listagem dos elementos na estrutura de dados fila.

Apresentação dos Dados

Primeiramente apresentaremos os dados coletados para a solução que utiliza memória sequencial para, em seguida, apresentar os dados para a solução que utiliza a memória dinâmica.

Dados coletados do algoritmo com a estrutura de dados usando memória sequencial

A Tabela 1 apresenta o tempo de execução em *clocks* considerando a estrutura de dados em estudo e divididas na quantidade de elementos que a estrutura podia suportar: 1.000, 3.000, 5.000, 10.000 e 50.000 elementos.

Tabelas 1, 2 e 3 – Tempo de execução em *clocks* na estrutura de dados com 1.000, 3.000, 5.000, 10.000 e 25.000 elementos considerando as operações de inserção, remoção e listagem para a implementação da estrutura de dados usando memória sequencial

Tabela de Tempo de Execução em *Clocks*								
Operação	Quantidade de Elementos							
	1.000				3.000			
Nº. da bateria	1	2	3	M	1	2	3	M
Inserção	128	115	112	118	327	300	303	310
Remoção	4.747	4.736	4.746	4.743	15.117	10.393	10.420	11.976
Listagem	10	6	13	10	49	36	33	39

Tabela de Tempo de Execução em *Clocks*								
Operação	Quantidade de Elementos							
	5.000				10.000			
Nº. da bateria	1	2	3	M	1	2	3	M
Inserção	565	526	538	543	1.035	1.024	1.049	1.036
Remoção	47.354	29.452	29.539	35.448	132.296	116.417	116.118	121.610
Listagem	62	55	55	57	69	56	56	60

Tabela de Tempo de Execução em *Clocks*								
Operação	Quantidade de Elementos							
	25.000							
Nº. da bateria	1	2	3	M				
Inserção	2.541	2.509	2.507	2.519				
Remoção	748.852	720.318	720.967	730.045				
Listagem	73	80	76	76				