

Atividade Prática 8

1 - Estruturas de Dados a serem monitoradas

Nosso código irá envolver, novamente, a Pilha implementada por Filas Circulares. Como exploramos na AP 6, a Pilha é completamente dependente da Fila Circular para seu funcionamento. Dessa maneira, estaremos monitorando a Pilha. Porém, consequentemente, estaremos monitorando a Fila também.

2 - Funções a serem instrumentadas

As funções `empilha()`, `desempilha()` e `printPilha()` serão instrumentadas. Elas utilizam as funções `enfila`, `desenfila()`, `filaFrente()` e `filaVazia()`.

3 - Definir fases de monitoramento

No nosso programa main, iremos criar a Pilha(e consequentemente duas Filas Circulares), empilhar 25 números e desempilhar 13 números para depois imprimirmos a Pilha.

- Fase 0: Criação da Pilha e das Filas.
- Fase 1: Empilhamento dos elementos.
- Fase 2: Desempilhamento dos elementos.
- Fase 3: Impressão da Pilha.

4 - Instrumentar o Código

O código foi instrumentado com o memlog.

```
6   Pilha::Pilha(){
7       this->id = idPilha;
8       ESCRIVEMEMLOG((long int) (&(this->fila1)), sizeof(double), this->id);
9       idPilha++;
10  }
11
21  void Pilha::empilha(int valor){
22      this->fila2.enfila(valor);
23      ESCRIVEMEMLOG((long int) (&(this->fila2.array[fila2.filaFrente()])), sizeof(double), this->fila2.id);
24
25      while(this->fila1.filaVazia() == false){
26          LEMEMLOG((long int) (&(this->fila1.array[fila1.filaFrente()])), sizeof(double), this->fila1.id);
27          this->fila2.enfila(this->fila1.filaFrente());
28          this->fila1.desenfila();
29      }
30
31      FilaCircular filaAuxiliar = this->fila1;
32      this->fila1 = this->fila2;
33      this->fila2 = filaAuxiliar;
34
35      std::cout << valor << " Empilhado\n";
36  }
```

```

void Pilha::desempilha(){
    if(this->fila1.filaVazia() == true){
        return;
    }
    else{
        this->fila1.desenfila();
        LEMEMLOG((long int) (&(this->fila1.array[fila1.filaFrente()])), sizeof(double), this->fila1.id);
        ESCREVEMEMLOG((long int) (&(this->fila1.array[fila1.filaFrente()])), sizeof(double), this->fila1.id);
    }

    std::cout << "Desempilhando\n";
}

```

```

51 void Pilha::printPilha(){
52     FilCircular filaAuxiliar = this->fila1;
53
54     while(filaAuxiliar.filaVazia() == false){
55         std::cout << filaAuxiliar.filaFrente() << " ";
56         LEMEMLOG((long int) (&(filaAuxiliar.array[filaAuxiliar.filaFrente()])), sizeof(double), filaAuxiliar.id);
57         ESCREVEMEMLOG((long int) (&(filaAuxiliar.array[filaAuxiliar.filaFrente()])), sizeof(double), filaAuxiliar.id);
58         filaAuxiliar.desenfila();
59     }
60     std::cout << "\n";
61 }

```

```

4
5 char lognome[1000];
6
7 int main(){
8     lognome[0] = 'l';
9     lognome[1] = 'o';
10    lognome[2] = 'g';
11
12    iniciaMemLog(lognome);
13    ativaMemLog();
14
15    defineFaseMemLog(0);
16    Pilha pilha, pilha2;
17
18    defineFaseMemLog(1);
19    for(int i = 0; i < 25; i++){
20        pilha.empilha(i);
21    }
22
23    defineFaseMemLog(2);
24    for(int i = 0; i < 13; i++){
25        pilha.desempilha();
26    }
27
28    std::cout << "Pilha após todas operações:\n";
29    defineFaseMemLog(3);
30    pilha.printPilha();
31
32    desativaMemLog();
33    finalizaMemLog();
34
35    return 0;
36 }

```

5 - Definir o Plano de Experimentos

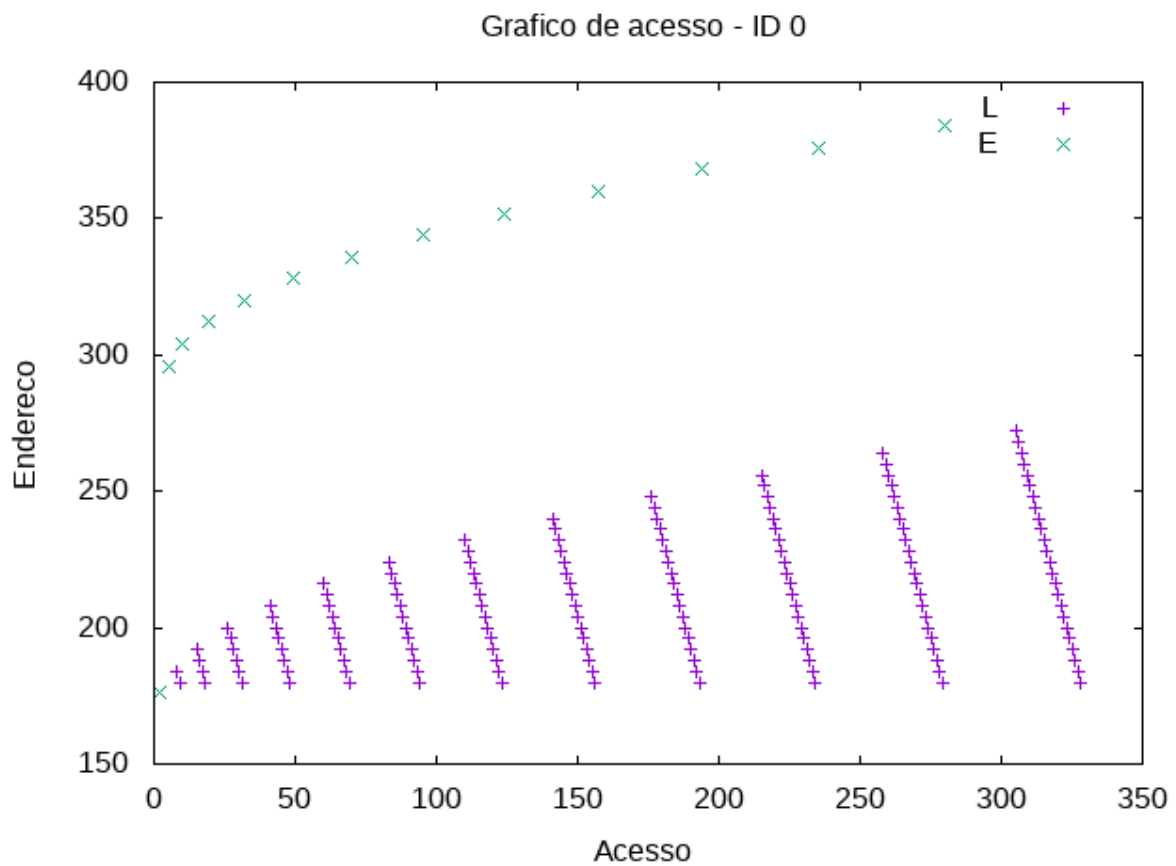
Como pode ser visto no código acima, iremos:

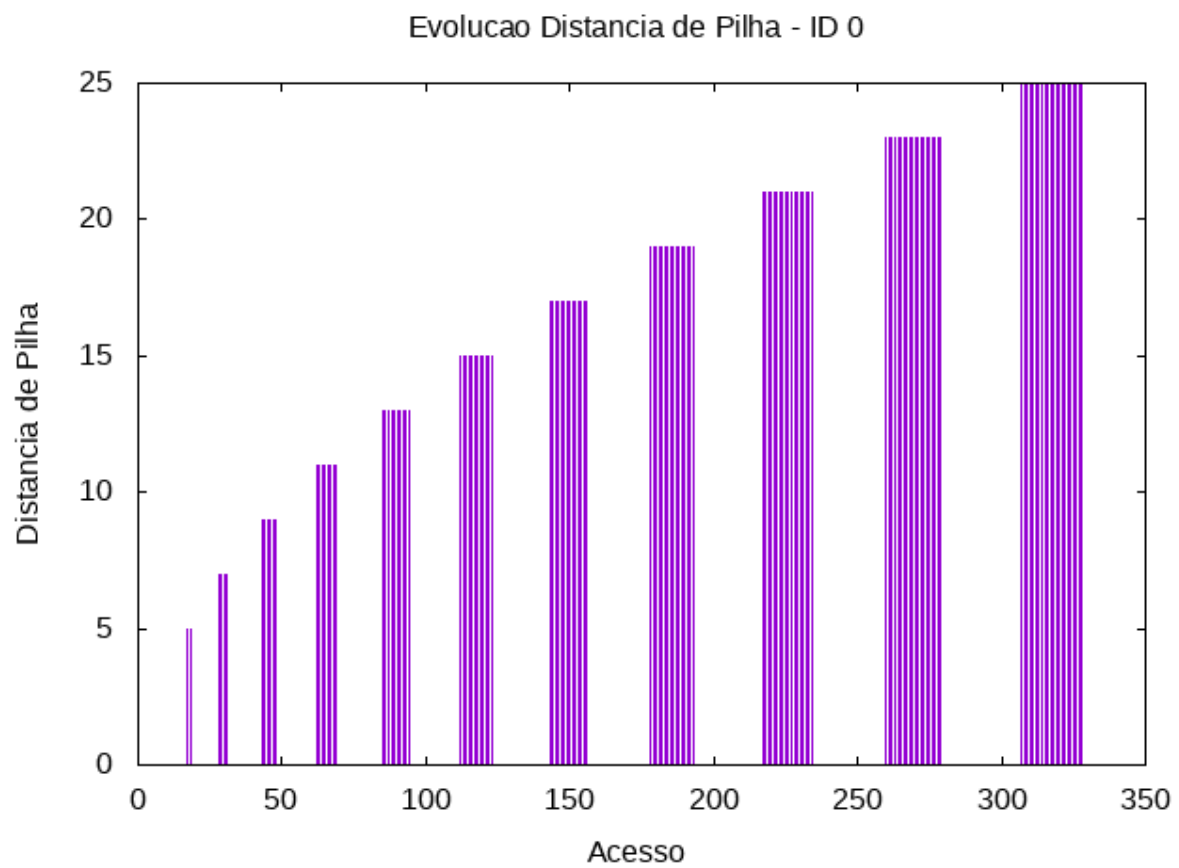
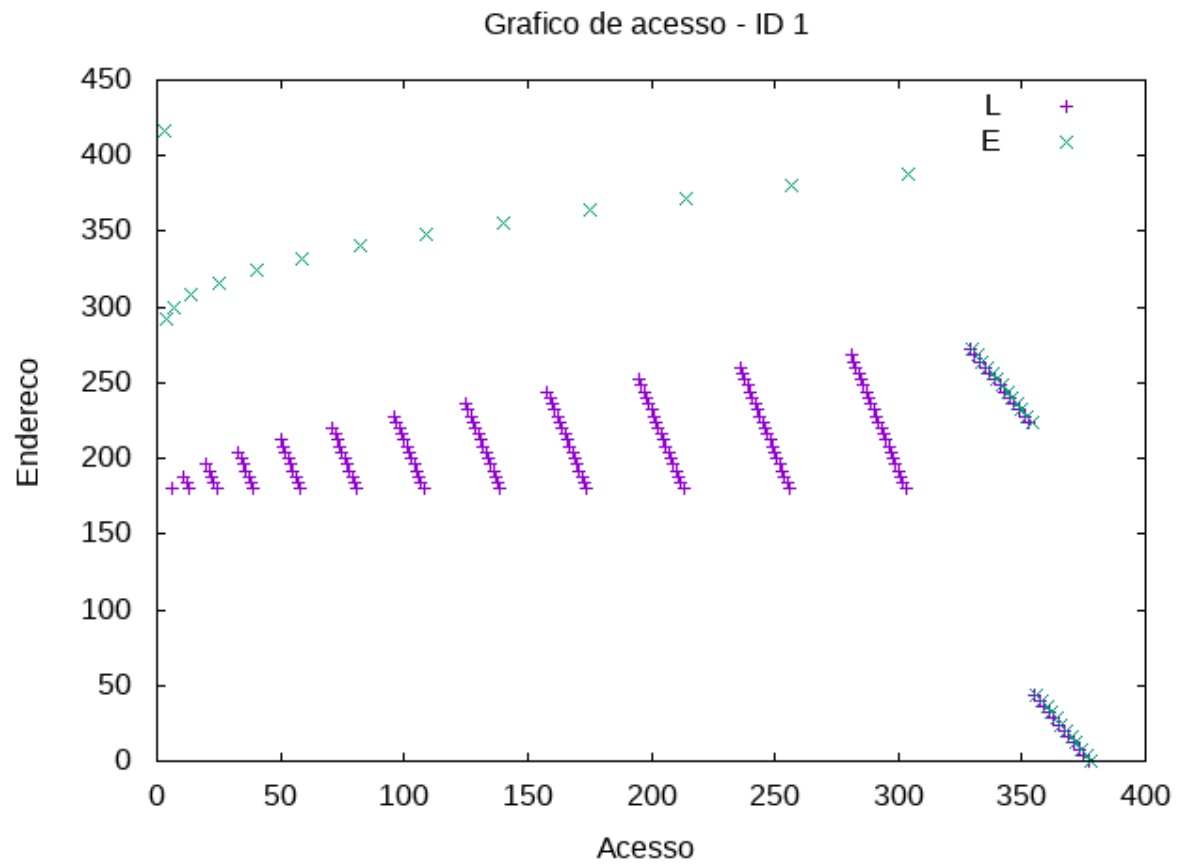
- Criar 7 Pilhas.
- Empilhar 25 elementos em uma Pilha.
- Desempilhar 13 elementos em uma Pilha.
- Exibir uma Pilha.

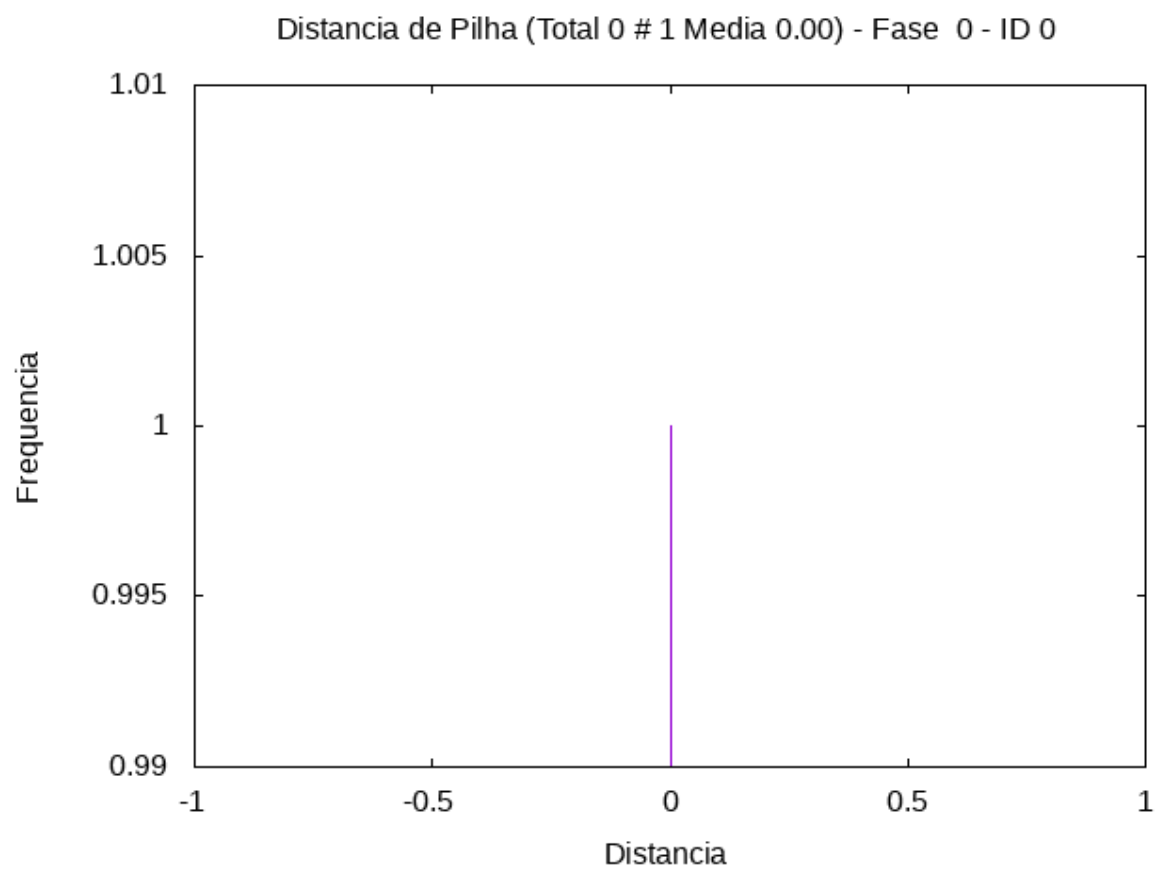
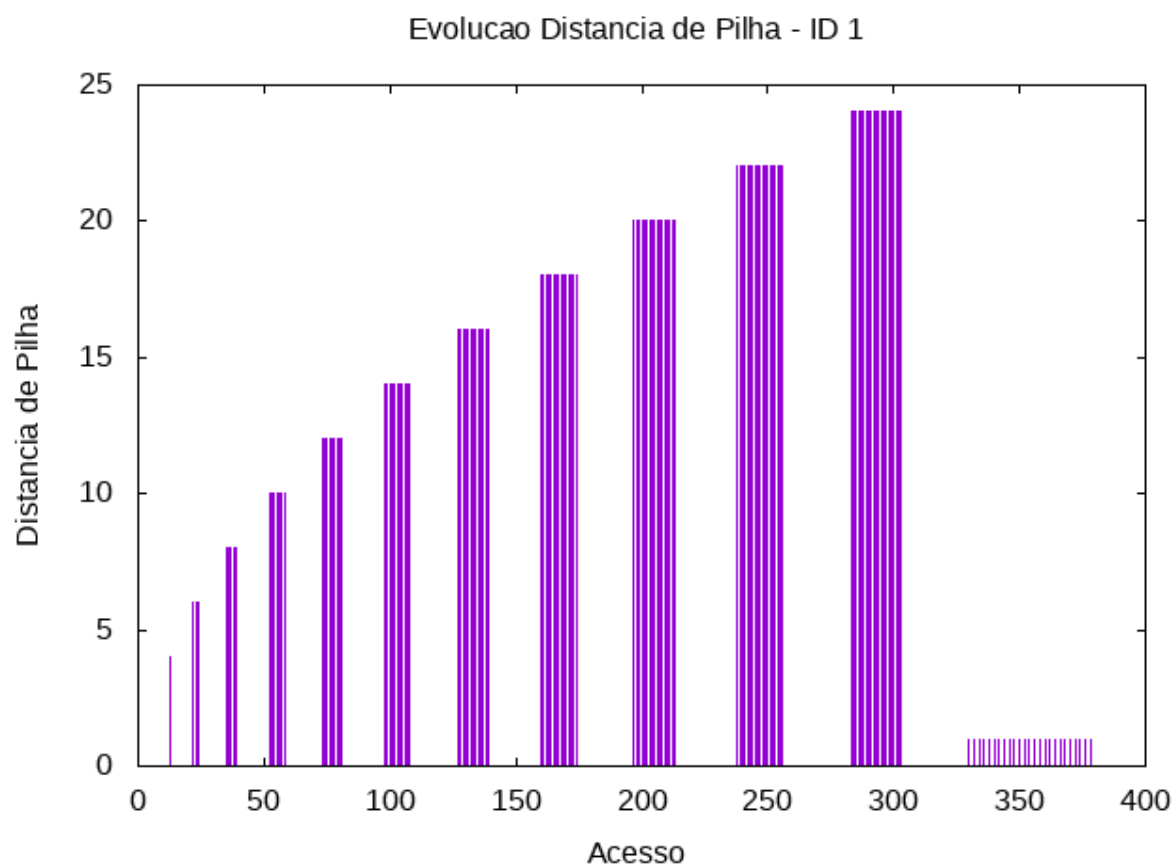
6 - Executar os Experimentos

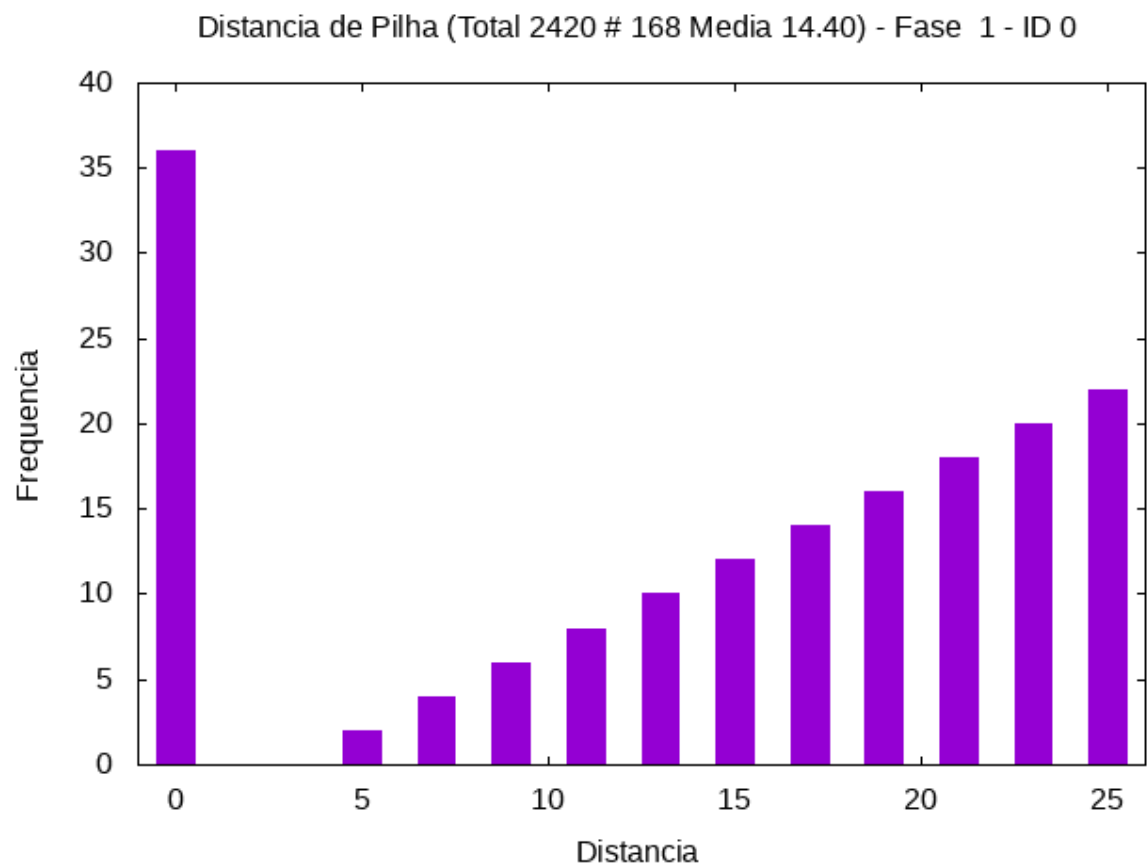
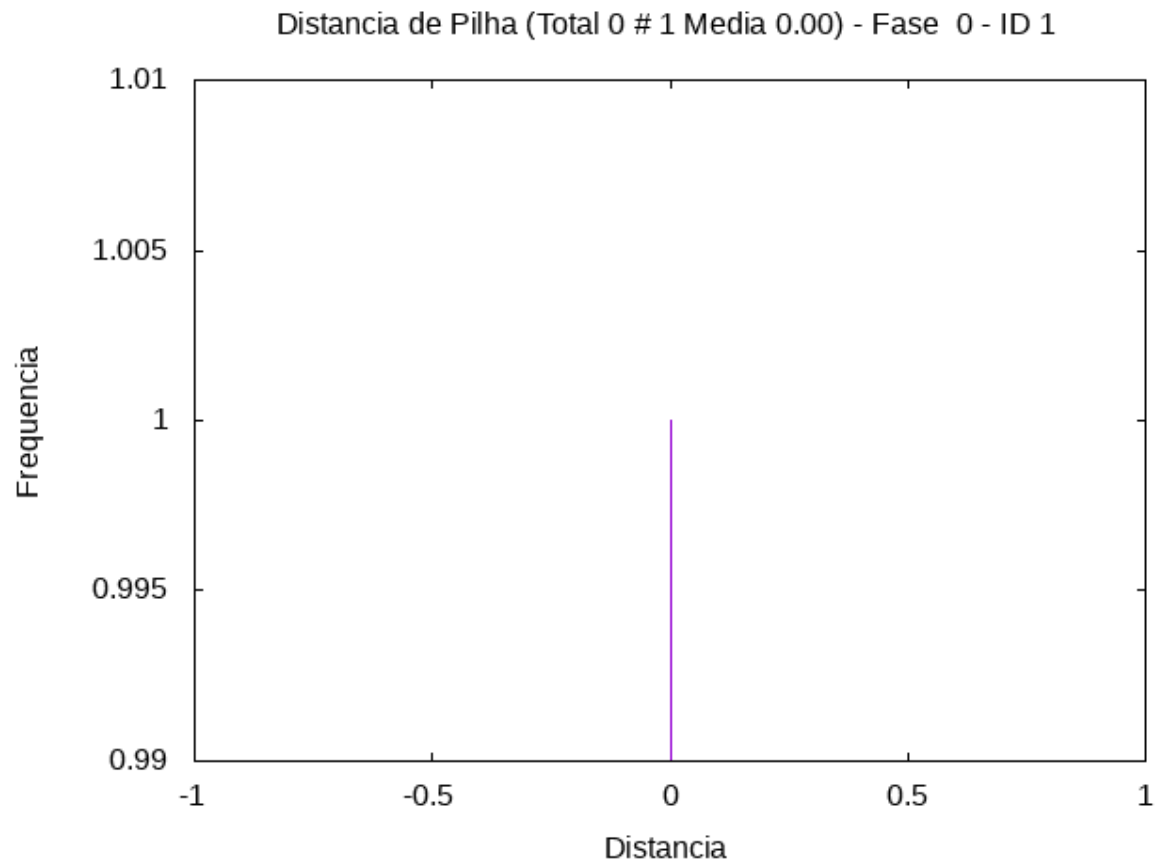
Executamos os experimentos, gerando nosso arquivo do memlog.

7 - Gerar as Visualizações

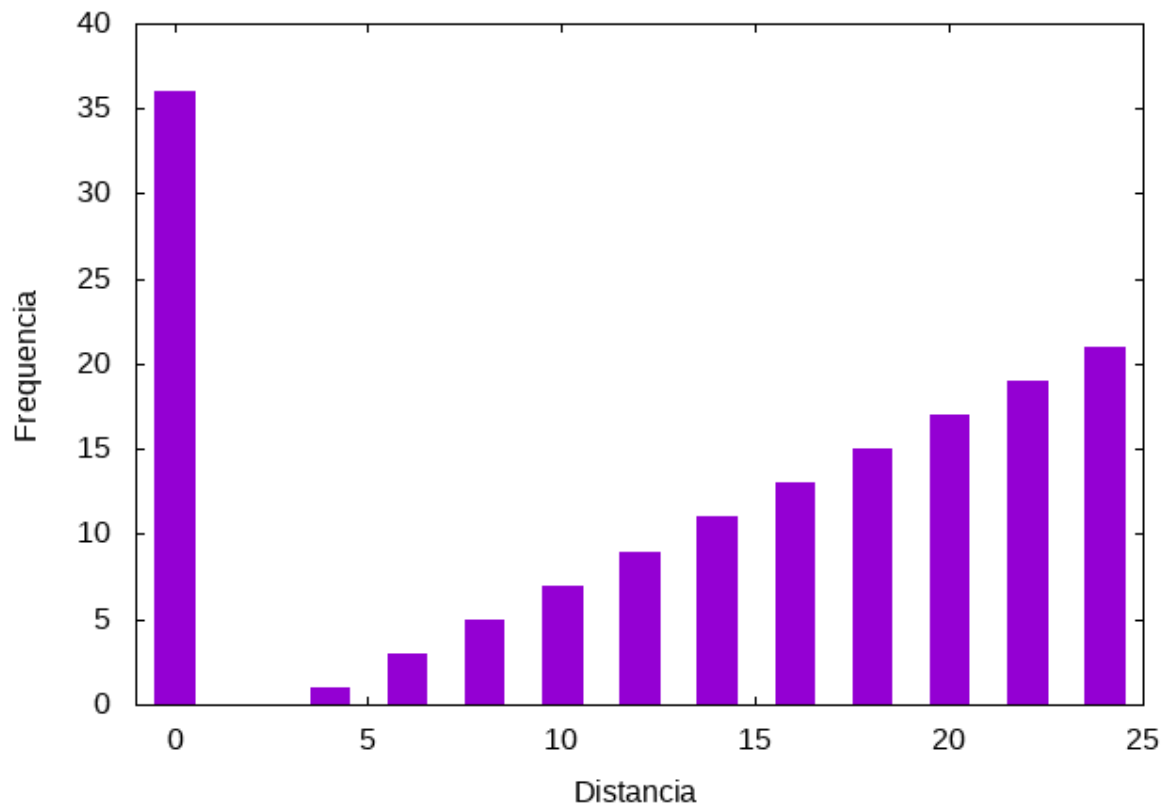




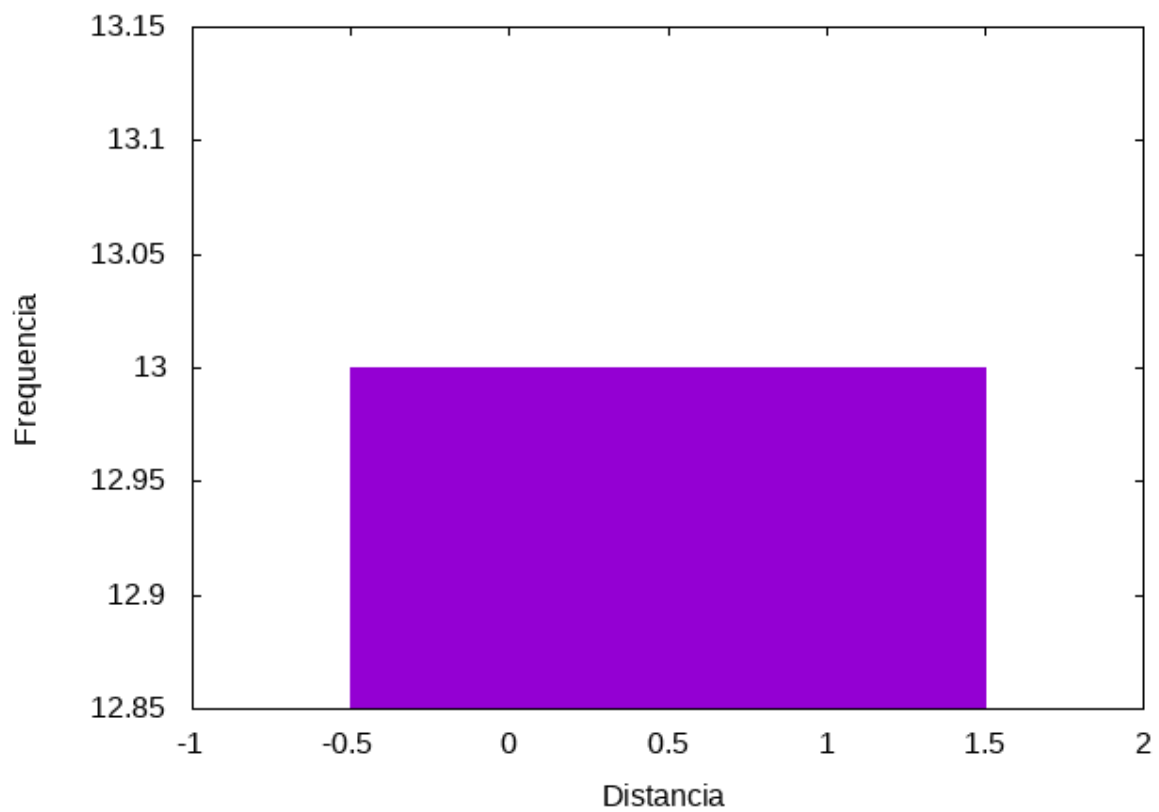


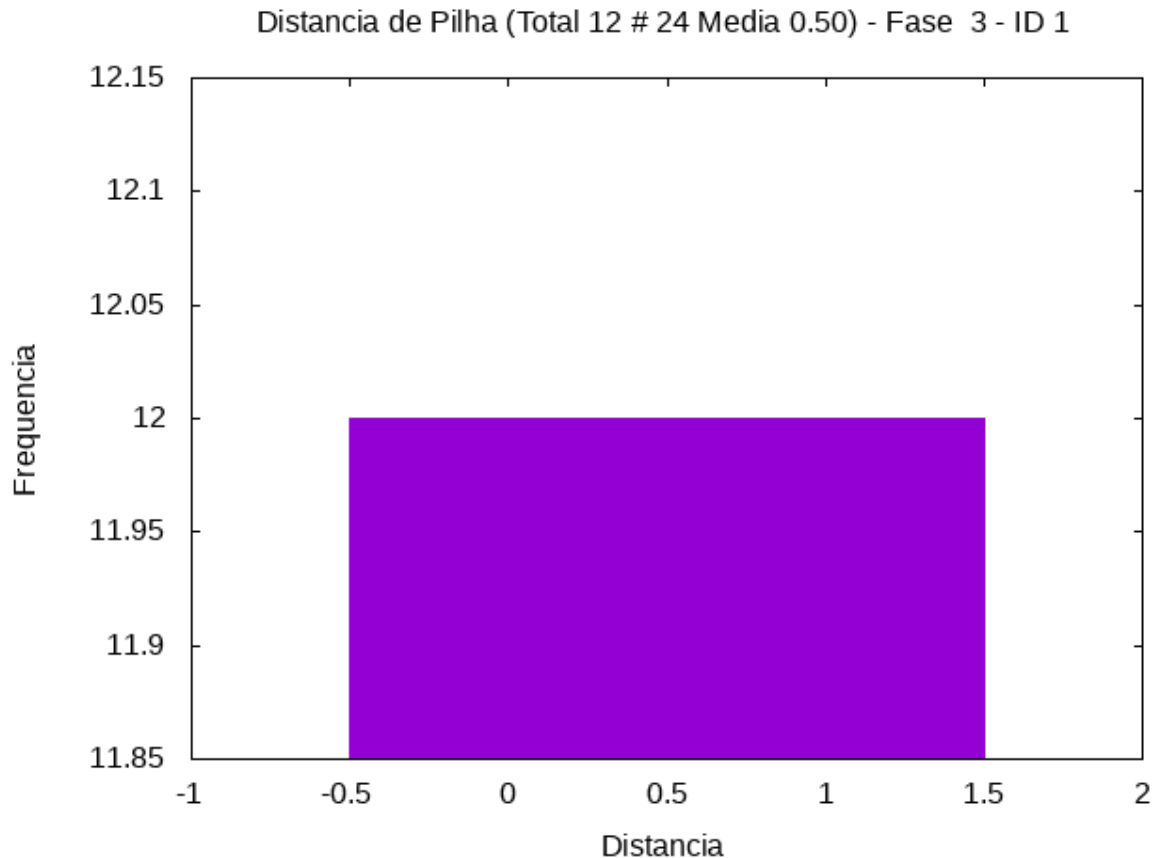


Distancia de Pilha (Total 2134 # 157 Media 13.59) - Fase 1 - ID 1



Distancia de Pilha (Total 13 # 26 Media 0.50) - Fase 2 - ID 1





8 - Analisar os Resultados e Visualizações

Primeiramente, o gráfico de acesso nos permite visualizar com clareza cada operação relacionada às duas filas circulares, de ID 0 e 1.

Tanto na fila 0 e na fila 1, a instância única de escrita em cada momento se refere ao empilhamento de um valor. A cadeia longa de leituras se refere à cópia de seus valores para a criação da fila auxiliar utilizada no método de empilhamento.

Por fim, exclusivamente na fila 1, a cadeia de escritas em conjunto com leituras se refere ao método de impressão da pilha, onde a fila é copiada uma vez e tem todos seus valores acessados.

Ademais, a Distância de Pilha também permite tais análises. A da fase 0 indica sua alocação única. A da fase 1 reflete o distanciamento da parte de trás da fila que nós observamos a modo de que mais elementos são empilhados. A fase 2 se demonstra constante pois, o desenfilamento apenas diminui o topo da pilha, mas o array da fila ainda possui o elemento, por isso não há diminuição da distância. A fase 3 também permanece constante, porque os elementos são apenas acessados para sua impressão.