Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Ciência da Computação

DCC 205 - Estrutura de Dados

Analisamem

Manual do Usuário

V1.2

Abril de 2022

[**1.Descrição**](#_7yy3ddl5yxu5) **2**

[**2.Formato do Arquivo de Entrada**](#_omvzqsdw4i02) **2**

[**3.Instalação**](#_8l9j9fngedu3) **3**

[**4.Compilação**](#_9zmawg6tpyeg) **4**

[**5.Teste**](#_fxrzvkii3426) **4**

[**6.Utilização**](#_xiu691ivcqwv) **9**

[6.1 Gráfico de Acesso](#_cw3mcf6rpxyf) 9

[6.1.1 Soma de Matrizes](#_i6ojs7z6crik) 10

[6.1.2 Multiplicação de Matrizes](#_ki7hui3w728n) 12

[6.1.3 Transposição](#_lwqrzhvty2aw) 15

[6.2 Distâncias de Pilha](#_h0rmqsksb435) 16

[6.3 Uso da Análise de Localidade de Referência](#_i3bvoyom0c2y) 20

[6.3.1 Entendimento](#_agqwa1i0n30) 20

[6.3.2 Depuração](#_fr8d3bs8os08) 20

# 1.Descrição

A aplicação analisamem tem por objetivo analisar a localidade de referência de aplicações a partir de arquivos de registro de acesso gerados pela biblioteca memlog.

# 2.Formato do Arquivo de Entrada

A biblioteca memlog gera um arquivo ASCII contendo 4 tipos de eventos:

1. **Início de registro**

Início de registro, identificado pela letra I, é gravado quando da chamada de iniciaMemLog e contem as seguintes informações:

I <contador> <temporelogio>

Onde <contador> é o contador global de eventos gravados e <temporelogio> é o tempo em segundos, com resolução de nanosegundos, desde 1/1/1970. A seguir mostramos um exemplo de evento:

I 1 73403.203730681

1. **Escrita**

Escrita de dados, identificado pela letra E, é gravado quando da chamada de escreveMemLog e contem as seguintes informações:

E <fase> <contador> <idtad> <tempodesdeinicio> <endereco> <tamanho>

Onde <fase> é a fase definida por chamada anterior a defineFaseMemLog, <contador> é o contador global de eventos gravados, <idtad> é o número inteiro que define a estrutura de dados sendo rastreada (parâmetro id de escreveMemLog), <tempodesdeinicio> é o intervalo de tempo entre o tempo de relógio da chamada a escreveMemLog e o tempo de início de registro, <endereco> é o endereço de memória monitorado, e <tamanho> é o número de bytes monitorados. A seguir mostramos um exemplo de evento:

E 0 2 0 0.000027074 140732612523456 8

1. **Leitura**

Leitura de dados, identificado pela letra L, é gravado quando da chamada de leMemLog e contem as seguintes informações:

L <fase> <contador> <idtad> <tempodesdeinicio> <endereco> <tamanho>

Onde <fase> é a fase definida por chamada anterior a defineFaseMemLog, <contador> é o contador global de eventos gravados, <idtad> é o número inteiro que define a estrutura de dados sendo rastreada (parâmetro id de escreveMemLog), <tempodesdeinicio> é o intervalo de tempo entre o tempo de relógio da chamada a leMemLog e o tempo de início de registro, <endereco> é o endereço de memória monitorado, e <tamanho> é o número de bytes monitorados. A seguir apresentamos um exemplo do evento:

L 1 750042 0 0.279493578 140732612523456 8

1. **Fim de registro**

Fim de registro, identificado pela letra F, é gravado quando da chamada de finalizaMemLog e contem as seguintes informações:

F <contador> <temporelogio> <tempodesdeinicio>

Onde <contador> é o contador global de eventos gravados, <temporelogio> é o tempo em segundos, com resolução de nanosegundos, desde 1/1/1970, e <tempodesdeinicio> é o intervalo de tempo entre o tempo de relógio da chamada a finalizaMemLog e o tempo de início de registro. A seguir mostramos um exemplo de evento:

F 1000202 73403.570855182 0.367124501

# 3.Instalação

Para instalar a aplicação analisamen, basta descompactar o arquivo analisamem.tar.gz e depois compilá-lo, descrito a seguir.

> cd <diretoriodestino>

> gzip -d analisamem.tar.gz

> tar xvf analisamem.tar

# 4.Compilação

Para compilar basta:

> cd <diretoriodestino>

> make bin

# 5.Teste

São providos dois testes: biblioteca pilhaindexada e aplicacao

A biblioteca pilhaindexada pode ser testada utilizando o programa

testepilhaindexada, através dos seguintes comandos:

> cd <diretoriodestino>

> make test

Será impresso um conjunto de acessos a pilhaindexada, exercitando as

funções da biblioteca, como mostrado a seguir:

bin/testepilhaindexada

empilha 0 à 1 = 0

---------------------------------------

Topo 1 Max 5

1|0.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 1 à 1 = 1

---------------------------------------

Topo 1 Max 5

1|1.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 2 à 2 = 0

---------------------------------------

Topo 2 Max 5

2|2.000000000 0| Ant 2 Prox 1

1|1.000000000 0| Ant 2 Prox -1

---------------------------------------

empilha 3 à 1 = 2

---------------------------------------

Topo 1 Max 5

1|3.000000000 0| Ant 1 Prox 2

2|2.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 4 à 3 = 0

---------------------------------------

Topo 3 Max 5

3|4.000000000 0| Ant 3 Prox 1

1|3.000000000 0| Ant 3 Prox 2

2|2.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 5 à 4 = 0

---------------------------------------

Topo 4 Max 5

4|5.000000000 0| Ant 4 Prox 3

3|4.000000000 0| Ant 4 Prox 1

1|3.000000000 0| Ant 3 Prox 2

2|2.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 6 à 1 = 3

---------------------------------------

Topo 1 Max 5

1|6.000000000 0| Ant 1 Prox 4

4|5.000000000 0| Ant 1 Prox 3

3|4.000000000 0| Ant 4 Prox 2

2|2.000000000 0| Ant 3 Prox -1

---------------------------------------

empilha 7 à 2 = 4

---------------------------------------

Topo 2 Max 5

2|7.000000000 0| Ant 2 Prox 1

1|6.000000000 0| Ant 2 Prox 4

4|5.000000000 0| Ant 1 Prox 3

3|4.000000000 0| Ant 4 Prox -1

---------------------------------------

empilha 8 à 5 = 0

---------------------------------------

Topo 5 Max 5

5|8.000000000 0| Ant 5 Prox 2

2|7.000000000 0| Ant 5 Prox 1

1|6.000000000 0| Ant 2 Prox 4

4|5.000000000 0| Ant 1 Prox 3

3|4.000000000 0| Ant 4 Prox -1

---------------------------------------

empilha 9 à 1 = 3

---------------------------------------

Topo 1 Max 5

1|9.000000000 0| Ant 1 Prox 5

5|8.000000000 0| Ant 1 Prox 2

2|7.000000000 0| Ant 5 Prox 4

4|5.000000000 0| Ant 2 Prox 3

3|4.000000000 0| Ant 4 Prox -1

---------------------------------------

empilha 10 à 2 = 3

---------------------------------------

Topo 2 Max 5

2|10.000000000 0| Ant 2 Prox 1

1|9.000000000 0| Ant 2 Prox 5

5|8.000000000 0| Ant 1 Prox 4

4|5.000000000 0| Ant 5 Prox 3

3|4.000000000 0| Ant 4 Prox -1

---------------------------------------

empilha 11 à 3 = 5

---------------------------------------

Topo 3 Max 5

3|11.000000000 0| Ant 3 Prox 2

2|10.000000000 0| Ant 3 Prox 1

1|9.000000000 0| Ant 2 Prox 5

5|8.000000000 0| Ant 1 Prox 4

4|5.000000000 0| Ant 5 Prox -1

---------------------------------------

empilha 12 à 4 = 5

---------------------------------------

Topo 4 Max 5

4|12.000000000 0| Ant 4 Prox 3

3|11.000000000 0| Ant 4 Prox 2

2|10.000000000 0| Ant 3 Prox 1

1|9.000000000 0| Ant 2 Prox 5

5|8.000000000 0| Ant 1 Prox -1

---------------------------------------

empilha 13 à 5 = 5

---------------------------------------

Topo 5 Max 5

5|13.000000000 0| Ant 5 Prox 4

4|12.000000000 0| Ant 5 Prox 3

3|11.000000000 0| Ant 4 Prox 2

2|10.000000000 0| Ant 3 Prox 1

1|9.000000000 0| Ant 2 Prox -1

---------------------------------------

desempilha 5

---------------------------------------

Topo 4 Max 5

4|12.000000000 0| Ant 4 Prox 3

3|11.000000000 0| Ant 4 Prox 2

2|10.000000000 0| Ant 3 Prox 1

1|9.000000000 0| Ant 2 Prox -1

---------------------------------------

desempilha 4

---------------------------------------

Topo 3 Max 5

3|11.000000000 0| Ant 3 Prox 2

2|10.000000000 0| Ant 3 Prox 1

1|9.000000000 0| Ant 2 Prox -1

---------------------------------------

desempilha 3

---------------------------------------

Topo 2 Max 5

2|10.000000000 0| Ant 2 Prox 1

1|9.000000000 0| Ant 2 Prox -1

---------------------------------------

desempilha 2

---------------------------------------

Topo 1 Max 5

1|9.000000000 0| Ant 1 Prox -1

---------------------------------------

desempilha 1

---------------------------------------

Topo -1 Max 5

---------------------------------------

---------------------------------------

Topo -1 Max 5

---------------------------------------

O teste de aplicação utiliza os arquivos exemplo gerados pela biblioteca memlog que constam do diretório teste:

-rw-rw-r-- 1 meira meira 25950 nov 9 18:45 multlog.out

-rw-rw-r-- 1 meira meira 11950 nov 9 18:45 somalog.out

-rw-rw-r-- 1 meira meira 4950 nov 9 18:45 transplog.out

E o teste pode ser executado pelo comando e sua saída típica:

> make use

if test -d /tmp/out; then rm -rf /tmp/out; fi

mkdir /tmp/out

bin/analisamem -i teste/multlog.out -p /tmp/out/mult

minend 140725505023504 maxend 140725505024144 range 640 numend 82

minfase 0 maxfase 2 range 2 numfase 3

minid 0 maxid 2 numid 3

0 0 0 25

0 0 25 25

0 1 0 25

0 1 25 25

0 2 0 25

1 0 0 25

1 0 5 100

1 0 25 25

1 1 0 25

1 1 9 1

1 1 13 2

1 1 16 1

1 1 17 2

1 1 19 2

1 1 21 3

1 1 22 2

1 1 23 2

1 1 24 1

1 1 25 109

1 2 0 25

1 2 1 100

1 2 25 50

2 2 0 25

2 2 25 25

bin/analisamem -i teste/somalog.out -p /tmp/out/soma

minend 140723273169200 maxend 140723273169840 range 640 numend 82

minfase 0 maxfase 2 range 2 numfase 3

minid 0 maxid 2 numid 3

0 0 0 25

0 0 25 20

0 1 0 25

0 1 25 20

0 2 0 25

1 0 0 20

1 0 20 20

1 1 0 20

1 1 20 20

1 2 0 25

1 2 20 20

1 2 25 20

2 2 0 20

2 2 20 20

bin/analisamem -i teste/transplog.out -p /tmp/out/transp

minend 140727060730464 maxend 140727060730656 range 192 numend 26

minfase 0 maxfase 2 range 2 numfase 3

minid 0 maxid 0 numid 1

0 0 0 25

0 0 21 4

0 0 22 4

0 0 23 4

0 0 24 4

0 0 25 4

1 0 0 24

1 0 11 1

1 0 13 1

1 0 15 1

1 0 16 1

1 0 17 2

1 0 18 1

1 0 19 8

1 0 20 1

2 0 0 20

2 0 20 20

gnuplot /tmp/out/\*.gp

Warning: empty y range [25:25], adjusting to [24.75:25.25]

Warning: empty y range [25:25], adjusting to [24.75:25.25]

ls /tmp/out/

mult-acesso-0-0.gpdat mult-hist-1-2.gp soma-hist-1-0.eps

mult-acesso-0-1.gpdat mult-hist-1-2.gpdat soma-hist-1-0.gp

mult-acesso-0-2.gpdat mult-hist-2-0.gpdat soma-hist-1-0.gpdat

mult-acesso-0.eps mult-hist-2-1.gpdat soma-hist-1-1.eps

mult-acesso-0.gp mult-hist-2-2.eps soma-hist-1-1.gp

mult-acesso-1-0.gpdat mult-hist-2-2.gp soma-hist-1-1.gpdat

mult-acesso-1-1.gpdat mult-hist-2-2.gpdat soma-hist-1-2.eps

mult-acesso-1-2.gpdat soma-acesso-0-0.gpdat soma-hist-1-2.gp

mult-acesso-1.eps soma-acesso-0-1.gpdat soma-hist-1-2.gpdat

mult-acesso-1.gp soma-acesso-0-2.gpdat soma-hist-2-0.gpdat

mult-acesso-2.eps soma-acesso-0.eps soma-hist-2-1.gpdat

mult-acesso-2.gp soma-acesso-0.gp soma-hist-2-2.eps

mult-hist-0-0.eps soma-acesso-1-0.gpdat soma-hist-2-2.gp

mult-hist-0-0.gp soma-acesso-1-1.gpdat soma-hist-2-2.gpdat

mult-hist-0-0.gpdat soma-acesso-1-2.gpdat transp-acesso-0-0.gpdat

mult-hist-0-1.eps soma-acesso-1.eps transp-acesso-0.eps

mult-hist-0-1.gp soma-acesso-1.gp transp-acesso-0.gp

mult-hist-0-1.gpdat soma-acesso-2.eps transp-acesso-1-0.gpdat

mult-hist-0-2.eps soma-acesso-2.gp transp-hist-0-0.eps

mult-hist-0-2.gp soma-hist-0-0.eps transp-hist-0-0.gp

mult-hist-0-2.gpdat soma-hist-0-0.gp transp-hist-0-0.gpdat

mult-hist-1-0.eps soma-hist-0-0.gpdat transp-hist-1-0.eps

mult-hist-1-0.gp soma-hist-0-1.eps transp-hist-1-0.gp

mult-hist-1-0.gpdat soma-hist-0-1.gp transp-hist-1-0.gpdat

mult-hist-1-1.eps soma-hist-0-1.gpdat transp-hist-2-0.eps

mult-hist-1-1.gp soma-hist-0-2.eps transp-hist-2-0.gp

mult-hist-1-1.gpdat soma-hist-0-2.gp transp-hist-2-0.gpdat

mult-hist-1-2.eps soma-hist-0-2.gpdat

# 6.Utilização

Uma vez que se obtenha o arquivo de registro de acessos a memória, basta usar o aplicativo analisamem para gerar os vários arquivos e gráficos, como demonstrado.

Para as discussões a seguir vamos utilizar o programa matop, que implementa três operações com matrizes bidimensionais alocadas estaticamente:

* Soma de matrizes
* Multiplicação de matrizes
* Transposição de matrizes

## 6.1 Gráfico de Acesso

O Gráfico de Acesso apresenta, para cada TAD monitorado, os acessos definidos por endereços da variável ao longo da execução, distinguindo Leituras (L) e Escritas (E). O eixo Acesso mostra o contador de acesso de e o eixo Endereço mostra o endereço de memória acessado (nesse caso, o processamento do analisamem subtrai o menor endereço registrado de cada endereço, de forma a facilitar a análise.

A seguir apresentamos os gráficos de acesso para as três operações, nominalmente soma de matrizes, multiplicação de matrizes e transposição.

### 6.1.1 Soma de Matrizes

A soma de matrizes, representada por *A+B=C*, envolve 3 matrizes de mesmas dimensões, onde os elementos das matrizes *A* e *B* são somados gerando a matriz *C*, ou seja:

*A[i,j]+B[i,j]=C[i,j]*

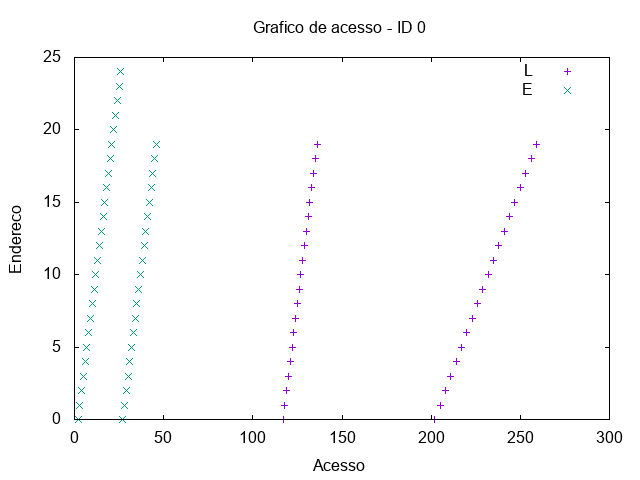
Onde *i* e *j* variam de acordo as dimensões da matriz. No exemplo mostrado a seguir, as matrizes são 4X5, ou seja, tem 4 linhas e 5 colunas.

Para fins de análise, a matriz A recebeu o identificador 0, a matriz B o identificador 1 e a matriz C o identificador 2. A seguir apresentamos os mapas de acesso de cada uma dessas matrizes

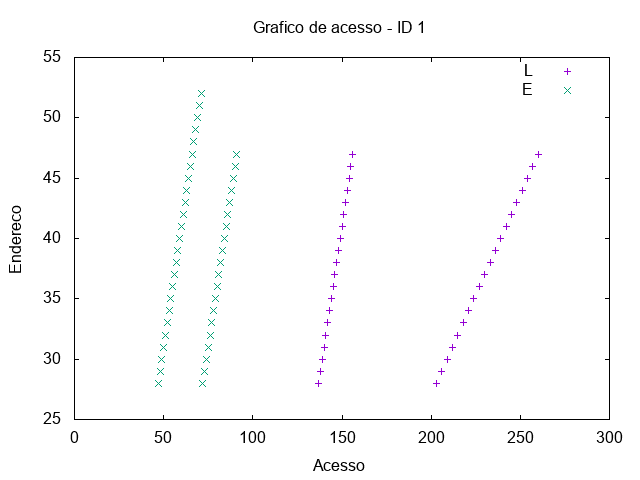
O gráfico de acesso da matriz A mostra as duas fases nas quais a referida matriz foi acessada.

A fase 0 compreende a inicialização da matriz. Inicialmente, até o evento 25, verificamos os acessos para inicializar a matriz A como nula, seguido de acessos para inicializá-la como aleatória compreendendo os eventos 26 a 45. A inicialização aleatória compreende menos eventos pois acessa apenas as posições da matriz que serão efetivamente utilizadas.

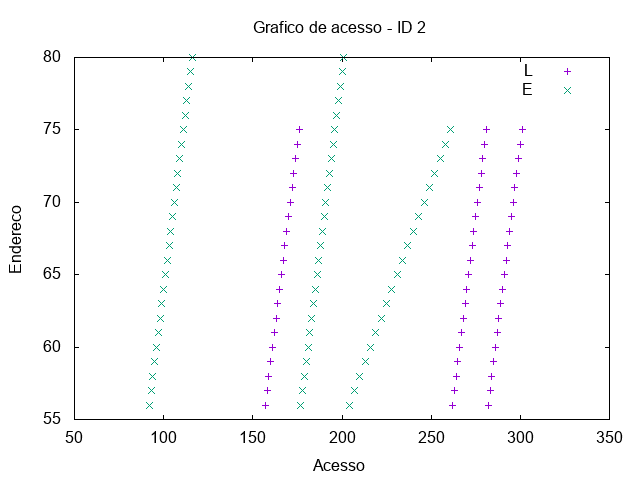
A fase 1 compreende a soma. O primeiro conjunto de acessos, entre 100 e 150, são resultado da função acessaMatriz, que tem por objetivo apenas inicializar as distâncias de pilha percorrendo a matriz linha a linha. O segundo conjunto de acessos, a partir do evento 200, mostra os acessos realizados durante a soma.



Os acessos à matriz B são muito semelhantes, quando se alteram apenas os identificadores dos eventos, sendo mostrados a seguir:



A análise se torna mais interessante quando avaliamos o gráfico de acesso da matriz C, mostrado abaixo:



Nesse caso temos 3 fases. A fase 0, inicialização, apenas realiza a inicializa a matriz C como nula, o que ocorre a partir do evento 90 até o evento 115, aproximadamente. A fase 1 se inicia após o evento 150, com o acessaMatriz e nova inicialização. A soma ocorre a partir do evento 200, quando cada escrita é precedida de uma leitura de cada elemento das matrizes A e B. Finalmente, a fase 2, que imprime a matriz, se inicia com a chamada da função acessaMatriz, seguida da impressão propriamente dita, compreendendo os eventos a partir de 255.

### 6.1.2 Multiplicação de Matrizes

A multiplicação de matrizes, representada por A\*B=C, envolve três matrizes onde o número de colunas da matriz A tem que ser idêntico ao número de linhas da matriz B. A matriz C, resultante, terá o número de linhas da matriz A e o número de colunas da matriz B como dimensões.

Cada elemento C[i,j] é o produto interno da linha i da matriz A pela coluna j da matriz B. Por exemplo, considerando que as matrizes são 3X3, temos:

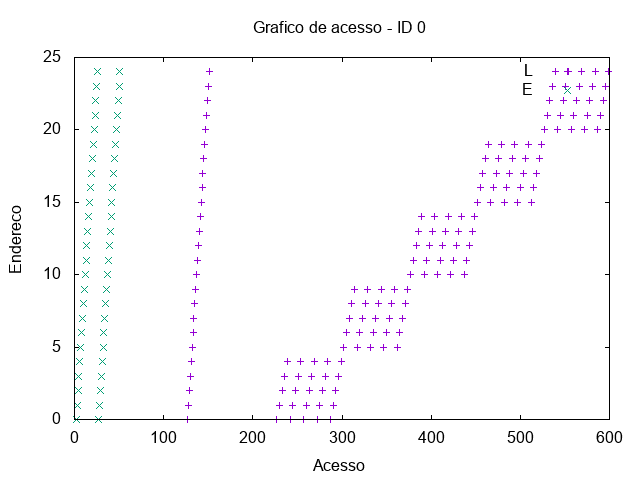
*C[1,0] = A[1,0]\*B[0,0]+A[1,1]\*B[1,0]+A[1,2]\*B[2,0]*

Para a discussão a seguir, consideramos que o preenchimento da matriz C é feito linha a linha e as matrizes A, B e C tem dimensões 5X5, ou seja, 5 linhas e 5 colunas.

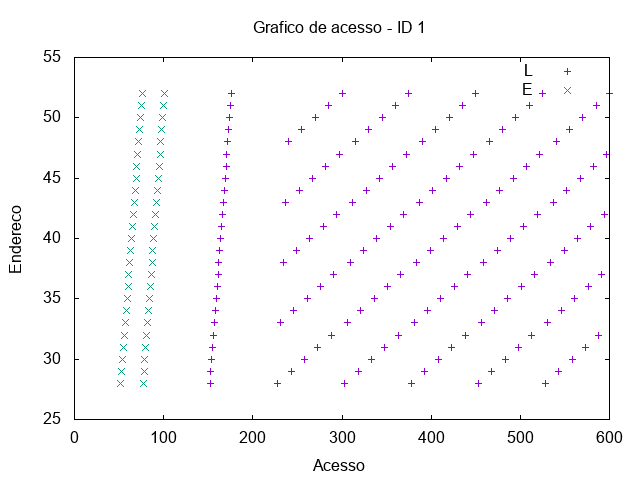
Para fins de análise, a matriz A recebeu o identificador 0, a matriz B o identificador 1 e a matriz C o identificador 2. A seguir apresentamos os gráficos de acesso de cada uma dessas matrizes.

O gráfico de acesso à matriz A, mostrado abaixo, mostra as duas fases na qual a matriz é acessada. A fase 0, até o evento 50, possui os acessos de inicialização nula e inicialização aleatória.

A fase 1 é a multiplicação das matrizes propriamente dita, a qual se inicia com a execução da função acessaMatriz para inicializar as distâncias de pilha, seguido dos acessos propriamente ditos. Analisando o gráfico, percebemos que cada linha da matriz é acessada 5 vezes, uma para cada coluna da matriz B. Por exemplo, a linha 0, que ocupa os endereços de 0 a 4, é acessada no intervalo de eventos entre 220 e 300, aproximadamente. Na sequência, a linha 1, endereços 5 a 9, é acessada também 5 vezes, no intervalo de eventos 300 a 380. O padrão de acesso se repete para as linhas 2, 3 e 4.



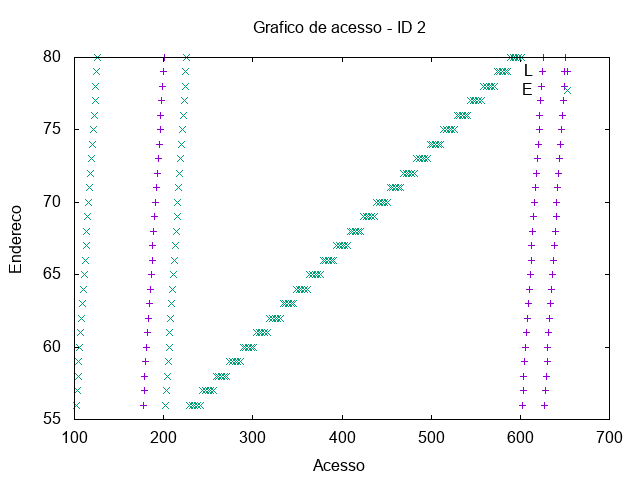
A seguir podemos avaliar o gráfico de acesso da matriz B, quando visualizamos a fase 0, com as inicializações nula e aleatória, compreendendo os eventos 50 a 100, à semelhança da matriz A. A fase 1 se inicia com a execução da função acessaMatriz, eventos 150 a 174, aproximadamente. A seguir, a partir do evento 220, aproximadamente, podemos visualizar o impacto do acesso por colunas em uma matriz organizada por linha, onde os acessos à coluna 0 são distanciados por 5 endereços, e o mesmo ocorre nos acessos subsequentes, com uma localidade de referência bem inferior à observada na matriz A.



A nossa análise é concluída verificando o gráfico de acesso da matriz C, a seguir, onde podemos verificar a inicialização nula, entre os eventos 100 e 125, durante a fase 0.

A fase 1 se inicia com a execução de acessaMatriz seguida da inicialização de todos os seus elementos com 0. Os eventos 220 a 580 compreendem o cálculo de cada elemento da matriz C, iniciando por C[0,0] e percorrendo linha a linha. Note que para cada posição da matriz são realizadas 5 escritas em sequência, uma para cada soma do produto de elementos da matriz A e B.

A fase 2, quando a matriz C é impressa, é representada por novo acessaMatriz e a impressão propriamente dita no intervalo de eventos entre 600 e 650.



### 6.1.3 Transposição

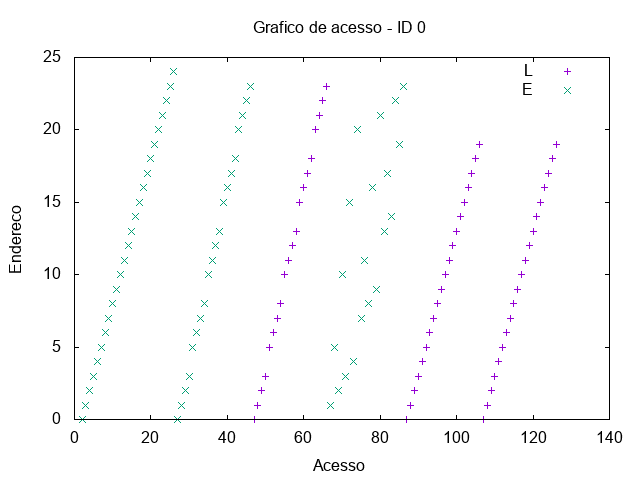
A transposta de uma matriz A é uma matriz que apresenta os mesmos elementos de A invertidos em termos de linhas e colunas. Ela é obtida movendo ordenadamente os elementos das linhas de A para as colunas da transposta, ou seja, para cada elemento *A[i,j]* será colocado em *AT[j,i].*

Observando o gráfico de acesso da transposição de uma matriz 5x4, criada a partir de uma matriz estática 5x5, que vai resultar em uma matriz 4x5, podemos visualizar o padrão de acesso das 3 fases.

A fase 0, inicialização, compreende os eventos até 40, aproximadamente, quando a matriz é inicializada com 0 e depois com valores aleatórios.

A fase 1 é a transposição propriamente dita, que se inicia com a função acessaMatriz (eventos 45 a 60), seguido da transposição. Podemos perceber os acessos à linha 0 e coluna 0 de forma intercalada, quando ambos os elementos são escritos na posição do outro. O processo continua linha a linha e coluna a coluna, mas cada vez com menos elementos, tendo em vista a transposição tomar como base duas matrizes triangulares.

A fase 2, quando a matriz transposta é impressa, é caracterizada pelo acessaMatriz seguido da impressão propriamente dita. É interessante notar que o padrão de acesso da função acessaMatriz da fase 2 difere da fase 1, tendo em vista a mudança das dimensões da matriz com a operação de transposição.



## 6.2 Distância de Pilha

Para analisar as distâncias de pilha vamos nos concentrar apenas na fase 1 da multiplicação de matrizes, que apresenta algumas nuances interessantes.

A análise de distâncias de pilha é realizada com base em dois gráficos:

* **Histograma de distância de pilha**:

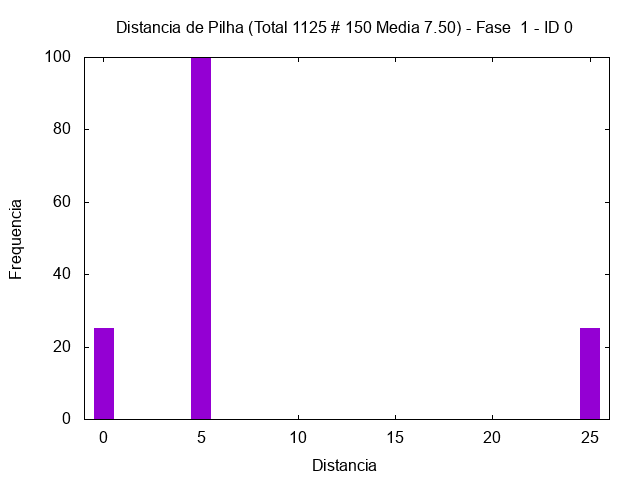
O histograma da distância de pilha mostra, para cada estrutura de dados (i.e., id) e fase, a frequência de ocorrência da distância de pilha, desconsiderando os acessos iniciais, ou seja, aqueles quando o endereço é acessado a primeira vez e colocado na pilha de endereços.

* **Evolução da distância de pilha**:

A evolução da distância de pilha mostra, para cada evento de acesso para o qual a distância de pilha foi registrada, qual foi a distância de pilha computada. Nesse caso, devemos considerar conjuntamente o histograma de distância de pilha e o gráfico de acesso para melhor entendimento.

A seguir vamos analisar o padrão de acesso da fase 1 da multiplicação de matrizes, cujos gráficos de acesso tiveram comportamentos bastante diferentes.

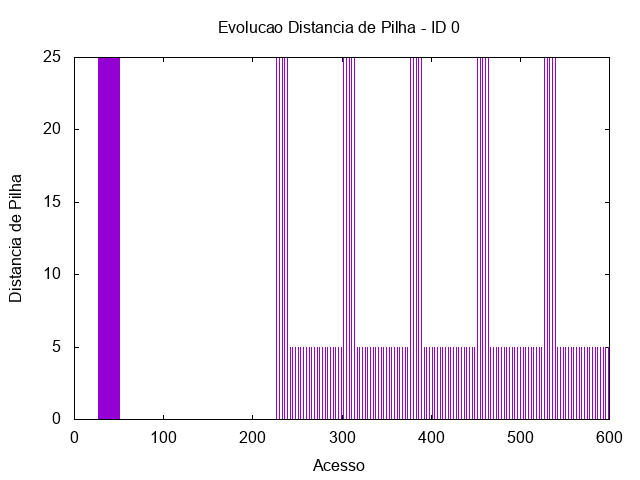
O gráfico a seguir apresenta o histograma de distância de pilha da fase 1 da matriz A (id 0). Temos basicamente a ocorrência de 3 barras de frequência. A primeira barra está associada à distância 0, relativa à função acessaMatriz. A segunda barra está associada à distância 5, e representa os consecutivos acessos à cada linha. Já a terceira barra, associada à distância 25 é associada ao primeiro acesso a cada coluna e será explicada a seguir.



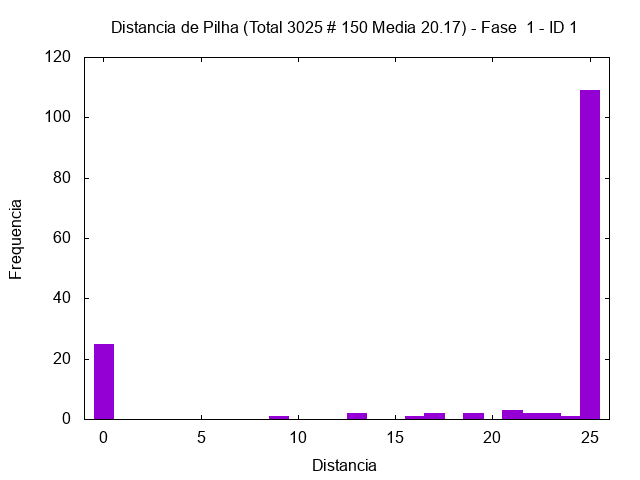
Quando verificamos o gráfico da evolução da distância de pilha entendemos melhor o padrão de acesso que explica as frequências associadas a distâncias de pilha 5 e 25, em particular essa última.

Como mencionado, a função acessaMatriz lê a matriz linha a linha e portanto acessa sequencialmente a matriz A, que está alocada por linha. Ao fim da execução da acessaMatriz, temos 25 elementos na pilha, sendo o topo o elemento A[4,4] e o fundo da pilha o elemento A[0,0]. O processo de multiplicação se inicia pelo elemento A[0,0], cuja distância de pilha é 25. No momento que é acessado, esse elemento é colocado no topo da pilha e o fundo da pilha passa a ser ocupado pelo elemento A[0,1], que é o próximo a ser acessado, também com distância 25. O padrão de acesso continua para o restante da linha, com distâncias de pilha 25 para os elementos A[0,2], A[0,3] e A[0,4]. Nesse momento, o topo da pilha possui a linha 0, mas tomada da direita para a esquerda, ou seja, A[0,0] ocupando a profundidade 5 da pilha (assumindo que o topo tem profundidade 1). A linha 0 então é acessada 4 vezes, e cada um desses acessos tem uma distância pilha 5, pois o elemento acessado é sempre na profundidade 5 da pilha.

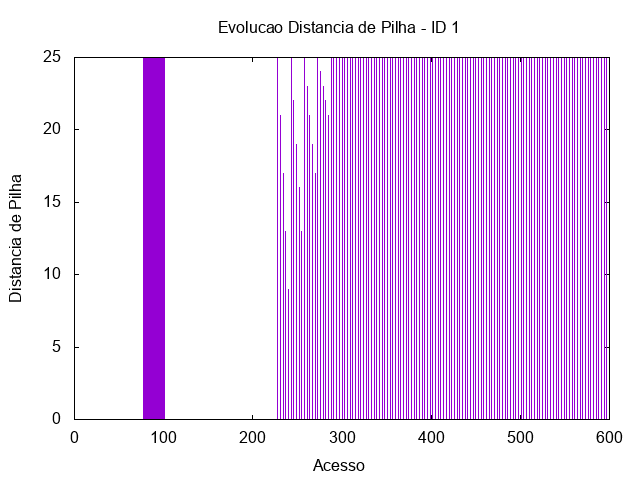
Após esgotarmos o processamento associado à linha 0, iniciamos a linha 1, com 5 distâncias de pilha 25, seguidas de 20 distâncias de pilha 5, o que se repete para as linhas 2, 3 e 4, como mostra o gráfico da evolução da distância de pilha da matriz A, abaixo.



A análise das distâncias de pilha da Matriz B é mais intrigante e será feita a seguir. Quando analisamos o histograma das distâncias de pilha da Matriz B (ID 1), verificamos que a maior parte do histograma pode ser entendido a partir do código, com um grande número de distâncias igual a 25 e um número menor, mas significativo, de distâncias igual a 0. Entretanto, há um número pequeno de ocorrências de várias distâncias de pilha, nominalmente 10, 13, 16, 17, 19, 21, 23 e 24. Como explicá-las?

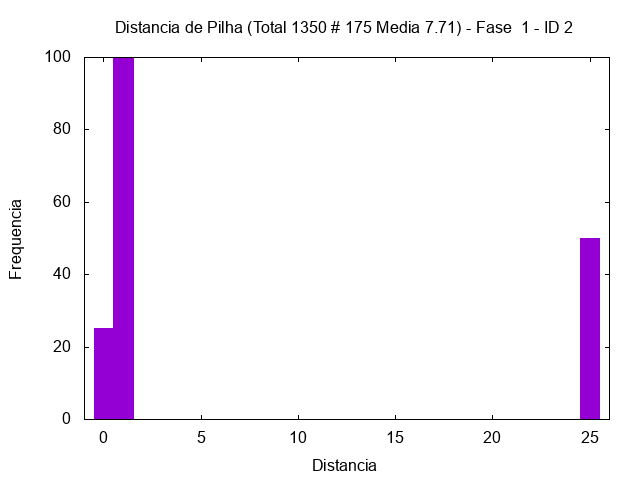


Podemos entender essas distâncias de pilha inesperadas analisando o gráfico de evolução da distância de pilha, apresentado abaixo. Podemos perceber que, conforme esperado, a maior parte das distâncias de pilha na fase 1 (a partir do evento 220, aproximadamente, é igual a 25 e que as ocorrências das distâncias de pilha inesperadas são entre os eventos 220 e 280, aproximadamente.

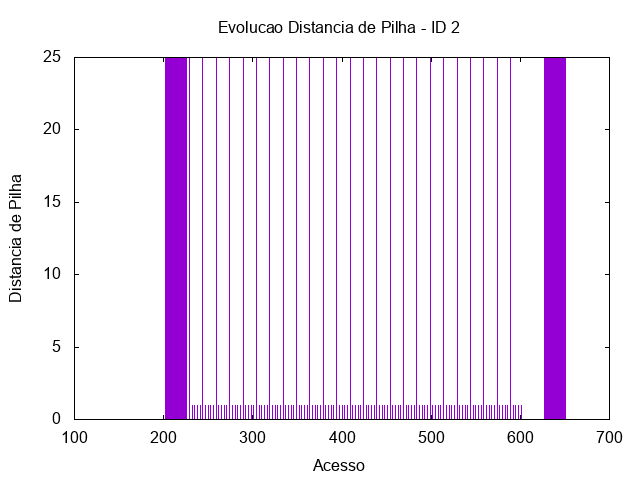


Como mencionado anteriormente, a fase 1 se inicia com uma execução da função acessaMatriz, que organiza os 25 elementos da matriz de tal forma que o topo da pilha é A[4,4] e o fundo da pilha é A[0,0]. Assim, quando acessamos o elemento A[0,0], temos uma distância de pilha igual a 25, pois ele está no fundo da pilha. O próximo elemento a ser acessado é A[1,0], que ocupava, após a execução de acessaMatriz, a profundidade 20. Entretanto, quando A[0,0] é acessado, ele é colocado no topo e A[1,0] passa a ter profundidade 21, conforme mostra o gráfico. O terceiro acesso é A[2,0], que inicialmente tinha profundidade 15, mas alcança 17 com os acessos a A[0,0] e A[1,0]. A seguir temos A[3,0], que tinha profundidade 10, mas assume profundidade 13 após os 3 acessos que a precedem, e A[4,0], que tinha profundidade 5, mas assume profundidade 9 após os 4 acessos que precedem. Isso conclui o processamento da primeira coluna. Quando acessamos a segunda coluna (A[0,1]), ela está no fundo da pilha e a distância de pilha volta a ser 25. Os valores de distância de pilha que se seguem: 22, 19, 16 e 13 são explicados por lógica similar à apresentada para a primeira coluna, assim como para as demais colunas. É interessante notar que, a partir do segundo acesso à primeira coluna, todas as distâncias de pilha são iguais a 25, conforme esperado, pois a pilha já se reorganizou para o acesso à matriz B por colunas.

Para concluir a nossa análise, vamos verificar o histograma de distâncias de pilha da matriz C (ID 2). Verificamos que a maior parte dos acessos é associado a uma distância de pilha 1, havendo ainda um número significativo de acessos com distância 25. Para entendermos melhor esse histograma, vamos recorrer novamente ao gráfico de evolução da distância de pilha, como discutido a seguir.



Verificamos que há dois grupos de acessos com distância de pilha 25 (eventos 200-225 e 635-650, aproximadamente), que correspondem à inicialização com valores nulos e impressão da matriz resultante. Já os eventos entre 225 e 600 compreendem a multiplicação propriamente dita. Percebemos que há 25 ocorrências de distância de pilha 25, que é quando o elemento da matriz é acessado pela primeira vez para a soma de um produto interno. A seguir temos, para cada elemento, 4 eventos com distância de pilha 1, que são as somas que se seguem para o mesmo elemento, que já estava no topo da pilha.



Desta forma, utilizando os gráficos providos pelo aplicativo analisamem, fomos capazes de entender a dinâmica de acesso à memória do programa completamente.

## 6.3 Uso da Análise de Localidade de Referência

Nessa seção apresentamos algumas possibilidades do uso da análise de localidade de referência para entendimento e melhoria dos programas.

### 6.3.1 Entendimento

As análises apresentadas nas seções 6.1 e 6.2 ilustram como os recursos providos pelo analisamem podem ser utilizados para entender os padrões de acesso àos vários TADs. Esse entendimento é importante para confirmar as expectativas quando do projeto e implementação dos vários algoritmos, ao mesmo tempo que se avalia o impacto de efeitos dinâmicos, inerentes à execução do programa. O fenômeno relatado na seção 6.2, quando foram detectadas várias distâncias de pilha para a matriz B é um exemplo de fenômeno a ser entendido. O próprio padrão de acesso, analisado com base no código que o gerou, é fundamental para esse entendimento.

### 6.3.2 Depuração

Uma vez entendido o comportamento de um programa em relação ao padrão de acesso e consequente localidade de referência, essa informação pode subsidiar pelo menos duas formas de depurar o programa:

* **Reorganizar estruturas de dados**

A reorganização da estrutura de dados tem por objetivo melhorar a localidade de referência ao dados. Isso pode incluir estruturas auxiliares, como índices, ou o alinhamento dos dados com as dimensões das linhas de cache e outras estruturas de memória, evitando a fragmentação (por exemplo, o acesso a um elemento da estrutura de dados obriga a carga de duas linhas de cache, o que é claramente ineficiente).

* **Refatorar código**

O código pode ser refatorado com vistas a melhorar a localidade de referência, explorando ou não a reorganização das estruturas de dados e o padrão de acesso requerido pelo algoritmo.

Em ambos os casos, é necessário que o aperfeiçoamento seja avaliado experimentalmente, demonstrando os benefícios da mudança proposta.