C/OMP - Cálculos Estatísticos sobre Dados de Experimentos

Um pesquisador possui um conjunto de amostras agrupadas em colunas de uma matriz de dados (cada coluna representa as amostras de um experimento realizado). Há ***n*** amostras em cada experimento e há um total de ***E*** experimentos. Os dados coletados nos experimentos são inteiros que variam entre 0 e 99. O pesquisador precisa calcular, para cada experimento (cada coluna) as seguintes métricas: **média aritmética, média harmônica, mediana, moda, variância amostral, desvio padrão** e **coeficiente de variação**. Cada métrica pode ser definida da seguinte forma:

Média aritmética: Somatório de todos os elementos da amostra, divididos pelo tamanho da amostra (somas das linhas da coluna da matriz divididos pela quantidade de linhas);

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

Média harmônica: Razão entre o tamanho da amostra e o somatório do inverso das amostras:

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Mediana: Elemento médio da amostra (elemento médio da **coluna ordenada**). Para um número par de elementos, a mediana de um vetor ***vet*** seria a média entre os elementos do meio ***((vet[n/2]+vet[n/2+1])/2)***. Em outras palavras seriam as posições do vetor:

$$\frac{n+1}{2} \quad \text{ou} \quad \frac{n}{2} \text{ e } \frac{n}{2}+1$$

Moda: Elemento mais frequente da amostra (elemento que mais aparece na coluna, se houver mais de um, considera-se somente o primeiro. Se não houver, retorna -1).

Variância amostral: Soma dos quadrados das diferenças entre o elemento da amostra e a **média aritmética** calculada, dividido pelo tamanho da amostra menos 1 unidade.

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}$$

Desvio padrão: Raiz quadrada da **variância**.

$$S = \sqrt{S^2}$$

Coeficiente de variação: Razão entre o **desvio padrão** e a **média aritmética**.

$$CV = \frac{s}{\overline{x}}$$

Considere este exemplo com 04 experimentos (E = 4) e 06 amostras por experimento (***n*** = 6). Dada a matriz A[6,4]:

9 8 4 5
4 12 20 40
8 8 4 4
8 12 4 21
33 44 20 1
10 18 17 10

O resultado que o pesquisador deseja é o conjunto das métricas calculadas em cada coluna (ou experimento), como este:
Média aritmética: 12.0, 17.0, 11.5, 13.5
Média harmônica: 8.1, 12.1, 6.6, 3.7
Mediana: 8.5, 12.0, 10.5, 7.5
Moda: 8.0, 8.0, 4.0, -1.0
Variância amostral: 110.0, 188.4, 68.7, 217.9
Desvio Padrão: 10.5, 13.7, 8.3, 14.8
Coeficiente de variação: 0.9, 0.8, 0.7, 1.1

O objetivo deste exercício é desenvolver um algoritmo em C e OpenMP, seguindo parte do projeto PCAM já realizado antes.
Use as diretivas task e simd no seu algoritmo.
Ao final desta descrição do exercício há um código sequencial para suporte a este exercício em OMP.

O algoritmo a ser desenvolvido executará em uma máquina MIMD com memória compartilhada (uma máquina multicore). Os números de experimentos, amostras e núcleos de processamento podem ser grandes, porém, assume-se que todas as amostras podem ser armazenadas na memória da máquina, no mínimo uma vez.

A entrada será dada por um arquivo texto contendo, na primeira linha, a quantidade de linhas e colunas da matriz, separadas por um único espaço. Nas linhas posteriores, estão os elementos da matriz do tipo ***int***, onde as linhas são separadas por uma quebra de linha simples e as colunas por um único espaço. A matriz deve ser lida por meio do redirecionamento de fluxo de entrada (usando o redirecionador “<”). Em outras palavras, não use ponteiros para abrir o arquivo de entrada.

Um exemplo do arquivo de entrada (***entrada.txt***) é:
6 4
9 8 4 5
4 12 20 40
8 8 4 4
8 12 4 21
33 44 20 1
10 18 17 10

Para executar no ***bash***, por exemplo, utilize este padrão:
.\est_desc < entrada.txt <enter>

Obs: na linha de comando acima, considera-se que o programa foi inserido em ***est_desc.c*** e o executável chama-se ***est_desc*** e está no diretório atual. O arquivo de entrada chama-se ***entrada.txt***.

A saída deve ser impressa, utilizando o *output (**stdout**)* padrão, apenas com os elementos correspondentes de cada métrica solicitada, separados por um espaço simples. Haverá um espaço no final da linha após o último número. Cada métrica é separada por uma quebra de linha simples. A impressão deve ser feita considerando uma única casa decimal.

Saída esperada (deve ser usada no seu algoritmo – siga o padrão da saída fornecida no algoritmo sequencial abaixo):
12.0 17.0 11.5 13.5
8.1 12.1 6.6 3.7
8.5 10.0 4.0 12.5
8.0 8.0 4.0 -1.0
110.0 188.4 68.7 217.9
10.5 13.7 8.3 14.8
0.9 0.8 0.7 1.1

Submeta o seu algoritmo (**um por grupo**) na Ferramenta Iguana uma solução por grupo. Indique no início do código, como comentário, os nomes dos integrantes do grupo que de fato fizeram a atividade.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<omp.h>

//Quicksort adaptado de //https://www.geeksforgeeks.org/quick-sort/
int partition (double *arr, int low, int high, int C){
    int i, j;
    double pivot,swap;

    // pivot (Element to be placed at right position)
    pivot = arr[high*C];

    i = (low - 1);  // Index of smaller element

    for (j = low; j <= high-1; j++)
    {
        // If current element is smaller than or
        // equal to pivot
        if (arr[j*C] <= pivot)
        {
            i++;    // increment index of smaller element

            // swap arr[i] and arr[j]
            swap = arr[i*C];
              arr[i*C] = arr[j*C];
              arr[j*C] = swap;
        }
    }

    //swap arr[i + 1] and arr[high]
    swap = arr[(i + 1)*C];
    arr[(i + 1)*C] = arr[high*C];
    arr[high*C] = swap;

    return (i + 1);

} // fim partition


/* low  --> Starting index,  high  --> Ending index */
void quicksort(double *arr, int low, int high, int C){
    int pi;

    if (low < high)  {
        /* pi is partitioning index, arr[pi] is now
           at right place */
```

```
        pi = partition(arr, low, high, C);

        quicksort(arr, low, pi - 1, C);  // Before pi
        quicksort(arr, pi + 1, high, C); // After pi
    }

} // fim quicksort

/* This function takes last element as pivot, places
   the pivot element at its correct position in sorted
    array, and places all smaller (smaller than pivot)
   to left of pivot and all greater elements to right
   of pivot
   https://www.geeksforgeeks.org/quick-sort/
*/

void ordena_colunas(double *matriz, int lin, int col) {
  int j;

  for (j = 0; j < col; j++) {
      //manda o endereco do primeiro elemento da coluna, limites inf e sup e a largura da
matriz
      quicksort(&matriz[j], 0, lin-1, col);
  }
}

void calcula_media(double *matriz, double *vet, int lin, int col){
    int i,j;
    double soma;
    for(i=0;i<col;i++){
        soma=0;
        for(j=0;j<lin;j++){
            soma+=matriz[j*col+i];
        }
        vet[i]=soma/j;
    }
}

void calcula_media_harmonica(double *matriz, double *vet, int lin, int col){
    int i,j;
    double soma;
    for(i=0;i<col;i++){
        soma=0;
        for(j=0;j<lin;j++){
            soma+=(1/(matriz[j*col+i]));
        }
        vet[i]=lin/soma;
```

```
    }
}

void calcula_mediana(double *matriz, double *vet, int lin, int col) {
  int j;
  for (j = 0; j < col; j++) {
    vet[j] = matriz[((lin/2)*col)+j];
    if(!(lin%2))  {
      vet[j]+=matriz[((((lin-1)/2)*col)+j)];
      vet[j]*=0.5;
    }
  }
}

//Adaptado de https://www.clubedohardware.com.br/forums/topic/1291570-programa-em-
c-que-calcula-moda-media-e-mediana/
double moda_aux(double *matriz,int lin){
    int i, j;
    double *cont;
    cont=(double*)malloc(lin*sizeof(double));
          float conta=0, moda;

          for(i=0;i<lin;i++){
        for(j=i+1;j<lin;j++){

                              if(matriz[i]==matriz[j]){
                                    cont[i]++;
                                          if(cont[i]>conta){
                                                conta=cont[i];
                                                moda=matriz[i];
                                          }
                              }

        }
        cont[i]=0;
    }
    free(cont);
    if(conta == 0){
          return -1;
          }
          else{
                    return moda;
          }

}
```

```
void calcula_moda(double *matriz,double *moda,int lin, int col){
    int i,j;
    double *aux=(double *)malloc(lin*sizeof(double));
    for(i=0;i<col;i++){
        for(j=0;j<lin;j++)
        {
            aux[j]=matriz[j*col+i]; //Faz a transposta da linha para coluna
        }
        moda[i]=moda_aux(aux,lin);
    }
    free(aux);

}

void calcula_variancia(double *matriz, double *media,double *variancia, int lin, int col)
{
    int i,j;
    double soma;
    for(i=0;i<col;i++){
        soma=0;
        for(j=0;j<lin;j++){
            soma+=pow((matriz[j*col+i]-media[i]),2);
        }
        variancia[i]=soma/(lin-1);
    }
}

void calcula_desvio_padrao(double *variancia,double *dp, int col)
{
    int i;
    for(i=0;i<col;i++){
        dp[i]=sqrt(variancia[i]);
    }
}

void calcula_coeficiente_variacao(double *media,double *dp,double *cv, int col)
{
    int i;
    for(i=0;i<col;i++){
        cv[i]=dp[i]/media[i];
    }
}


int main(int argc,char **argv){
    int lin,col,i,j; //Define as variáveis de índices e dimensões
    double *matriz,*mediana,*media,*media_har,*moda,*variancia,*dp,*cv; //Define a matriz
```

```
(forma linear), vetores de medidas estatísticas
    fscanf(stdin, "%d ", &lin); //Lê a quantidade de linhas da matriz
    fscanf(stdin, "%d\n", &col); //Lê a quantidade de colunas da matriz
    matriz=(double *)malloc(lin*col * sizeof(double)); //Aloca a matriz
    media=(double *)malloc(col * sizeof(double)); //Aloca o vetor de media
    media_har=(double *)malloc(col * sizeof(double)); //Aloca o vetor de media harmônica
    mediana=(double *)malloc(col * sizeof(double)); //Aloca o vetor de mediana
    moda=(double *)malloc(col * sizeof(double)); //Aloca o vetor de moda
    variancia=(double *)malloc(col * sizeof(double)); //Aloca o vetor de variância
    cv=(double *)malloc(col * sizeof(double)); //Aloca o vetor de coeficiente de variação
    dp=(double *)malloc(col * sizeof(double)); //Aloca o vetor de desvio padrão
    for(i=0;i<lin;i++){
        for(j=0;j<col;j++){
            fscanf(stdin, "%lf ",&(matriz[i*col+j])); //Lê a matriz
        }
    }
    calcula_media(matriz,media,lin,col);
    calcula_media_harmonica(matriz,media_har,lin,col);
    ordena_colunas(matriz,lin,col);
    calcula_mediana(matriz,mediana,lin,col);
    calcula_moda(matriz,moda,lin,col);
    calcula_variancia(matriz,media,variancia,lin,col);
    calcula_desvio_padrao(variancia,dp,col);
    calcula_coeficiente_variacao(media,dp,cv,col);
    for(i=0;i<col;i++)
        printf("%.1lf ",media[i]);
    printf("\n");
    for(i=0;i<col;i++)
        printf("%.1lf ",media_har[i]);
    printf("\n");
    for(i=0;i<col;i++)
      printf("%.1lf ",mediana[i]);
    printf("\n");
    for(i=0;i<col;i++)
      printf("%.1lf ",moda[i]);
    printf("\n");
    for(i=0;i<col;i++)
        printf("%.1lf ",variancia[i]);
    printf("\n");
    for(i=0;i<col;i++)
        printf("%.1lf ",dp[i]);
    printf("\n");
    for(i=0;i<col;i++)
        printf("%.1lf ",cv[i]);
    free(matriz); //Desaloca a matriz
    free(media); //Desaloca o vetor de media
    free(media_har); //Desaloca o vetor de media_har
```

```
    free(mediana); //Desaloca vetor de mediana
    free(moda); //Desaloca vetor de moda
    free(variancia);  //Desaloca vetor de variância
    free(dp); //Desaloca vetor de desvio padrão
    free(cv); //Desaloca vetor de coeficiente de variação
}
```