Projeto e Análise de Algoritmos
Engenharia da Computação - 2023.1 - Prof. Philippe Leal
Lista de Exercícios - Encontro 01 (02/06/2023)

1) Considere a matriz $A = [a_{ij}]_{n \times m}$, onde $n = 4$ e $m = 5$, com número inteiros gerados aleatoriamente de 1 até 20. Faça um algoritmo para gerar a matriz A e verificar se ela satisfaz a seguinte condição:

$$\underset{1 \leq j \leq m}{\text{Min}} \sum_{i=1}^{n} |a_{ij}| \quad \leq \quad \underset{1 \leq i \leq n}{\text{Max}} \prod_{j=1}^{m} a_{ij}$$

Crie e utilize uma **função** para gerar a matriz e outra para realizar a verificação. De acordo com o retorno da função de verificação, deve-se imprimir na função *main*: "Condicao Satisfeita" ou "Condicao Nao Satisfeita".

Obs.: Não é permitido utilizar qualquer estrutura de dados para auxiliar.

2) Considere uma matriz M de ordem 4 de números inteiros gerados aleatoriamente de 0 até 29. Faça um algoritmo para gerar esta matriz e imprimir na tela se ela é ou não uma **Matriz Ortogonal**.

Crie e utilize quatro **funções**: uma para gerar a matriz M, outra para calcular a sua Matriz Transposta (M^T), outra calcular a multiplicação $M \times M^T$ e a quarta para retornar se a matriz M é Ortogonal ou não. A impressão desta informação tem que ser na função *main*.

Obs.: Se uma matriz quadrada M é uma matriz ortogonal, então $M \times M^T = I$, onde M^T é a Matriz Transposta de M e I a Matriz Identidade.

3) Considere um vetor que armazena 10 números inteiros pares e 10 números inteiros ímpares todos embaralhados, ou seja, sem qualquer ordem preestabelecida. Faça um algoritmo para ler este vetor do teclado e depois organizá-lo de modo que os números **pares** fiquem nas posições **ímpares** do vetor e os números **ímpares** fiquem nas posições **pares** do vetor.

Crie e utilize duas **funções**: uma para preencher o vetor pelo teclado e o outra para organizá-lo.

Obs.: Não é permitido utilizar qualquer estrutura de dados para auxiliar a organização.

IMPORTANTE

1) Esta atividade deve ser feita **individualmente**;

2) **TODOS OS EXERCÍCIOS TÊM QUE SER FEITOS NO PAPEL**;

3) Cada aluno(a) deve enviar a imagem (de boa qualidade) do exercício 01 até às **17h20** do dia **02/06/2023** com o seguinte **Assunto** e **Nome do Arquivo**:

PAA-Encontro01-A-Exe01-SeuNome

4) Cada aluno(a) deve enviar a imagem (de boa qualidade) dos exercícios 02 e 03 até às **23h59** do dia **09/06/2023** com o seguinte **Assunto** e **Nome do Arquivo**:

PAA-Encontro01-A-Exe02e03-SeuNome

5) Os arquivos .c dos exercícios devem ser enviados para o e-mail:

`philippeleal@yahoo.com.br`

6) Após a hora e a data marcada para o envio da resposta, **NÃO É MAIS PERMITIDO ENVIÁ-LA**;

7) O e-mail considerado para correção será o **ÚLTIMO** enviado pelo(a) aluno(a) **dentro do prazo determinado**;

8) E-mails com o Assunto fora do padrão **NÃO SERÃO ACEITOS**.

Questão 01.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

/* Definições de tamanho de matriz e faixa de números aleatórios */
#define N 4
#define M 5
#define MIN 1
#define MAX 20

/* Função para gerar uma matriz de tamanho NxM preenchida com números aleatórios de 1 a 20 */
void generateMatrix(int matrix[N][M]){
    /* Semeia a função rand com o tempo atual para garantir que os números gerados sejam diferentes a cada execução do programa */
    srand(time(NULL));
    for(int i = 0; i < N; i++){
        for(int j = 0; j < M; j++){
            /* Preenche a posição atual da matriz com um número aleatório de 1 a 20 */
            matrix[i][j] = (rand() % (MAX)) + MIN;
            printf("%d ", matrix[i][j]);

        }
        printf("\n");
    }
}

/* Função para verificar se a matriz satisfaz a condição especificada */
int verifyCondition(int matrix[N][M]){
    int minSum = INT_MAX, maxMult = INT_MIN;

    for(int i = 0; i < N; i++){
        int sum = 0;
        for(int j = 0; j < M; j++){
            sum += abs(matrix[i][j]);
        }
        /* Se a soma da linha atual é menor que o mínimo encontrado até agora, atualiza o mínimo */
        if(sum < minSum){
            minSum = sum;
        }
    }

    printf("\nSomatorio Minimo dentre todas as linhas:%d\n", minSum);

    for(int i = 0; i < M; i++){
        int mult = 1;
        for(int j = 0; j < N; j++){
            mult *= matrix[j][i];
        }
        /* Se a multiplicação da coluna atual é maior que o máximo encontrado até agora, atualiza o máximo */
        if(mult > maxMult){
            maxMult = mult;
        }
    }

    printf("\nProdutorio Maximo dentre todas as colunas:%d\n", maxMult);

```

```c
    /* Retorna 1 (verdadeiro) se o maior produto das colunas é maior ou igual à menor soma das linhas.
       Caso contrário, retorna 0 (falso). */
    return maxMult >= minSum;
}

int main(){

    printf("\n\nXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n\n");

    int matrix[N][M];
    generateMatrix(matrix);

    if(verifyCondition(matrix)){
        printf("\nCondicao satisfeita.\n");
    }
    else{
        printf("\nCondicao nao satisfeita.\n");
    }


    printf("\n\nXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n\n");
    return 0;
}

```

Questão 02.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LINHAS 4
#define COLUNAS 4

void geraMatriz(int matriz[LINHAS][COLUNAS]) {
  srand(time(NULL));
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      matriz[i][j] = rand() % 100;
    }
  }
}

void geraMatrizTransposta(int matriz[LINHAS][COLUNAS], int matrizTransposta[LINHAS][COLUNAS]) {
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      matrizTransposta[j][i] = matriz[i][j];
    }
  }
}

void multiplicaMatriz(int matriz[LINHAS][COLUNAS], int matrizTransposta[LINHAS][COLUNAS], int
matrizResposta[LINHAS][COLUNAS]) {
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {

      matrizResposta[i][j] = 0;

      for (int k = 0; k < LINHAS; k++) {
        matrizResposta[i][j] += matriz[i][k] * matrizTransposta[k][j];
      }

    }
  }
}

int verificaOrtogonal(int matrizResposta[LINHAS][COLUNAS]) {
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      if(i==j) {
        if(matrizResposta[i][j] == 1)
        continue;
        else
        return 0;
      } else {
          if(matrizResposta[i][j] == 0)
          continue;
          else
          return 0;
          }

    }
  }
  return 1;
}
```

```c

int main() {

  int matriz[LINHAS][COLUNAS];
  int matrizTransposta[LINHAS][COLUNAS];
  int matrizResposta[LINHAS][COLUNAS];

  //Função 1 -> Criar Matriz:
  geraMatriz(matriz);

  //Função 2 -> Calcular Transposta:
  geraMatrizTransposta(matriz, matrizTransposta);

  //Função 3 -> Multiplicar Matrizes:
  multiplicaMatriz(matriz, matrizTransposta, matrizResposta);

  //Função 4 -> Verificar se a matriz é Ortogonal:
  int resposta = 0;
  resposta = verificaOrtogonal(matrizResposta);

  //Exibindo Matriz:
  printf("\n\n------------------------Matriz Gerada Aleatoriamente----------------------\n");
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      printf(" %02d ", matriz[i][j]);
    }
    printf("\n");
  }

  //Exibindo Matriz Transposta:
  printf("-----------------------------Matriz Transposta-----------------------------\n");
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      printf(" %02d ", matrizTransposta[i][j]);
    }
    printf("\n");
  }

  //Exibindo Matriz Resposta:
  printf("-----------------------------Matriz Resposta-------------------------------\n");
  for(int i = 0; i < LINHAS; i++) {
    for(int j = 0; j < COLUNAS; j++) {
      printf(" %05d ", matrizResposta[i][j]);
    }
    printf("\n");
  }

  printf("---------------------------------------------------------------------------\n");

  if(resposta == 1)
    printf("Matriz Ortogonal!");
  else
    printf("Matriz Nao Ortogonal!");

  printf("\n---------------------------------------------------------------------------\n\n");

  return 0;
}
```

Questão 03.c

```c
#include <stdio.h>
#define TAMANHO 20

// Função para preencher o vetor pelo teclado
void preencherVetor(int vetor[TAMANHO]) {
    printf("\n\n");
    printf("Digite os %d numeros do vetor:\n", TAMANHO);
    for (int i = 0; i < TAMANHO; i++) {
        scanf("%d", &vetor[i]);
    }
    printf("\n\n");
}

// Função para organizar o vetor
void organizarVetor(int vetor[TAMANHO]) {
    for (int i = 0; i < TAMANHO; i++) {
        for (int j = i ; j < TAMANHO; j++) {
            if ((i % 2 == 0 && vetor[j] % 2 != 0) || (i % 2 != 0 && vetor[j] % 2 == 0)) {
                int aux = vetor[i];
                vetor[i] = vetor[j];
                vetor[j] = aux;
                break;
            }
        }
    }
}

// Função para imprimir o vetor
void imprimirVetor(int vetor[TAMANHO]) {
    printf("Vetor organizado:\n");
    printf("|");
    for (int i = 0; i < TAMANHO; i++) {
        printf(" %d |", vetor[i]);
    }
    printf("\n\n");
}

int main() {

    int vetor[TAMANHO];

    preencherVetor(vetor);
    organizarVetor(vetor);
    imprimirVetor(vetor);

    return 0;
}

```