Aluno(a):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 1a) | 3.0 |  |
| 2a) | 2.0 |  |
| 3a) | 2.5 |  |
| 4a) | 2.5 |  |
|  | 10.0 |  |

**LEIA COM CUIDADO**

1. A prova é individual e sem consulta.
   1. **Qualquer tentativa de “cola” resultará na anulação da prova do aluno ou dos alunos envolvidos**.
   2. Os aparelhos celulares deverão permanecer desligados e guardados fora do alcance durante toda a prova. **Aparelhos celulares ligados ou de alguma forma visíveis serão tratados como tentativa de “cola”**.
2. A interpretação faz parte da questão.
   1. **Não há perguntas durante a prova.**
   2. Em caso de dúvida: escreva a dúvida; corrija o que você achar necessário; e resolva a questão corrigida.
3. O tempo de prova é 1:45 h.
4. **Após o início da prova, não será possível sair e voltar à sala.**
5. As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
6. A prova pode ser feita a lápis.

**Questão 1 (3.0 pontos)** Implemente uma função em C que receba como entrada um grafo não dirigido *G*, com pelo menos 1 nó, e retorne 1, se *G* for um grafo conexo, e 0, em caso contrário. A função deve ter o seguinte protótipo:

int conexo(Graph\* G)

Considere que o grafo está representado pela sua matriz de adjacências:

typedef struct graph Graph;

struct graph {

int nv; /\* número de nós no grafo \*/

int\*\* adj; /\* matriz de adjacências do grafo \*/

};

A função deverá se basear em algum dos algoritmos de grafos apresentados em sala, ***de menor custo para a tarefa***. Não é necessário incluir o algoritmo na solução, mas indique o protótipo do algoritmo adotado e qualquer outra estrutura de dados necessária para a sua implementação.

Argumente porque a sua implementação da função conexo está ***correta*** e é de ***menor custo*** para a tarefa.

Resposta:

**Questão 2 (2.0 pontos)** Escreva uma rotina em C para codificar um ponto, de um espaço tridimensional, usando a curva de Morton. A rotina deve ter o seguinte protótipo:

long encodeMorton(Ponto\* p)

A rotina:

* recebe como entrada um ponteiro para um ponto p, conforme definido abaixo
* devolve como saída um inteiro longo m, onde
  + m possui 32 bits
  + m codifica o ponto p, da direita para a esquerda, considerando as coordenadas na ordem x, y e z

Assuma, por simplicidade, que um ponto possui coordenadas inteiras, definidas pelas 8 bits mais à direita de números inteiros. O tipo Ponto é definido como

typedef struct ponto Ponto;

struct ponto {

int x; /\* usa apenas os 8 bits mais à direita do int \*/

int y; /\* usa apenas os 8 bits mais à direita do int \*/

int z; /\* usa apenas os 8 bits mais à direita do int \*/

}

Resposta:

**Questão 3 (2.5 pontos)**. Dado um grafo não dirigido, o *coeficiente de Adamic-Adar* de dois nós *x* e *y*, *score(x,y)*, é definido como:

* + - 1. Se Γ(x) ∩ Γ(y)=∅, então *score(x, y)* = 0
      2. Em caso contrário, *score(x, y)* é dado pela equação abaixo:

onde Γ(*w*) denota o conjunto de nós ligados por arestas a um nó *w*, |*s*| denota a cardinalidade de um conjunto *s* e log(x) denota o logaritmo natural de x, ou seja, o logaritmo na base *e*.

Implemente uma função em C que receba como entrada um grafo não dirigido *G* e dois nós, *x* e *y*, do grafo *G* e retorne o coeficiente de Adamic-Adar de *x* e *y*. Considere que o grafo está representado por listas de adjacências:

typedef struct graph Graph;

typedef struct listNode ListNode;

struct listNode {

int vertex;

ListNode\* link;

};

struct graph {

int nv; /\* número de vértices no grafo \*/

ListNode\*\* vv; /\* vv[i] aponta p/ lista de vértices adjacentes a i \*/

};

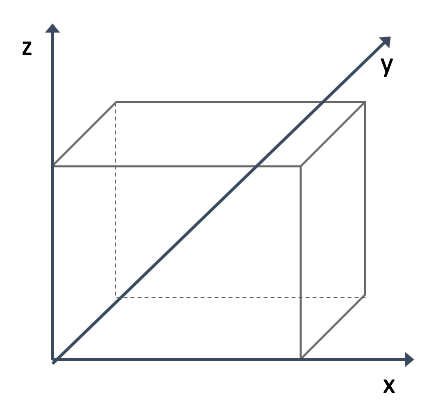
A função deve ter o seguinte protótipo:

float Adamic-Adar(Graph\* G, int x, int y);

onde x e y são dois inteiros indicando a posição dos nós no vetor vv.

Nota: use a função logf(x)para computar o valor do logaritmo natural do argumento x. Esta função recebe e devolve um float.

Resposta:

**Questão 4 (2.5 pontos).** Considere uma grade não regular tridimensional tal que as fronteiras lateral esquerda, inferior e frontal não pertencem à grade; para cada célula, as fronteiras lateral esquerda, inferior e frontal não pertencem à célula.

Cada célula da grade armazena uma lista encadeada de pontos. Considere os seguintes tipos que representam a grade:

typedef struct lista Lista;

struct lista {

float x, y, z; /\* ponto na lista \*/

Lista\* prox; /\* ponteiro para próximo elemento da lista \*/

};

typedef struct grade Grade;

#define NX 200 /\* número de elementos do vetor gX \*/

#define NY 135 /\* número de elementos do vetor gY \*/

#define NZ 400 /\* número de elementos do vetor gZ \*/

struct grade {

float gX[NX]; /\* vetor que define as divisões no eixo X da grade \*/

float gY[NY]; /\* vetor que define as divisões no eixo Y da grade \*/

float gZ[NZ]; /\* vetor que define as divisões no eixo Z da grade \*/

Lista\* prim[NX][NY][NZ]; /\* lista por célula (inicializada com NULL) \*/

};

Implemente uma função que receba como entrada uma grade *g*, com a definição acima, e um ponto *(px, py, pz)* e retorne

0 se o ponto não foi encontrado na grade

1 se o ponto foi encontrado na grade

-1 se o ponto está fora da grade

A função deve seguir o seguinte protótipo:

int ocorre(Grade\* g, float x, float y, float z);

Resposta: