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

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

1. A prova é individual e sem consulta. Qualquer tentativa de “cola” resultará na anulação da prova de ambos os alunos envolvidos.
2. A interpretação faz parte da questão. Em caso de dúvida escreva a dúvida e a sua interpretação na resposta.
3. O tempo de prova é 1:45 h.
4. As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
5. A prova pode ser feita a lápis.

1) (2.5 pontos) Escreve uma rotina em C para inserção de chaves, definidas como números inteiros, em uma tabela de dispersão com endereçamento aberto e função de dispersão h definida como

h(K, i) = ((K % N) + i) % N

onde

K = chave a ser inserida

N = tamanho da tabela de dispersão

i = 0, 1, 2, ... = número da tentativa de inserção da chave na tabela

Assuma que as posições são preenchidas, inicialmente, com o valor -1 e que a tabela seja declarada como:

#define N 1007; /\* tamanho da tabela de dispersão \*/

#define VAZIO (-1); /\* indica que a posição está desocupada \*/

int tabela[N]; /\* tabela de dispersão \*/

A função deve ter o seguinte protótipo:

int insere(int K)

e deve retornar:

-1 se a chave não puder ser inserida

0 se a chave já existe

1 se a chave for inserida corretamente

Resposta:

2) (2.5 pontos) Considere a seguinte árvore B de ordem 5 (todo nó, exceto a raiz, tem nó mínimo 2 chaves e no máximo 4 chaves):

100

53 77 123 200

10 40 60 70 80 90 110 113 130 138 140 170 230 236 242 245

Realize as seguintes operações, utilizando sempre a árvore resultante da operação anterior. Indique os nós que sofrem modificações após cada operação, bem como a ocorrência de divisão, redistribuição ou concatenação:

1. (0.5 ponto) Inserção de 243.
2. (1.0 ponto) Remoção de 40.
3. (1.0 ponto) Remoção de 100.

Resposta:

3) (2.5 pontos) Um nó *m é vizinho* de um nó *n* sse há uma aresta entre *m* e *n* no grafo. O *clustering coefficient* de um nó *n* é definido como o número de arestas existentes entre os vizinhos de *n* dividido pelo número de todas as possíveis arestas entre os vizinhos de *n*; se *n* tem *0* ou *1* vizinho, então o *clustering coeficiente* é *0.*

Implemente uma função em C que receba como entrada um grafo não dirigido e um nó do grafo e retorne o *clustering coefficient* do nó no grafo. A função deve ter o seguinte protótipo:

int cc(Graph\* G, int n);

Considere que o grafo está representado por listas de adjacências:

typedef struct graph Graph;

typedef struct listNode ListNode;

struct listNode {

int node;

ListNode\* link;

};

struct graph {

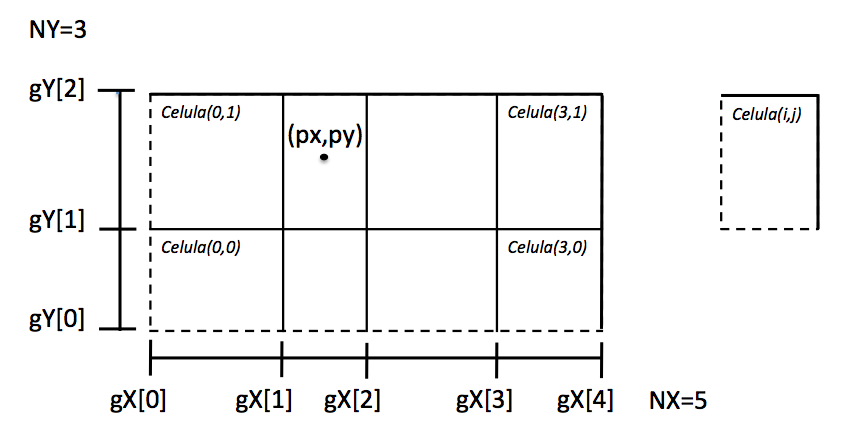
int nv; /\* número de nós no grafo \*/

ListNode\*\* vv; /\* vv[i] aponta para os vizinhos do nó i \*/

};

Resposta:

4) (2.5 pontos) Considere uma grade não regular bidimensional tal que (ver figura abaixo): as bordas lateral esquerda e inferior não pertencem à grade; para cada célula, as bordas lateral esquerda e inferior não pertencem a cada 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; /\* 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 \*/

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 \*/

Lista\* prim[NX][NY]; /\* 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)*, como na figura, e insira o ponto na grade, retornando 1 se a inserção foi bem sucedida, e 0, em caso contrário. A função deve seguir o seguinte protótipo:

int insere(Grade\* g, float px, float py);

e deve retornar:

0 se o ponto não puder ser inserido

1 se o ponto for inserido corretamente

Resposta: