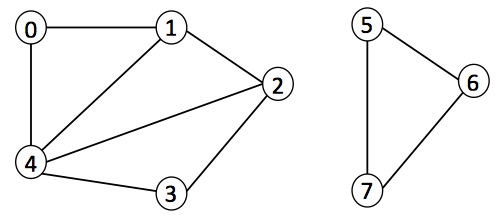
Aluno(a):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 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.
6. (2.5 pontos) O *giant coefficient* de um grafo não dirigido é definido como o número de vértices da maior componente conexa do grafo dividido pelo total de vértices do grafo.
   1. (0,5 ponto) Compute o *giant coefficient* do grafo abaixo. Mostre os passos da computação.



Resposta:

* 1. (2,0 pontos) Implemente uma função em C que receba como entrada um grafo não dirigido e retorne o *giant coefficient* do grafo. A função deve ter o seguinte protótipo:

int gc(Graph\* G);

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 \*/

int\* vis; /\* vis[i]=0 sse vertice i não foi visitado \*/

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

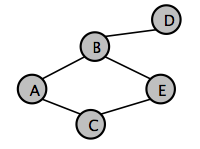
};

Resposta:

1. (2.5 pontos) O *diâmetro* de um grafo não dirigido é definido como o maior valor da menor distância entre quaisquer pares de vértices pertencentes ao grafo.
   1. (2,0 ponto) Usando uma das funções para grafos descritas em sala como subrotina, descreva em pseudo-código um algoritmo para computar o diâmetro de um grafo.

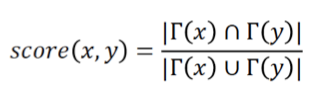
Resposta:

* 1. (0,5 ponto) Usando o algoritmo definido no item (a), mostre todos os passos para computar o diâmetro do grafo abaixo, inclusive os passos da subrotina.



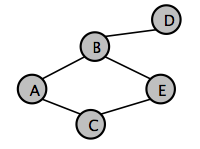
Resposta:

1. (2.5 pontos) Dado um grafo não dirigido, o *coeficiente de Jaccard* de dois nós *x* e *y*, *score(x,y)*, é definido como:



onde Γ(*n*) denota o conjunto de nós ligados por arestas a um nó *n* e |*s*| denota a cardinalidade de um conjunto *s*.

1. (0,5 ponto) Compute *score(A,D)* e *score(A,E)* no grafo abaixo.



Resposta:

1. (2,0 pontos) 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 Jaccard 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 \*/

};

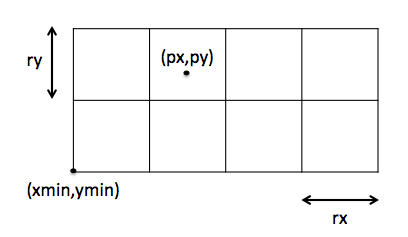
A função deve ter o seguinte protótipo:

int jaccard(Graph\* G, x, y);

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

Resposta:

1. (2,5 ponto): Considere uma grade regular bidimensional com *NX* células na horizontal e *NY* células na vertical. O tamanho de cada célula é dado por *rx* na horizontal e *ry* na vertical e o ponto inferior esquerdo da grade está localizado na posição (*xmin, ymin*), como apresentado esquematicamente na figura abaixo:



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

#define NY 135

struct grade {

float rx; /\* tamanho da dimensão x da célula \*/

float ry; /\* tamanho da dimensão y da célula \*/

float xmin, ymin; /\* posição mínima 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 retorne 1, se o ponto ocorre na grade, e 0, em caso contrário. A função deve seguir o seguinte protótipo:

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

Resposta: