Aluno(a):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ matrícula:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 1a) | 1.0 |  |
| 2a) | 2.0 |  |
| 3a) | 2.0 |  |
| 4a) | 2.0 |  |
| 5a) | 3.0 |  |
|  |  |  |

1. A prova é individual e sem consulta.
2. A interpretação faz parte da questão.
3. O tempo de prova é 1:30 h.
4. As respostas devem seguir as questões.
5. Caso precise de rascunho use o verso da folha.
6. Caso parte da resposta esteja no verso, indique claramente este fato.
7. (1.0 ponto) Faça a representação como árvore binária da seguinte árvore n-ária.



J

Resp.:

Uma modificação direta daria em:



J

rotacionando para a visualização teríamos:



J

1. (2.0 pontos) Escreva em C uma função para determinar se uma árvore é AVL. A função deve receber como parâmetro o endereço do nó raiz da árvore e retornar o valor 1 se for AVL e 0 caso contrário. Use a seguinte estrutura:

|  |
| --- |
| **struct** no {  **void** \*chave;  **struct** no \*esq, \*dir;  }; |

Resp.:

#include <stdlib.h>

struct no {

void \*chave;

struct no \*esq, \*dir;

};

int checkAVL( no \*node ) {

if( node==NULL ) return 0;

int esq = checkAVL( node->esq );

int dir = checkAVL( node->dir );

if( ( dir < 0 ) || ( esq < 0 ) ) return -1;

if( abs( dir - esq ) > 1 ) return -1;

return( max( abs( dir ), abs( esq ) ) + 1 );

}

int AVL( no \*node ) {

if( checkAVL( node ) >= 0 ) return 1;

else return 0;

}

1. (2.0 pontos)

a. (1.0 ponto) Qual é o número máximo de chaves que uma árvore B de ordem *m* (ou seja, todo nó tem nó máximo *m* filhos e *m-1* chaves) e altura *h* pode armazenar? Explique sua resposta.

Dica: A soma dos *n* primeiros termos de uma série geométrica é dada por:

*a + ar + ar2 + ar3+ ...+ arn-1 = a*

b. (1.0 ponto) Qual será a altura de uma árvore B de ordem 50 que armazenar 100.000 chaves? Explique sua resposta.

Resp.:

a) Uma árvore de ordem *m* e altura *h* pode ter no máximo *N* chaves, onde

*N = (m-1)\*(1 + m + m2 + m3+ ... + mh) = (m-1) = mh+1 - 1*

a) Para uma árvore de ordem 50 armazenar 10.000 chaves deverá ter altura *h*, onde *h* é o menor inteiro tal que:

10000 ≤ 50*h+1 – 1*

10001 ≤ 50*h+1*

Logo, *h =* 2

1. (2.0 pontos)

a. (1.0 ponto) Crie uma árvore B de ordem 4 (ou seja, todo nó tem nó máximo 4 filhos e 3 chaves) através da inserção das chaves exatamente na ordem:1, 3, 6, 8, 14, 32, 38. Mostre todos os passos.

b. (1.0 ponto) Mostre todos os passos para remover a chave 8.

Resp.: (a) Inserções de 1, 3, 6, 8, 14, 32, 38:

|  |  |  |
| --- | --- | --- |
| 1 |  |  |

|  |  |  |
| --- | --- | --- |
| 1 | 3 |  |

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 3 | 6 | 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 | 8 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 | 8 | 14 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 | 8 | 14 | 32 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 | 8 |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 |  |  |  | 14 | 32 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 | 8 |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 |  |  |  | 14 | 32 | 38 |

(b) Remoção de 8:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 |  |  |  | 14 | 32 | 38 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 3 | 14 |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | 6 |  |  |  | 32 | 38 |  |

1. (3.0 pontos) Considere o grafo ponderado não direcionado abaixo.

a. (1.0 ponto) Liste os índices na ordem que eles serão visitados se for realizada uma busca em amplitude (breadth-first search) começando pelo nó A. Assuma que os vizinhos de um nó são visitados por sua ordem alfabética.



3

9

b. (1.0 ponto) Mostre como representar o grafo, incluindo os pesos, através de uma matriz de adjacências e de listas de adjacências, ambas modificadas para incluir os pesos.

c. (1.0 ponto) Usando o algoritmo de Kruskal, mostre uma árvore geradora mínima para o grafo. Indique em que ordem as arestas serão consideradas.

Resp.:

a) A – B – G – C – F – E – D

b)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| A |  | 5 |  |  |  |  | 2 |
| B | 5 |  | 5 |  |  | 1 |  |
| C |  | 5 |  |  | 4 | 3 |  |
| D |  |  |  |  | 1 |  |  |
| E |  |  | 4 | 1 |  | 9 |  |
| F |  | 1 | 3 |  | 9 |  | 3 |
| G | 2 |  |  |  |  | 3 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | 5 |  |  | G | 2 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B | A | 5 |  |  | C | 5 |  |  | F | 1 |  |

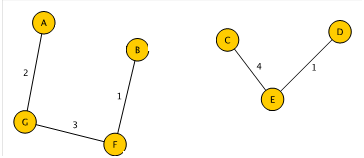
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | 5 |  |  | E | 4 |  |  | F | 3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| D | E | A |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | C | 4 |  |  | D | 1 |  |  | F | 9 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F | B | 1 |  |  | C | 3 |  |  | E | 9 |  |  | G | 3 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | 2 |  |  | F | 3 |  |

c) 

3

Protótipos e macros que podem ser úteis:

**stdio.h:**

int scanf (char\* formato, ...);

int printf (char\* formato, ...);

FILE\* fopen (char\* nome, char\* modo);

int fclose (FILE\* fp);

int fscanf (FILE\* fp, char\* formato, ...);

int fprintf (FILE\* fp, char\* formato, ...);

char\* fgets(char\* str, int size, FILE\* fp));

int sscanf(char\* str, char\* formato, ...);

**math.h:**

double sqrt (double x);

double pow (double x, double exp);

double cos (double radianos);

double sin (double radianos);

**string.h:**

int strlen (char\* s);

int strcmp (char\* s, char \*t);

char\* strcpy (char\* destino, char\* origem);

char\* strcat (char\* destino, char\* origem);

**stdlib.h:**

int abs ( int n );

void\* malloc (int nbytes);

void free (void\* p);

void qsort (void \*vet, int n, int tam, int (\*comp) (const void\*, const void\*));