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

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

1. A prova é individual e sem consulta.
2. A interpretação faz parte da questão.
3. O tempo de prova é 2:00 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.

**1) (2,0 pontos)** Dado a estrutura de dados abaixo, relativa a uma árvore binária, escreva uma função para verificar se a árvore é binária de busca.

typedef struct no t\_no;

struct no

{

int info;

t\_no \*esq, \*dir;

};

t\_no \*raiz;

A função deve receber como parâmetro um ponteiro para a raiz da árvore e deve retornar os valores: 1 (sim, é binária de busca) ou 0 (não é binária de busca).

Resp.:

int bin(t\_no \*r)

{

int esq, dir :

if (r == null)

return 1;

else {

if (r->dir != null && r->dir->info < r->info)

return 0;

else dir = bin(r->dir);

if (r->esq != null && r->esq->info > r->info)

return 0;

else esq = bin(r->esq);

return(min(esq,dir));

}

}

**2)** **(2.0 pontos)** A seguinte função realiza inserção de chaves em uma tabela de dispersão, utilizando desempate interno de colisões:

**#define MAX 8 /\* tamanho da tabela de hash \*/**

**#define DEC 2**

**#define VAZIO (-1)**

**int hash[MAX];**

**int insere (int x)**

**{**

**int k, pos;**

**pos = x % MAX;**

**for (k = 0; k < MAX; k++)**

**{**

**if (hash[pos] == VAZIO) /\* entrada vazia \*/**

**{**

**hash[pos] = x; /\* insere a chave \*/**

**return (pos);**

**}**

**if (hash[pos] == x) /\* chave duplicada \*/**

**return (pos);**

**pos = pos - DEC; /\* colisão \*/**

**if (pos < 0)**

**pos = pos + MAX;**

**}**

**return (-1); /\* não inseriu \*/**

**}**

Pede-se:

1. **(1.0 ponto)** Partindo de uma tabela vazia (todas as posições contêm –1) e utilizando a rotina dada, realize a inserção das seguintes chaves: 18, 28, 42, 78 e 32, desenhando a tabela a cada inserção.
2. **(1.0 ponto)** Escreva a expressão matemática da função de dispersão H(x,k) implicitamente utilizada na rotina acima. Por que H(x,k) não é uma boa função de dispersão? Justifique sua resposta.

***Resp.:***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | -1 |  | -1 |  | 42 |  | 42 |  |  |
| 1 | -1 |  | -1 |  | -1 |  | -1 |  |  |
| 2 | 18 |  | 18 |  | 18 |  | 18 |  |  |
| 3 | -1 |  | -1 |  | -1 |  | -1 |  |  |
| 4 | -1 |  | 28 |  | 28 |  | 28 |  |  |
| 5 | -1 |  | -1 |  | -1 |  | -1 |  |  |
| 6 | -1 |  | -1 |  | -1 |  | 78 |  |  |
| 7 | -1 |  | -1 |  | -1 |  | -1 |  |  |
|  |  |  |  |  |  |  |  |  | 32? |
| pos= | 2 |  | 4 |  | 2 |  | 6 |  | 4 |
| pos-dec= |  |  |  |  | 0 |  |  |  | 2 |

Não é possivel inserir 32. A variável pos assume só valores pares, posições estas que já estão todas ocupadas.

(b) A função de hash h(x) = x % MAX não é adequada pois MAX deveria ser um número primo não próximo a uma potência de 2, ou um número sem divisores primos menores que 20.

**3)** **(3.0 pontos)** Considere a árvore B de ordem 4 (ou seja, todo nó tem nó máximo 4 filhos e 3 chaves) abaixo:

100

50 75 120 200

10 40 60 70 80 90 110 115 130 135 140 220 230 240

Realize as seguintes operações, utilizando sempre a árvore resultante da operação anterior. Redesenhe a árvore a cada passo, indicando os nós que sofrem modificações.

a) **(1.0 ponto)** Inserção de 150.

b) **(1.0 ponto)** Remoção de 120 (após a inserção de 150).

c) **(1.0 ponto)** Remoção de 100 (após a inserção de 150 e a remoção de 120).

Resp.:

(a)

100

50 75 120 135 200

10 40 60 70 80 90 110 115 130 140 150 220 230 240

(b)

100

50 75 135 200

10 40 60 70 80 90 110 115 130 140 150 220 230 240

(c)

110

50 75 135 200

10 40 60 70 80 90 115 130 140 150 220 230 240

**4) (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 profundidade (depth-first search) começando pelo nó A. Assuma que os vizinhos de um nó são visitados por sua ordem alfabética.



1

2

3

9

b. (2.0 pontos) Usando o algoritmo de Dijkstra, compute qual é o caminho mínimo de A a C. Mostre todos os passos do algoritmo.

Resp.:

(a)

Nó visitado / Pilha

A BG

B CFG

C EFFG

E DFFG

D FFG

F FG

G

G

Ordem: A - B - C - E - D - F – G

(b)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Passos  / Nós | 0 | Corrente  / não vis | 1 | Corrente  / não vis | 2 | Corrente  / não vis | 3 | Corrente  / não vis | 4 | Corrente  / não vis | 5 | Corrente  / não vis | 6 | Corrente  / não vis | 7 |
| A | 0 | A | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |
| B | ∞ | x | 5 | x | 5 | x | 5 | B | 5 |  | 5 |  | 5 |  | 5 |
| C | ∞ | x | ∞ | x | ∞ | x | 8 | x | 8 | C | 8 |  | 8 |  | 8 |
| D | ∞ | x | ∞ | x | ∞ | x | ∞ | x | ∞ | x | ∞ | x | ∞ | D | 8 |
| E | ∞ | x | ∞ | x | ∞ | x | 7 | x | 7 | x | 7 | E | 7 |  | 7 |
| F | ∞ | x | ∞ | x | 5 | F | 5 |  | 5 |  | 5 |  | 5 |  | 5 |
| G | ∞ | x | 2 | G | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |

Ou

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Passos) | 1 | Corrente  / não vis | 1 | Corrente  / não vis | 2 | Corrente  / não vis | 3 | Corrente  / não vis | 4 | Corrente  / não vis | 5 | Corrente  / não vis | 6 | Corrente  / não vis | 7 |
| A | 0 | A | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |
| B | ∞ | x | 5 | x | 5 | B | 5 |  | 5 |  | 5 |  | 5 |  | 5 |
| C | ∞ | x | ∞ | x | ∞ | x | 10 | x | 8 | x | 8 |  | 8 |  | 8 |
| D | ∞ | x | ∞ | x | ∞ | x | ∞ | x | ∞ | x | ∞ | x | 8 | D | 8 |
| E | ∞ | x | ∞ | x | ∞ | x | ∞ | x | 7 | E | 7 | E | 7 |  | 7 |
| F | ∞ | x | ∞ | x | 5 | x | 5 | F | 5 |  | 5 |  | 5 |  | 5 |
| G | ∞ | x | 2 | G | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |

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\*));