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

|  |  |  |
| --- | --- | --- |
| 1a) | 3.0 |  |
| 2a) | 2.0 |  |
| 3a) | 2.0 |  |
| 4a) | 3.0 |  |
|  | 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) Considere uma rotina 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

(a) (1.0 ponto) Assuma que a tabela está originalmente vazia e que N=1007. Mostre a computação da função de dispersão e indique em que posição da tabela cada chave será inserida, nesta ordem: 17, 1024, 18, 1031.

(b) (2.0 pontos) Escreve uma rotina em C para inserção de chaves, definidas como números inteiros, em uma tabela de dispersão com endereçamento aberto com a função de dispersão definida acima. 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:

(a)

(b)

2) (2.0 pontos) Considere a seguinte sequencia de inteiros: 90, 60, 30, 15, 45.

* 1. (0.5 ponto) Mostre, passo a passo, como o vetor armazenando um *heap* mínimo é construído pela inserção sucessiva destes 5 elementos, na ordem dada. Comente brevemente cada passo do algoritmo de inserção.
  2. (1.5 ponto) Mostre, passo a passo, como fica o vetor após a remoção do segundo menor elemento do heap construído no Item (a). Comente brevemente cada passo do algoritmo de remoção. Cada passo deve corresponder a uma operação de inserção ou remoção do *heap* e deve usar apenas o vetor armazenando o *heap*, uma variável auxiliar *T* para trocas e uma variável *R* para devolver o resultado.

Resposta:

a) Use o modelo abaixo na sua resposta:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **Explicação** |
| 90 |  |  |  |  | Acrescente 90 depois do final do heap (que está vazio). | |
| 90 | 60 |  |  |  | Acrescente 60 depois do final do heap. | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |
|  |  |  |  |  |  | |

b) Use o modelo abaixo na sua resposta:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  | T | R | | **Explicação** |
| 15 | 30 | 60 | 90 | 45 |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  |  | |  |  |

(3) (2.0 pontos) Os algoritmos abaixo são usados para resolver problemas de tamanho *n*. Determine a complexidade, no pior caso, de cada algoritmo. Explique sua resposta.

a) (1.0 ponto)

for ( i=1; i < n; i \*= 2 ) {

for ( j = n; j > 0; j += 2 ) {

for ( k = j; k < n; k += 2 ) {

sum += (-j \* k) << i/2;

}

}

}

b) (1.0 ponto)

for ( i=1; i < n; i \*= 2 ) {

for ( j = n; j > 0; j -= 2 ) {

for ( k = j; k < n; k += 2 ) {

sum += (-j \* k) << i/2;

}

}

}

Resposta:

(4) (3.0 pontos) Considere um TAD de Conjuntos, com a interface apresentada em sala:

typedef struct \_bitvector BitVector;

e a implementação:

struct \_bitvector {

int max;

int \*vector;

};

Implemente em C uma função que computa a união de dois conjuntos. A interface da função é a seguinte:

BitVector\* bvUnion(BitVector\* a, BitVector\* b);

Nota: assuma que os conjuntos a e b foram originalmente inicializados pela função bvInit.

Resposta: