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

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

* A prova é individual e sem consulta.
* A interpretação faz parte da questão.
* O tempo de prova é 1:45 h.
* As respostas devem seguir as questões.
* Caso precise de rascunho use o verso da folha.
* Caso parte da resposta esteja no verso, indique claramente este fato.

1. (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)

int funcao\_C(int n) {

int i, j, sum = 0;

for (i=0; i<n; i++) {

sum+=i;

}

for (j=0; j<n; j++){

sum+=2\*j;

}

return sum;

}

**Resposta:**

1. (2.0 pontos) Uma tabela de dispersão de tamanho 11 é implementada com encadeamento externo através da seguinte função de dispersão:
   1. (1.0 ponto) Desenhe a estrutura de dados após a inserção das chaves (nesta ordem):

7,10,15,14,17,3,21,25

* 1. (1.0 ponto) Explique, com base no exemplo anterior, como remover sucessivamente cada uma das seguintes chaves:

5, 7, 10, 3

**Resposta:**

1. (2.0 pontos) Considere a seguinte sequencia de inteiros: 90, 60, 30, 15.
   1. (0.5 ponto) Mostre, passo a passo, como o vetor armazenando um *heap* mínimo é construído pela inserção sucessiva destes 4 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 terceiro 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 usar apenas o vetor armazenando o *heap*, uma variável auxiliar *T* para trocas e uma variável *R* para devolver o resultado.

**Resposta:**

1. (2.0 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 diferença de dois conjuntos. A função recebe como entrada dois conjuntos, *a* e *b*, e retorna o conjunto *c* formado pelos elementos de *a* que não estão em *b*. A interface da função é a seguinte:

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

**Resposta:**

1. (2.0 pontos) Considere uma partição dinâmica do conjunto {0,1,2,...8}, representada em uma floresta otimizada com UNION por altura (em caso de empate na altura, a raiz com o maior rótulo deverá ser colocada como filha da outra raiz) e FIND com compressão de trajetória.

Assuma que a floresta inicialmente resulta da aplicação das operações *MAKE\_SET(i)*, para *i=0,...,8.*

Mostre a floresta após aplicar sucessivamente cada uma de operações das sequencias abaixo:

1. (1.0 ponto) UNION(0,2), UNION(1,5), UNION(3,8), UNION(4,6), UNION(0,3)
2. (1.0 ponto) UNION(1,4), UNION(0,1), FIND(6)

**Resposta:**