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

|  |  |  |
| --- | --- | --- |
| 1a) | 2,0 |  |
| 2a) | 3,0 |  |
| 3a) | 3,0 |  |
| 4a) | 2,0 |  |
|  |  |  |

* A prova é individual e sem consulta.
  + **Qualquer tentativa de “cola” resultará na anulação da prova do aluno ou dos alunos envolvidos**.
  + Os aparelhos celulares deverão permanecer desligados durante toda a prova. **Aparelhos celulares ligados serão tratados como tentativa de “cola”**.
* A interpretação faz parte da questão.
  + **Não há perguntas durante a prova.**
  + Em caso de dúvida escreva a dúvida e a sua interpretação na resposta.
* O tempo de prova é 1:45 h.
* **Após o início da prova, não será possível sair e depois voltar à sala.**
* As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
* A prova pode ser feita a lápis.

**Questão 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)

int funcao\_C(int n) {

int i, j, k, sum = 0;

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;

}

}

}

return sum;

}

b) (1,0 ponto)

int funcao\_C(int n) {

int i, j, sum = 0;

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

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

sum+=j;

}

}

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

sum+=2\*j;

}

return sum;

}

**Resposta**

**Questão 2** (3,0 pontos) Uma tabela de dispersão de tamanho 11 é implementada com encadeamento externo (resolução de colisões por listas encadeadas) através da seguinte função de dispersão:

/\* resto da divisão de x por 11 \*/

* 1. (1,5 ponto) Desenhe a estrutura de dados após a inserção das chaves (nesta ordem):

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

* 1. (0,5 ponto) Apresente uma sequencia de 11 chaves para as quais a função de hash produz um número máximo de colisões. Explique a sua resposta.
  2. (1,0 ponto) Considere uma tabela de hash de tamanho fixo *n*. Suponha que *m* chaves tenham sido inseridas na tabela, onde *m* < *n*. Qual o custo, no pior caso, da pesquisa por uma chave? Explique a sua resposta.

**Resposta**

**Questão 3** (3,0 pontos) Considere conjuntos de elementos com suas prioridades. Assuma que as prioridades são números inteiros e que os conjuntos são representados apenas pelas prioridades dos seus elementos, como nos exemplos em sala.

* 1. (1,5 ponto) Considere o conjunto *C*={90, 60, 30, 15, 45}. Mostre, passo a passo, como um vetor armazenando um *heap* mínimo para o conjunto *C* é construído, da melhor maneira possível. Comente brevemente cada passo do algoritmo.
  2. (1,5 ponto) Mostre, passo a passo, como inserir 5 no vetor abaixo, que armazena um *heap* mínimo. Comente brevemente cada passo da inserção.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posição | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  | … |
| prioridade | 7 | 9 | 18 | 15 | 17 | 21 | - |  | … |

**Resposta**

**Questão 4** (2,0 pontos)

Considere a representação de conjuntos utilizando vetores de bits, definidos como:

struct \_bitvector {

int max; /\* número máximo de elementos do conjunto \*/

int \*vector;

};

typedef struct \_bitvector BitVector;

a) (1,5 ponto) Implemente em C uma rotina que recebe como entrada dois conjuntos *A* e *B* e devolve o conjunto *D* = *A* – *B*. A rotina não deverá chamar as operações definidas em sala para vetores de bits.

Nota: *A* – *B*, a *diferença* de *A* e *B*, é o conjunto de todos os elementos de *A* que não estão em *B*.

b) (0,5 ponto) Qual a complexidade, no pior caso, da rotina implementada no item (a), medida em termos do número de elementos dos conjuntos. As operações bit-a-bit influenciam a complexidade, no pior caso? Explique cuidadosamente a sua resposta.

**Resposta**