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

|  |  |  |
| --- | --- | --- |
| 1a) | 1,0 |  |
| 2a) | 1,5 |  |
| 3a) | 2,5 |  |
| 4a) | 2,5 |  |
| 5a) | 2,5 |  |
|  | 10,0 |  |

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

**Questão 1** (1,0 ponto) Os algoritmos abaixo são usados para resolver problemas de tamanho *n*. Determine a complexidade, no pior caso, de cada algoritmo. Explique sua resposta.

a) (0,5 ponto)

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

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

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

}

}

}

for ( i=1; i < n; i \*= 5 ) {

for ( j = n; j > 0; j /= 3 ) {

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

}

}

}

b) (0,5 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:

**Questão 2** [1,5 pontos]:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 30 |  |  | sp | ! | " | # | $ | % | & | ' |
|  | 40 | ( | ) | \* | + | , | - | . | / | 0 | 1 |
|  | 50 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; |
|  | 60 | < | = | > | ? | @ | A | B | C | D | E |
|  | 70 | F | G | H | I | J | K | L | M | N | O |
|  | 80 | P | Q | R | S | T | U | V | W | X | Y |
|  | 90 | Z | [ | \ | ] | ^ | \_ | ` | a | b | c |
|  | 100 | d | e | f | g | h | i | j | k | l | m |
|  | 110 | n | o | p | q | r | s | t | u | v | w |
|  | 120 | x | y | z | { | | | } | ~ |  |  |  |

Figura 1: Códigos ASCII de alguns caracteres que podem ser impressos (sp representa espaço).

A Figura 1 apresenta os códigos ASCII de alguns caracteres. Por exemplo, os códigos das letras “d” e “e” são respectivamente “100” e “101”. Considere as seguintes funções de hash, onde s é uma cadeia de caracteres ASCII formada apenas por letras minúsculas do alfabeto, representando a chave, e M é o tamanho da tabela de hash:

int hash1(string s, int M)

{ int h = 0;

for (i=0; s[i]!='\0'; i++)

h = s[i] + (h << 6) + (h << 16) - h;

h = abs(h) % M;

return h;

}

int hash2(string s, int M)

{ int i, h=0;

for (i=0; s[i]!='\0'; i++)

h+=s[i];

h = abs(h) % M;

return h;

}

a) [0,5 ponto] Estas funções estão corretas? Explique a sua resposta.

b) [1,0 ponto] Estas funções são boas funções de hash? Explique a sua resposta.

Resposta:

**Questão 3** (2,5 pontos) Considere o problema de representar 2 subconjuntos de um universo *A* de alunos, definidos como:

*E* = { *x* ∈ *A* / *x* estuda Engenharia de Computação }

*I* = { *x* ∈ *A* / *x* cursou INF1010 }

Suponha que os subconjuntos são representados utilizando vetores de bits, definidos como:

struct \_bitvector {

int max;

int \*vector;

};

typedef struct \_bitvector BitVector;

a) (1,0 ponto) Suponha que:

*A* = {A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18}

*E* = {A1, A3, A4, A10, A18}

*I* = {A2, A3, A4}

Considere que um número inteiro ocupe apenas 16 bits e que os bits são ordenados da direita para a esquerda. Assuma que A1 corresponde ao bit mais à direita, A2 corresponde ao segundo bit mais à direita, e assim por diante.

Mostre os vetores de bits que representam *E* e *I*.

b) (1,5 pontos) Implemente em C uma rotina que recebe como entrada *E* e *I* e devolve o subconjunto *N* de *A* definido como:

*N* = { *x* ∈ *A* / *x* não estuda Engenharia de Computação e *x* cursou INF1010 }

A rotina não deverá chamar as operações definidas em sala para a estrutura escolhida.

Resposta:

**Questão 4** (2,5 pontos) Considere o problema de agrupar os alunos de uma universidade pelo curso em que estão matriculados, assumindo que um aluno só pode estar matriculado em um curso, mas que ele pode mudar de curso ao longo do tempo. Assuma ainda que se deseja realizar duas operações:

* Determinar eficientemente se dois alunos não estão no mesmo curso;
* Implementar eficientemente a união de 2 cursos em um só.

a) (1,0 ponto) Qual estrutura de dados, dentre as apresentadas em sala, seria mais eficiente para resolver o problema acima? Explique sua resposta, indicando como as operações acima serão implementadas.

b) (0,5 pontos) Defina um typedef que corresponda à estrutura escolhida no item (a).

c) (1,0 ponto) Implemente em C uma rotina para unir 2 cursos em um só. A rotina deverá usar o typedef definido no item (b), mas não deverá chamar as operações definidas em sala para a estrutura escolhida.

Resposta:

**Questão 5** (2,5 pontos) Considere o problema de organizar um conjunto de alunos de tal forma que os alunos com maior CR tenham prioridade sobre os alunos que tem menor CR. Caso dois alunos tenham o mesmo CR, a ordem é indistinta. Assuma ainda que se deseja realizar as seguintes operações:

* Organizar inicialmente o conjunto de alunos, segundo a prioridade por CR;
* Remover o aluno com maior CR;
* Incluir um novo aluno, segundo a prioridade por CR.

a) (0,5 ponto) Porque um *heap* seria a estrutura mais eficiente para resolver o problema acima? Explique sua resposta.

b) (0,5 ponto) Considere inicialmente um conjunto de alunos com os seguintes CRs: 6.2, 7.2, 7.1, 6.0, 7.1. Mostre como construir o *heap*, utilizando um vetor, da forma mais eficiente possível, para organizar este conjunto de alunos com prioridades dadas pelo CR.

c) (1,5 ponto) Escreva em C uma função que recebe como argumento um *heap*, organizado como nos itens anteriores, e remove do *heap* o aluno com o segundo maior CR, usando o menor espaço e o menor tempo possíveis. A rotina deve retornar 1 se a alteração ocorreu corretamente e 0 em caso contrário. Assuma que o *heap* seja definido como abaixo:

struct \_heap {

int max; /\* tamanho máximo do heap \*/

int pos; /\* próxima posicao disponível no vetor \*/

float\* prioridade; /\* vetor das prioridades \*/

};

typedef struct \_heap Heap;

Assuma ainda que a assinatura da rotina seja:

int heap\_altera(Heap\* heap)

A rotina não deve chamar as operações de inserção ou remoção de um heap apresentadas em sala, mas poderá usar uma das operações auxiliares apresentadas em sala.

Resposta: