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

|  |  |  |
| --- | --- | --- |
| 1a) | 2,5 |  |
| 2a) | 2,5 |  |
| 3a) | 2,5 |  |
| 4a) | 2,5 |  |
|  | 10.0 |  |

**LEIA COM CUIDADO**

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 e guardados fora do alcance durante toda a prova. **Aparelhos celulares ligados ou de alguma forma visíveis 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 (2,5 pontos)**

a) (0,5 ponto) Suponha que *A* seja uma árvore ABB (Árvore de Busca Binária). Qual a complexidade no pior caso, expressa em função de *n*, do problema de determinar a altura de *A*? Explique a sua resposta.

b) (1,0 ponto) Suponha que *A* seja uma árvore AVL. Qual a complexidade no pior caso, expressa em função de *n*, do problema de determinar a altura de *A*? Explique a sua resposta.

c) (1,0 ponto) Suponha que *A* seja uma árvore-B de grau *m*. Qual a altura máxima *h* de *A*, expressa em função de *n* e *m*? Explique a sua resposta.

**Resposta**

(a) Uma ABB pode degenerar em apenas um ramo, com altura *h* = *n*-1. Assim, a complexidade, no pior caso, do problema de determinar a altura de uma árvore ABB será *O*(*n*).

(b) A altura de uma árvore AVL com *n* chaves (ou nós) é dada por *h* ≈ 1.44*log*(*n*) (ver notas de aula). Assim, a complexidade, no pior caso, do problema de determinar a altura de uma árvore AVL será *O*(*log*(*n*)).

(c) O número mínimo *k* de chaves de uma árvore-B de ordem *m* é *k* = 2 × ⎡*m*/2⎤*h* − 1. Assim, a altura máxima *h* de uma árvore-B de ordem *m* com *n* chaves será:

*n* = 2 × ⎡*m*/2⎤*h* − 1

*n* + 1 = 2 × ⎡*m*/2⎤*h*

**Questão 2 (2,5 pontos)**

Considere a árvore B de ordem 3 abaixo.

N1 100 108

N2 53 77 N3 103 N4 123 200

10 40 60 70 80 90 101 104 105 113 130 138 230 236

N5 N6 N7 N8 N9 N10 N11 N12

Considere um buffer pool que armazena 4 nós e é gerenciado de tal forma que os nós há mais tempo sem serem usados são retirados, quando necessário. **Em todos os casos dos itens (a) e (b),** **suponha que o buffer pool mantém o estado da operação anterior quando uma nova operação é iniciada**.

* 1. (1,5 pontos) Assuma que o buffer pool está no estado mostrado na linha 0 da tabela abaixo. Mostre os estados pelos quais passa o buffer pool quando são sucessivamente pesquisadas as chaves 40, 104 e 10, nesta ordem. Explique sua resposta.
  2. (1,0 ponto) Mostre os estados por que passa o buffer pool quando é inserida a chave 106, após as operações do item (a), reescrevendo em memória secundárias os nós novos ou modificados. Explique sua resposta.

Em todos os itens, refira-se aos nós usando os rótulos indicados na figura acima. Por exemplo, o nó N1 é a raiz. Use a tabela abaixo para mostrar o estado do buffer pool e incluir uma explicação para cada passo.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Operação | Buffer Pool  (nó mais novo à direita) | Explicação |
| 0 |  | N1 | Estado inicial do buffer pool |
| 1 | Pesquisa da chave 40 |  |  |
| 2 | Pesquisa da chave 104 |  |  |
| 3 | Pesquisa da chave 10 |  |  |
| 4 | Inserção da chave 106 |  |  |

***Resposta***

|  |  |  |  |
| --- | --- | --- | --- |
|  | Operação | Buffer Pool  (nó mais novo à direita) | Explicação |
| 0 |  | N1 | Estado inicial do buffer pool |
| 1 | Pesquisa da chave 40 | N1  N2 N1  N5 N2 N1 | Acesso a N1 no buffer pool  Acesso a N2  Acesso a N5 |
| 2 | Pesquisa da chave 104 | N1 N5 N2  N3 N1 N5 N2  N9 N3 N1 N5 | Acesso a N1 no buffer pool  Acesso a N3  Liberação de N2 e acesso a N9 |
| 3 | Pesquisa da chave 10 | N1 N9 N3 N5  N2 N1 N9 N3  N5 N2 N1 N9 | Acesso a N1 no buffer pool  Liberação de N5 e acesso a N2  Liberação de N3 e acesso a N5 |
| 4 | Inserção da chave 106 | N1 N5 N2 N9  N3 N1 N5 N2  N9 N3 N1 N5  N13 N9 N3 N1  N3 N13 N9 N1  N3 N13 N9 N1  N13 N3 N9 N1 | Estado do buffer pool  Liberação de N9 e acesso a N3  Liberação de N2 e acesso a N9  Liberação de N5, split de N9, criação de N13, inserção de 106 em N13  Inserção de 105 em N3  Gravação de N3  Gravação de N13 |

N1 100 108

N2 53 77 N3 103 N4 123 200

10 40 60 70 80 90 101 104 105 113 130 138 230 236

N5 N6 N7 N8 N9 N10 N11 N12

**106**

N1 100 108

N2 53 77 N3 103 N4 123 200

10 40 60 70 80 90 101 104 **106** 113 130 138 230 236

N5 N6 N7 N8 N9 **N13** N10 N11 N12

**105**

N1 100 108

N2 53 77 N3 103 **105** N4 123 200

10 40 60 70 80 90 101 104 106 113 130 138 230 236

N5 N6 N7 N8 N9 N13 N10 N11 N12

**Questão 3 (2,5 pontos)** Considere a árvore B de ordem 3 abaixo:

100 105

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 138 230 236

Realize as seguintes operações, indicando os nós que sofrem modificações (divisão, redistribuição ou concatenação) após cada operação:

1. (1.0 ponto) Inserção de 139 na árvore original.
2. (1.0 ponto) Remoção de 103 da árvore original (use a sucessora).
3. (0,5 ponto) A árvore acima pode ser considerada uma árvore B de ordem *n*, onde *n* > 3? Explique a sua resposta.

***Resposta***

a) Inserção de 139. Pesquise o nó onde a chave 139 deve ser inserida.

100 105

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 138 230 236

***139***

Divida o nó e insira a chave do meio (138) no pai.

100 105

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 139 230 236

***138***

Divida o nó e insira a chave do meio (138) no pai.

100 105

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 139 230 236

***138***

Divida o nó e crie uma nova raiz com a chave do meio (138).

100 138

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 139 230 236

***105***

***105***

100 138

53 77 103 123 200

10 40 60 70 80 90 101 104 113 130 139 230 236

b) Remoção de 103 da árvore original. A chave 103 ocorre em um nó interior e portanto deve ser trocada com a sua sucessora, 104 (a questão pedia para usar a sucessora).

100 105

53 77 104 123 200

10 40 60 70 80 90 101 113 130 138 230 236

Combine com o pai (caso 5 da remoção em árvores 2-3, aplicado aos nós em negrito acima):

100 105

53 77 123 200

10 40 60 70 80 90 101 104 113 130 138 230 236

Redistribua (caso 8 da remoção em árvores 2-3, aplicado aos nós em negrito acima):

77 105

53 100 123 200

10 40 60 70 80 90 101 104 113 130 138 230 236

d) A árvore pode ser considerada uma árvore B de ordem *n*=4. De fato, em uma árvore B de ordem 4, os nós internos poderão ter um mínimo de *f*=⎡4/2⎤=2 filhos e um mínimo de   
*k*=(*f*-1)=1 chaves e as folhas poderão ter um mínimo de *k*=1 chaves.

**Questão 4 (2,5 pontos)**

a) (0,5 ponto) Considere a tabela de dispersão estendida, *T*, parcialmente representada abaixo. Quantos apontadores em *T* apontam para o bloco *B1* ? Caso haja mais de um apontador, além do ilustrado na figura, em quais posições da tabela *T* eles estarão armazenados? Explique sua resposta.

*T*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *i =3* |  |  |  |
| 0 |  |  |  | *B1* |
| 1 |  |  |  | *j1 =2* | |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |

b) (1,0 ponto) Considere a tabela de dispersão estendida representada na figura abaixo. Suponha que cada bloco possa conter até 3 chaves. Mostre como a estrutura ficará após a remoção de *K21*, assumindo que a estrutura deve economizar espaço sempre que possível. Explique sua resposta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *i =2* |  |  |  |
| 0 |  |  | *K11* | *B1* |
| 1 |  |  | *K12* | *j1 =2* | |
| 2 |  |  |  |  | |
| 3 |  |  |  |  |
|  |  |  | *K21* | *B2* |
|  |  |  | *K22* | *j2 =2* | |
|  |  |  |  |  | |
|  |  |  |  |  |
|  |  |  | *K31* | *B3* |
|  |  |  | *K32* | *j3 =2* | | |
|  |  |  |  |  | | |
|  |  |  |  |  |
|  |  |  | *K41* | *B4* |
|  |  |  | *K42* | *j4 =2* | |
|  |  |  |  |  | |

c)(1,0 ponto) Assuma que *T* possui *n* chaves armazenadas. Qual o custo, no pior caso, para pesquisar uma chave *K* em *T* ? Explique sua resposta.

***Resposta***

a) Com *i = 3* e *j1 = 2*, há *p = 2(3-2)= 2* apontadores para *B1* . O segundo apontador estará em *T[6]* pois as chaves mapeadas em *T[2]* e *T[6]* possuem os 2 últimos bits iguais.

b) Após a remoção de *K21*, a chave *K22* deve ser movida para *B4* e o bloco *B2* descartado. De fato, os blocos *B2* e *B4* são “companheiros”, no sentido de que as chaves mapeadas em *T[1]* e *T[3]* possuem o último bit igual.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *i =2* |  |  |  |
| 0 |  |  | *K11* | *B1* |
| 1 |  |  | *K12* | *j1 =2* | |
| 2 |  |  |  |  | |
| 3 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  | |
|  |  |  |  |  | |
|  |  |  |  |  |
|  |  |  | *K31* | *B3* |
|  |  |  | *K32* | *j2 =2* | | |
|  |  |  |  |  | | |
|  |  |  |  |  |
|  |  |  | *K41* | *B4* |
|  |  |  | *K42* | *j4 =1* | |
|  |  |  | *K22* |  | |

c) No pior caso, todas as *n* chaves possuem os mesmo bits finais, depois de aplicada a função de hash. Ou seja, *T* degenera em uma lista encadeada de buckets, cada um com 2 chaves. O custo de pesquisar uma chave será então proporcional a *n/2* (pois cada bucket contém 2 chaves no máximo). Ou seja, o custo, no pior caso, para pesquisar uma chave *K* em *T* será *O*(*n*).