Aluno(a):_____Matrícula:_____

1ª)	3.0	
2ª)	2.0	
3ª)	2.0	
4ª)	3.0	
	10.0	

a) A prova é individual e sem consulta. Qualquer tentativa de "cola" resultará na anulação da prova do aluno ou de ambos os alunos envolvidos.
b) 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.
c) O tempo de prova é 1:45 h.
d) As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
e) A prova pode ser feita a lápis.

1) (3.0 pontos) A *densidade d* de uma árvore binária *B* com altura *h* e *n* nós é definida como

$$d = \frac{n}{N}$$

onde *N* é o número de nós de uma árvore binária cheia de altura *h*.

Implemente em C uma função que calcule a densidade de uma árvore binária. A função deve receber como entrada um apontador para a raiz da árvore e retornar a densidade da árvore:

```
float abb_densidade (Abb* r);
```

Adote a seguinte estrutura para os nós:

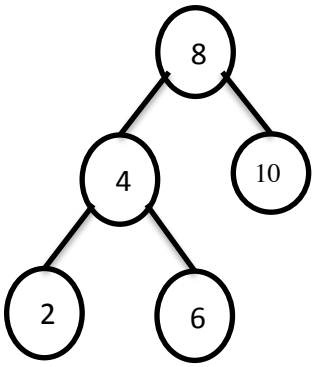
```
typedef struct _abb Abb;
struct _abb {
   Abb* esq;
   Abb* dir;
};
```

Resposta:

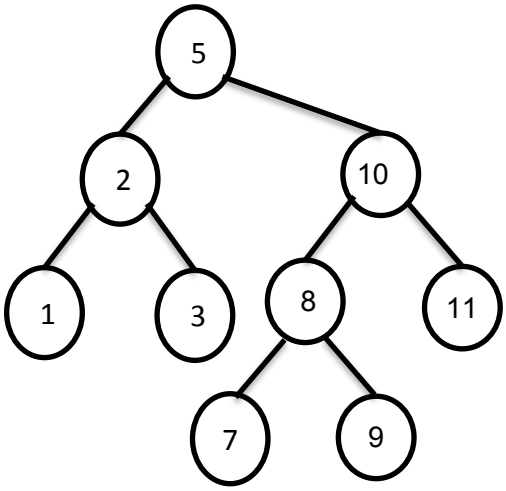
```
float abb_densidade (Abb* r)
{  float d; int n, h, N;
   n = abb_nos (Abb* r);
   h = abb_altura (Abb* r);
   N = 2**(h+1)-1;      /* número de nós de uma árvore binária
                           cheia de altura h */
   d = n/N;
   return d;
}

int abb_nos (Abb* r)    /* conta o número de nós */
{  int ne = 0, nd = 0;
   if (r == NULL) return 0;
   ne = abb_nos (r->esq);
   nd = abb_nos (r->dir);
   return 1 + ne + nd;
}

int abb_altura (Abb* r) /* calcula a altura */
{  int he = 0, hd = 0;
   if (r == NULL) return 0;
   he = abb_altura (r->esq);
   hd = abb_altura (r->dir);
   return 1 + (he > hd ? he : hd);
}
```

2) (2.0 pontos) Em cada um dos casos abaixo indique qual ou quais rotações devem ser feitas para rebalancear a árvore AVL após a operação especificada e indique os fatores de balanceamento antes e depois das rotações. Explique qual ou quais rotações foram escolhidas em função dos fatores de balanceamento.

a) (1.0 ponto) **Inserção da chave 7**.



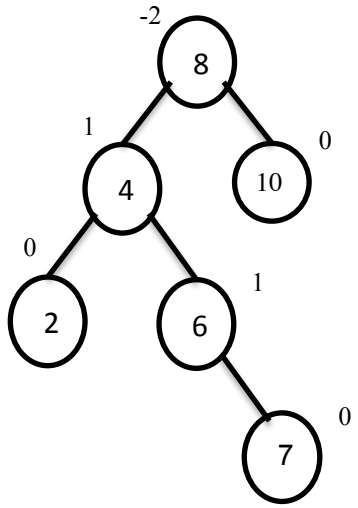
b) (0.5 ponto) **Remoção da chave 10**.



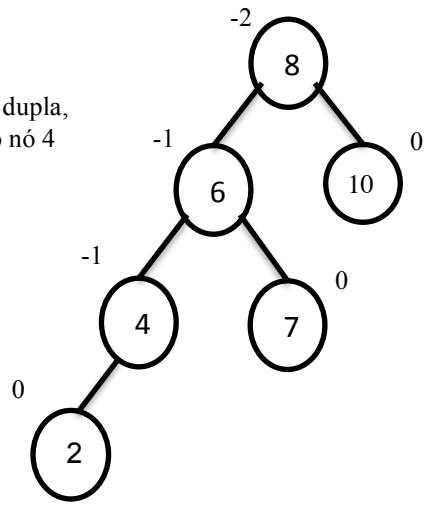
b) (0.5 ponto) **Remoção de uma segunda forma da chave 10 da árvore do item (b)**.

Resposta:

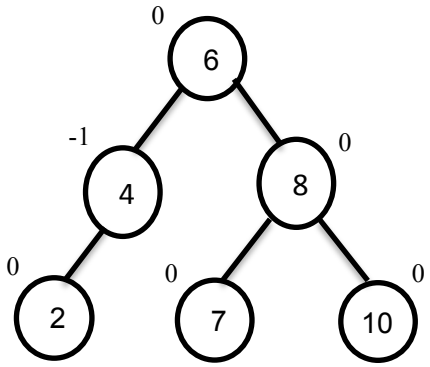
(i) A raiz possui fator de balanceamento -2 e o nó rotulado com 4 tem fator de balanceamento +1. Logo, deve-se fazer uma rotação à esquerda na sub-árvore à esquerda:



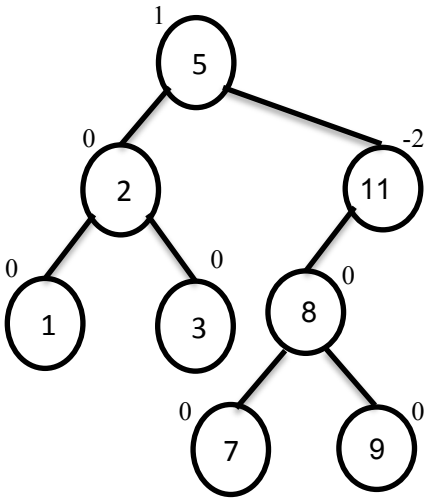
-2 e 1 indicam rotação dupla, primeiro à esquerda do nó 4



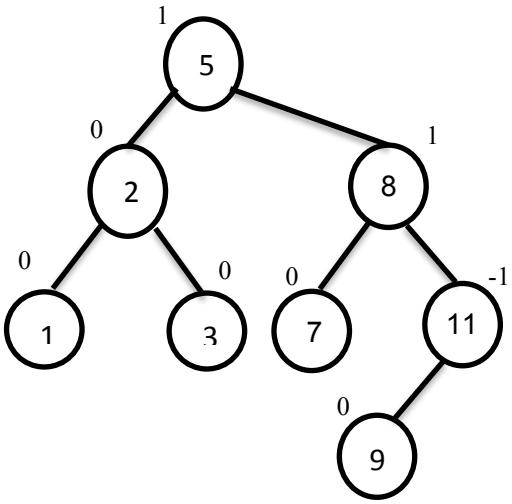
seguida de uma rotação à direita no nó 8:



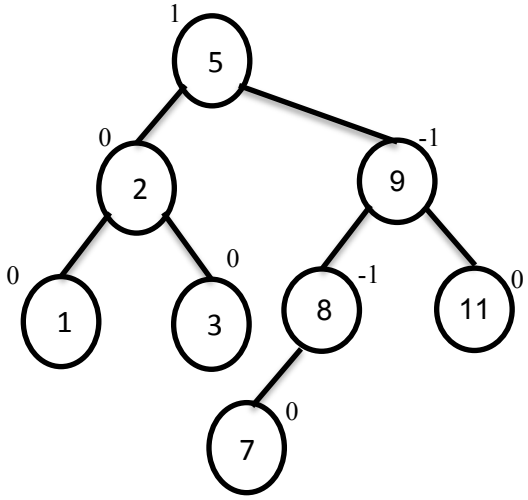
b) Remoção da chave 10. Substitui-se primeiro 10 pela sua sucessora, 11, e remove-se 10.



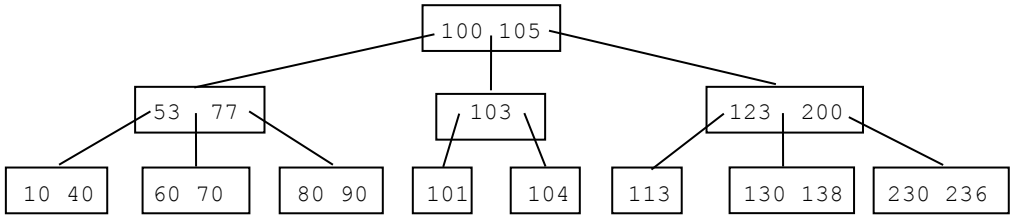
-2 indica rotação à direita em 11



c) Remoção da chave 10. Substitui-se primeiro 10 pela sua antecessora, 9, e remove-se 10.



3) (2.0 pontos) Considere a árvore B de ordem 3 abaixo:



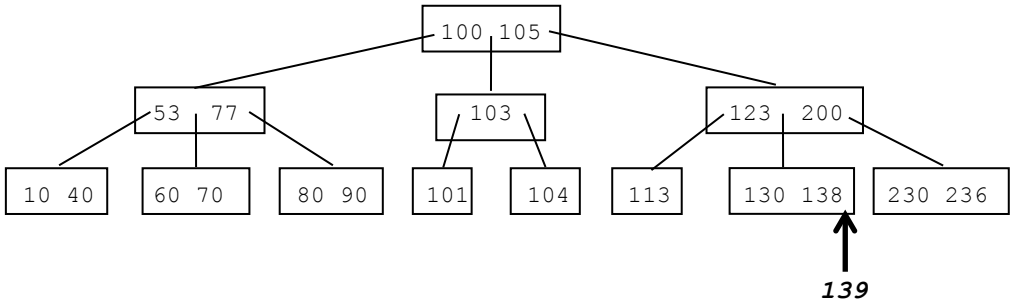
Realize as seguintes operações, indicando os nós que sofrem modificações (divisão, redistribuição ou concatenação) após cada operação:

a) (1.0 ponto) Inserção de 139 na árvore original.
b) (1.0 ponto) Remoção de 103 da árvore original (use a sucessora).

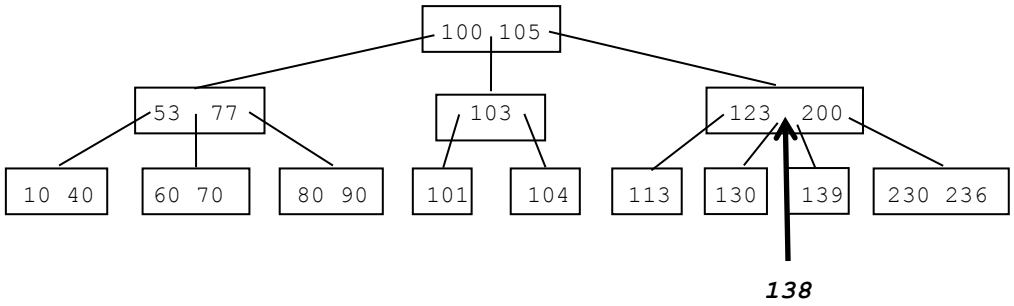
Resposta:

a) Inserção de 139.

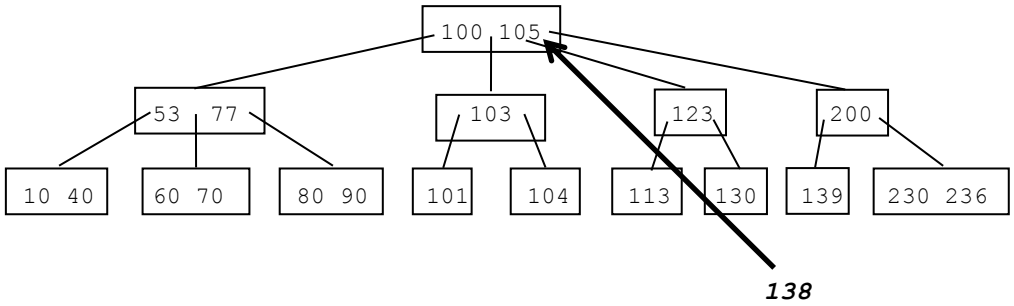
Pesquise o nó onde a chave 139 deve ser inserida.



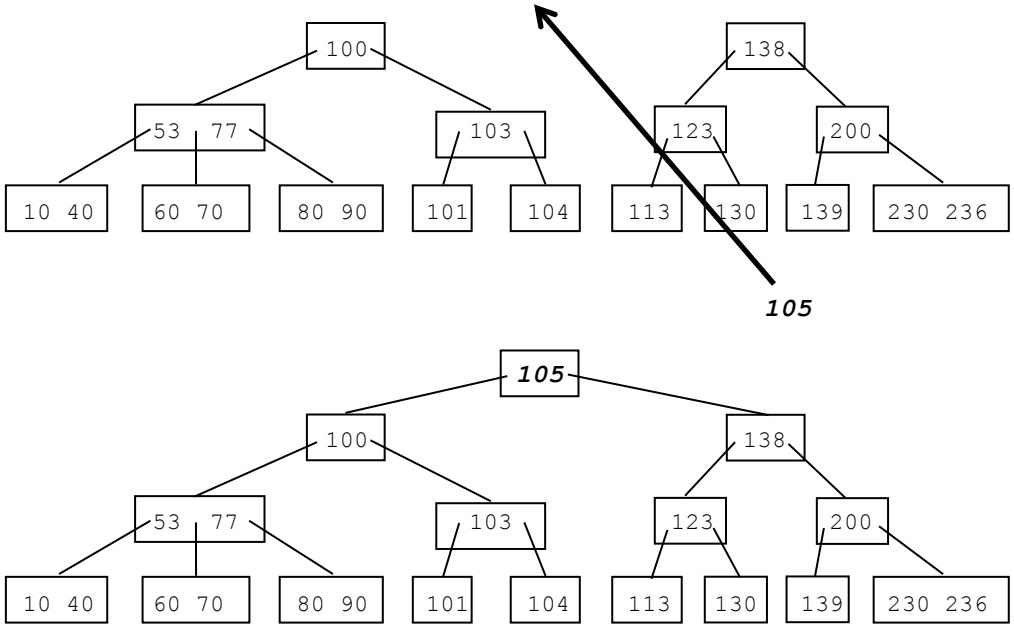
Divida o nó e insira a chave do meio (138) no pai.



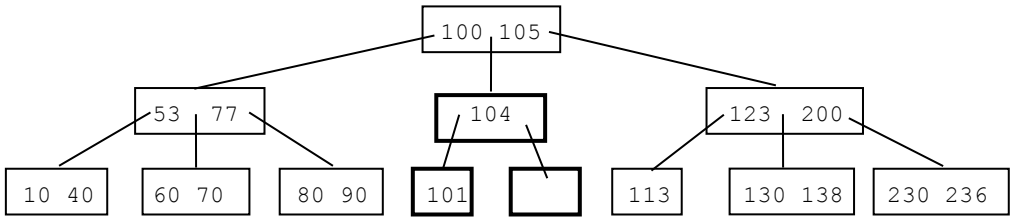
Divida o nó e insira a chave do meio (138) no pai.



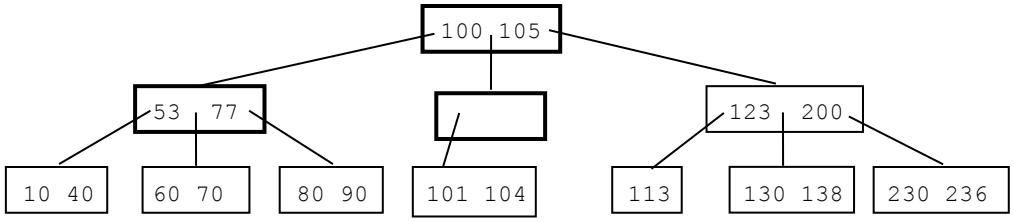
Divida o nó e crie uma nova raiz com a chave do meio (138).



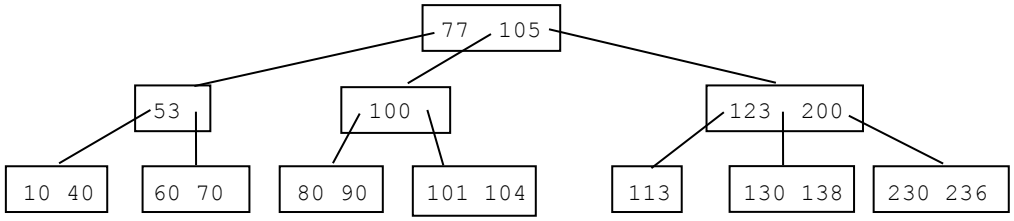
b) Remoção de 103 da árvore original. A chave 103 ocorre em um nó interior e portanto deve ser trocada com a sua sucessora, 104 (a questão pedia para usar a sucessora).



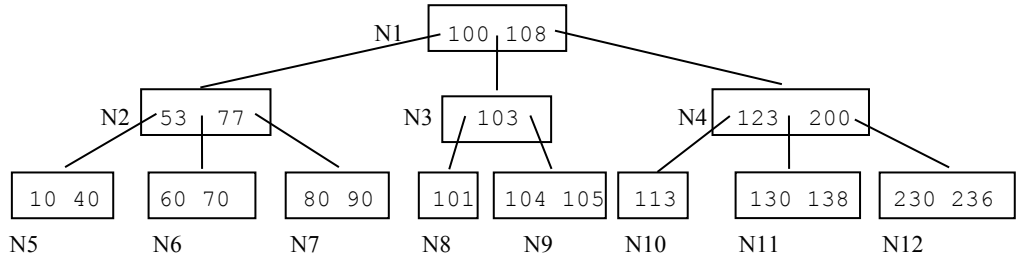
Combine com o pai (caso 5 da remoção em árvores 2-3, aplicado aos nós em negrito acima):



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



(4) (3.0 pontos) Considere a árvore B de ordem 3 abaixo.



Considere um buffer pool tal que possui capacidade de armazenamento de 4 nós e é gerenciado de tal forma que os nós há mais tempo sem serem usados são retirados, quando necessário. Suponha que o buffer pool contém inicialmente o nó N1, conforme mostrado na linha 0 da tabela abaixo, e que mantém o estado da operação anterior quando uma nova operação é iniciada.

a) (1.0 ponto) Assuma que o buffer pool está no estado deixado pela pesquisa da chave 40, conforme mostrado na linha 1 da tabela abaixo. Mostre os estados pelos quais passa o buffer pool quando são pesquisadas as chaves 104 e 10, nesta ordem. Explique sua resposta, seguindo o exemplo da pesquisa da chave 40 na linha 1 da tabela.

b) (1.0 ponto) Mostre os estados por que passa o buffer pool quando é inserida a chave 106, reescrevendo em memória secundárias os nós novos ou modificados. Explique sua resposta.

c) (1.0 ponto) Defina uma nova estratégia de gerência do buffer pool que diminua o número de acessos a memória secundária, quando comparada com aquela adotada no item (a), considerando um grande número de consultas em árvores B, como aquela mostrada na figura acima. Assuma que o buffer pool continua sendo capaz de armazenar 4 nós e que mantém o estado da operação anterior quando uma nova operação é iniciada.

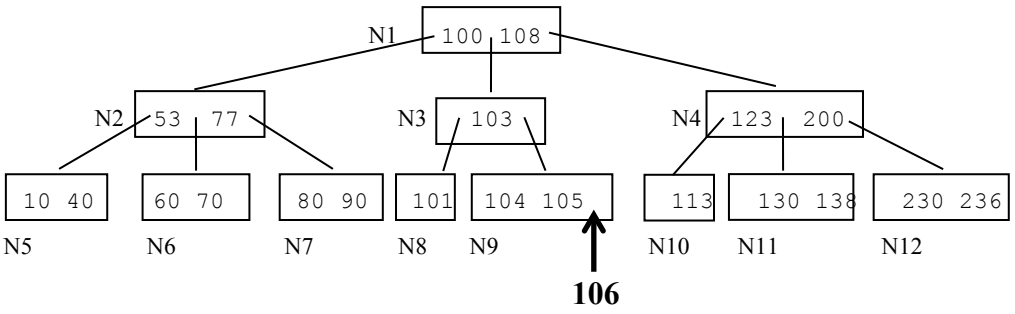
   i) (0.5 ponto) Descreva a sua estratégia e argumente porque ela diminui o número de acessos a memória secundária.

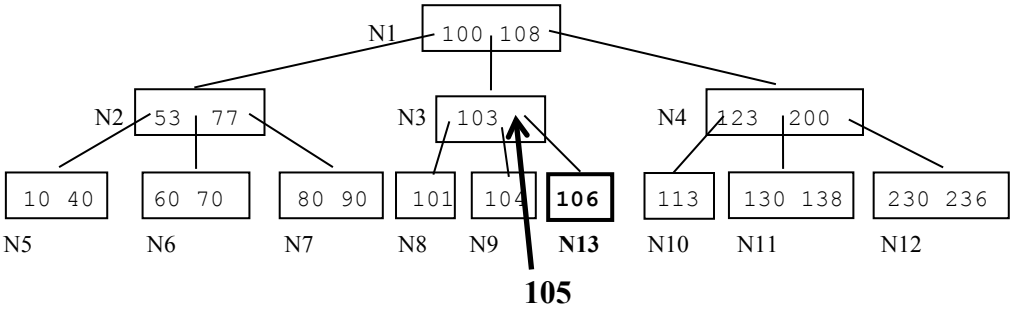
   ii) (0.5 ponto) Refaça as pesquisas do item (a) e mostre quando a nova estratégia diminui o número de acessos a memória secundária.

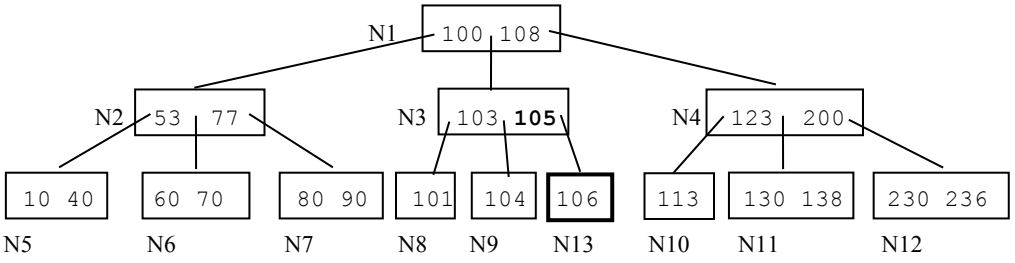
   Em todos os itens, refira-se aos nós usando os rótulos indicados na figura acima. Por exemplo, o nó N1 é a raiz. Use as tabelas abaixo para mostrar o estado do buffer pool e incluir uma explicação para cada passo (use a pesquisa da chave 40 como modelo).

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







(c) É possível tornar a gerência do buffer pool mais eficiente considerando também o nível do nó, em lugar apenas da idade no buffer pool: o nó *N* a ser descartado seria aquele tal que *(L-1)*I* é maior, onde *L* é o nível do nó (a raiz tem nível 1) e *I* é a sua idade no buffer pool.

Operação	Buffer Pool (nó mais novo à direita)	Acesso à memória secundária	Número total de acessos
	(vazio)		
(a) Pesquisa da chave 40	N1 N2 N1 N5 N2 N1	1 1 1	 3
(a) Pesquisa da chave 104	N1 N5 N2 N3 N1 N5 N2 N9 N3 N1 N2	0 1 1 – remove a folha N5, não N2	 2
(a) Pesquisa da chave 10	N1 N9 N3 N2 N2 N1 N9 N3 N5 N2 N1 N3	0 0 – não há necessidade de acessar N2 1 – remove a folha N9, não N3	 1