2ª Avaliação Parcial
Curso: Engenharia de Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Nome: Data ___/___/2018

1ª) Realize os seguintes procedimentos:

a) Insira a seguinte sequência de chaves em uma árvore binária de busca: 29, 22, 33, 65, 44, 12, 56, 41, 17. (1,4 ponto);

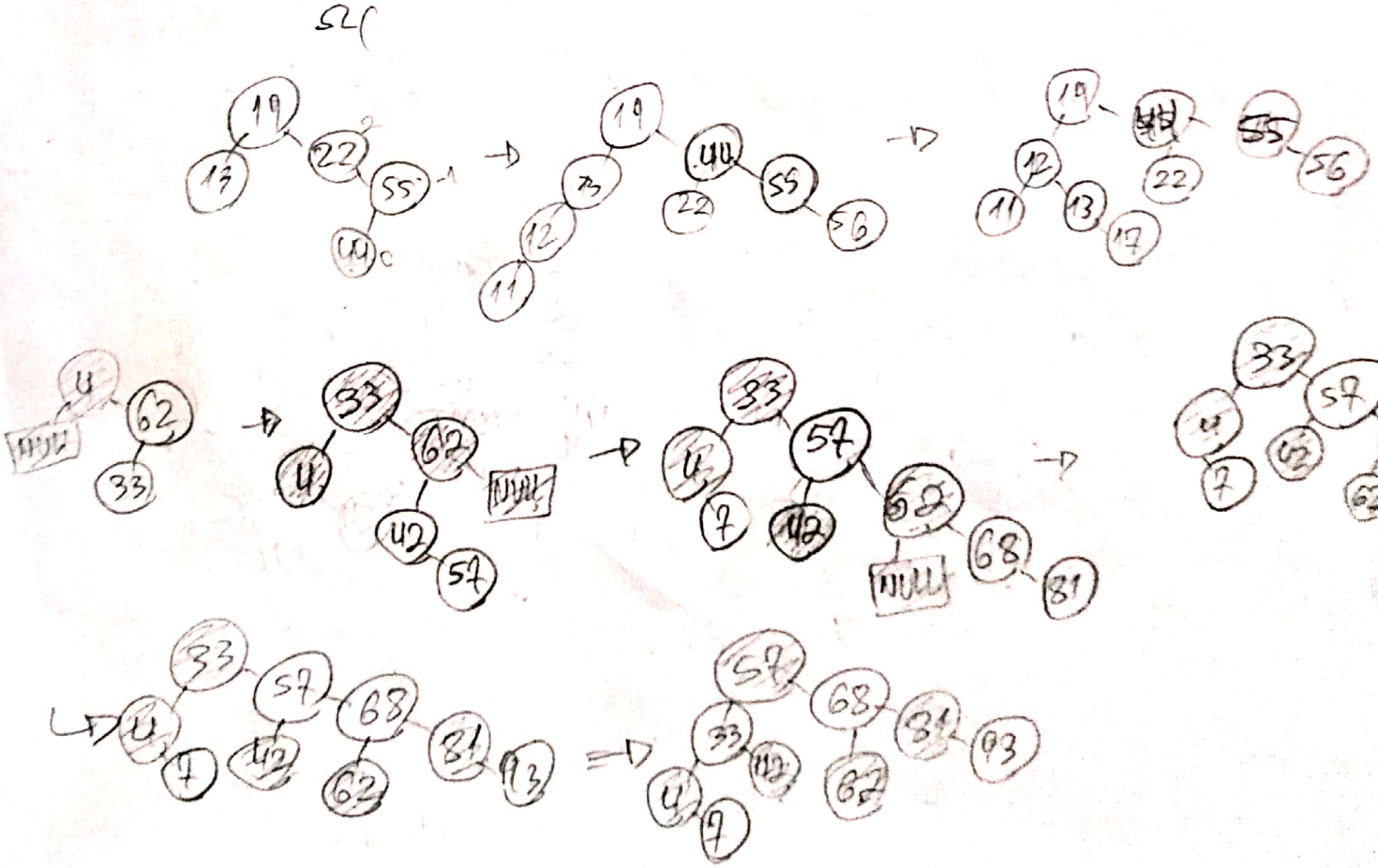
b) Remova os elementos 29, 17 e 12 (1,4 ponto). **Obs**: a remoção deve obrigatoriamente ocorrer na ordem apresentada.

2ª) Insira a seguinte sequência de chaves em uma árvore AVL: 19, 22, 13, 55, 44, 12, 56, 11, 17. (1,8 pontos)

3ª) Construa uma árvore rubro-negra para a seguinte sequência de chaves: 4, 62, 33, 42, 57, 68, 7, 81, 93. (1,8 pontos)

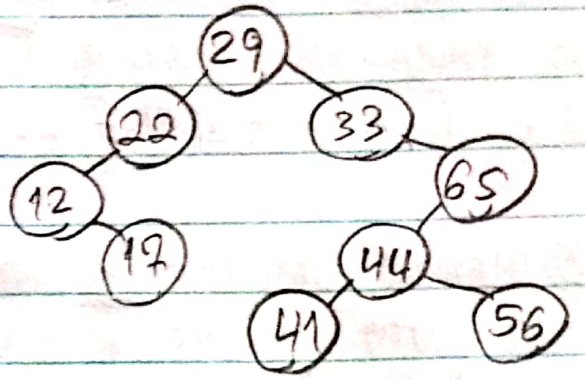
4ª) Explique como funciona o algoritmo **bubblesort**. Quais são suas complexidades no pior e no melhor caso? (1,8 pontos) $O(n^2)$ $\Omega(n)$

5ª) Explique como funciona o algoritmo **quicksort**. Quais são suas complexidades no pior e no melhor caso? (1,8 pontos) $O(n^2)$ $\Omega(n \log(n))$

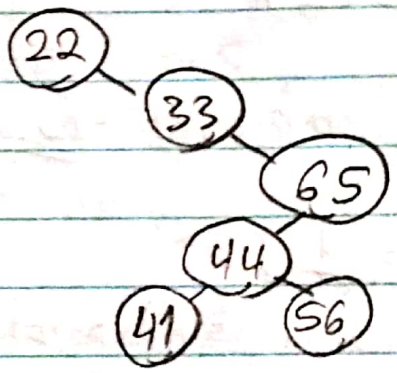


B1

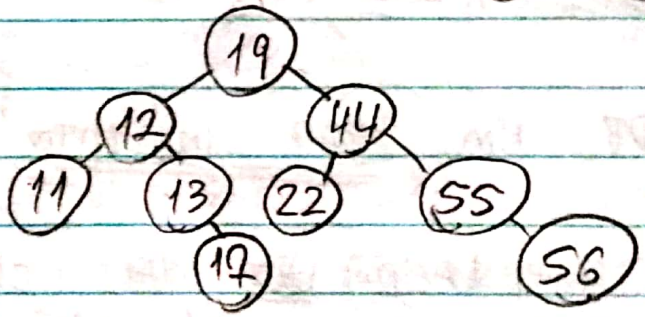
A



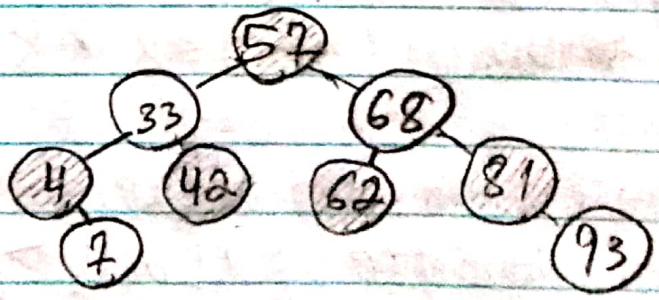
B



02



03



4) BUBBLE SORT

É COMPOSTO POR UM LAÇO for PRINCIPAL ONDE i RECEBE O TAMANHO n DO VETOR V E VAI DECREMENTANDO ATÉ CHEGAR EM ZERO

↳ for(i = n; i > 0; i--) { ... }

- DENTRO DESTE HÁ OUTRO LAÇO for QUE VAI DE ZERO ATÉ i, ONDE HÁ UMA CONDIÇÃO SE V[j] > V[j+1] ESSES ELEMENTOS SÃO TROCADOS

↳
```
for (j=0; j<i; j++) {
    if (V[j] > V[j+1]) {
        temp = V[j];
        V[j] = V[j+1];
        V[j+1] = temp;
    }
}
```

*ESSE PROCESSO LEVA OS MAIORES ELEMENTOS PARA O FINAL DO VETOR (EM SUAS POSIÇÕES CORRETAS) DEPOIS ESSES ELEMENTOS SÃO ISOLADOS LÁ, JÁ QUE j SÓ VAI ATÉ O TAMANHO DE i E i SEMPRE É DECREMENTADO, ENTÃO O PROCESSO CONTINUA PARA O RESTO DO VETOR ATÉ QUE TODOS OS ELEMENTOS ESTEJAM ORDENADOS, O PROCESSO NÃO É MUITO EFICIENTE NA PRÁTICA.

COMPLEXIDADE

↳ PIOR CASO: $O(n^2)$

↳ MELHOR CASO: $\Omega(n^2)$

(05) QUICKSORT → DIVIDIR PARA CONQUISTAR

O ALGORITIMO É UMA FUNÇÃO QUE RECEBE COMO PARAMETRO O TAMANHO n DO VETOR E UM PONTEIRO PARA UM VETOR V

↳ rapida (n, *v) {...}

INICIAMOS ESCOLHENDO UM PIVÔ X, QUE NESSE CASO SERÁ O PRIMEIRO ELEMENTO DO VETOR V[0]

↳ x = v[0];

CRIAMO UMA VARIAVEL a QUE SERÁ INICIALIZADA COM 1

↳ a = 1;

E TAMBÉM OUTRA VARIAVEL b QUE SERÁ INICIALIZADA COM n - 1

↳ b = n - 1;

DENTRO DE UM LAÇO INFINITO (while (1) {...}) FAREMOS O SEGUINTE:

- CRIAMO UM LAÇO QUE VAI INCREMENTA O VALOR DE a ENQUANTO v[a] ≤ x E ENQUANTO a < n

↳ while (v[a] ≤ x && a < n){ a++; }

- CRIAMO UM LAÇO QUE DECREMENTA O VALOR DE b ENQUANTO v[b] > x

↳ while (v[a] > x) { b--; }

- TESTAMOS A CONDIÇÃO SE (a < b) SE ESSA CONDIÇÃO FOR VERDADEIRA TROCAMOS OS VALORES v[a] E v[b] DE LUGAR E

(5) CONTINUAÇÃO...

E INCREMENTAMOS EM a E DECREMENTAMOS b.
SE A CONDIÇÃO NÃO FOR VERDADEIRA NÓS SAÍMOS DO LAÇO INFINITO

```
if (a<b) {
    temp = V[a];
    V[a] = V[b];
    V[b] = temp;
    a++;
    b--;
} else {
    break;
}
```

* QUANDO A FUNÇÃO SAI DO do if OS OUTROS DOIS while SÃO TESTADOS NOVAMENTE E O VALOR DE a E DE b PODEM SER ALTERADOS

QUANDO SAIRMOS DO while(1) NESSE PONTO A POSIÇÃO CORRETA DO PIVÔ SERÁ A POSIÇÃO b DO VETOR, OU SEJA V[b], ENTÃO TROCAMOS V[0]=x E V[b] DE LUGAR.

```
V[0] = V[b];
V[b] = x;
```

ENTÃO APLICAMOS A FUNÇÃO RECURSIVAMENTE À ESQUERDA DA POSIÇÃO b DO VETOR E À DIREITA

```
rapida(b, V); //esquerda do vetor principal
rapida(n-a, &V[a]); //direita do vetor principal
```

NO FINAL O VETOR ESTARÁ ORDENADO.

CONTINUA...

5 CONTINUAÇÃO

A COMPLEXIDADE DO QUICKSORT

- PIOR CASO: $O(n^2)$
- MELHOR CASO: $\Omega(n \log(n))$
- CASO MÉDIO: $\Omega(n \log(n))$