d) $T(n) = n + 2T(n/2)$ → mergesort.

$\sum_{i=0}^{\log_2 n - 1} N = N \times \log_2 N$

$T(n)$

$n + T(n/2) + T(n/2)$

$\frac{n}{2} \times 2 \quad \frac{n}{2} + T(n/4) + T(n/4) \quad \frac{n}{2} + T(n/4) + T(n/4)$

$\frac{n}{4} \times 4 \quad \frac{n}{4} + 2T(n/8) \quad - - -$

3.1

```
typedef struct BTree { int valor;
                       struct BTree *left, *right;
} Node, *BTree
```

a) int size (BTree t);

$T(N)$

$1 + 2T\left(\frac{N-1}{2}\right) + T\left(\frac{N-1}{2}\right)$

$1 + 2T\left(\frac{N-2}{4}\right)$

```
{ int res;
  if (t)
    res = 1 + size (t->left) + size (t->right)
  else res = 0;
  return res;
}
```

árvore balanceada

$T(N) = 1 + 2T\left(\frac{N-1}{2}\right)$

$O(N)$

$T(N)$

$2 \times 1 + 2T\left(\frac{N-1}{2}\right)$

$4 \times 1 + 4T\left(\frac{N-1}{4}\right)$

$\sum_{i=0}^{\log(N-1)} 2^{i+1}$

árvore não balanceada

$T(N) = 1 + T(N-1)$

$\sum_{i=0}^{N-1} 1$

b) int altura (BTree t) (Recorrencia = a anterior

```
{ if (t)
    res = 1 + max (altura (t->esq), altura (t->dir));
  else res = 0;
  return res
}
```

$O(N)$

c) BTree add (BTree t, int x)

```
if (t == NULL) {
  t = (BTree) malloc sizeof (Node);
  t->value = x
  t->left = t->right = null; }
else {
  if (t->value > x) t->left = add (t->left, x)
  else t->right = add (t->right, x);
}
return t; }
```

$T(0) = 4 \rightarrow$ was do lue?

- $T(N) = 1 + 2T(N-1) \Rightarrow O(N) = N$
- $T(N) = 1 + T\left(\frac{N-1}{2}\right) \quad O(N) = \log_{1/2} N$