SUB-ROTINA PARA ORDENAÇÃO POR INTERCALAÇÃO (*MERGE-SORT*)
Cormen et al. (2002), pp. 21-28

Algoritmos

```
//sub-rotina principal
sub-rotina ordenacao_por_intercalacao(A,ini,fim)
início
se ini<fim então                    //só ordena se houver mais de um elemento no arranjo
        meio=(ini+fim)/2;
        ordenacao_por_intercalacao(A,ini,meio);      //divide o problema
        ordenacao_por_intercalacao(A,meio+1,fim);
        intercala(A,ini,meio,fim);                   //combina soluções divididas
fim

//sub-rotina auxiliar
sub-rotina intercala(A,ini,meio,fim)
início
        n1=meio-ini+1;          //calcula o comprimento dos subarranjos
        n2=fim-meio;
        declare arranjos L[1..n1+1] e R[1..n2+1]      //declara subarranjos que serão
                                                     intercalados
        para i=1 até n1 faça            //preenche arranjos a partir de A
                L[i]=A(ini+i);
        para j=1 até n2 faça
                R[j]=A(meio+j+1);
        L[n1+1]=∞;              //coloca sentinela para saber quando o arranjo acabou
        R[n2+1]=∞;
        i=1;                    //intercala os elementos
        j=1;
        para k=ini até fim faça
                se L[i]≤R[j] então
                        A[k]=L[i];
                        i=i+1;
                senão A[k]=R[j];
                        j=j+1;
fim
```

Análise dos algoritmos

Sobre a sub-rotina principal:

- se $n=1$ elemento no arranjo, a ordenação não é necessária, ou seja, é feita apenas uma comparação
  - tempo constante $O(c)$
- se $n>1$
  - o problema é dividido em 2 subproblemas, cada um dos quais com metade do tamanho do problema original: 3 operações, tempo constante $O(c)$
  - cada subproblema é processado: $2T(n/2)$
  - as soluções são combinadas: complexidade da subrotina auxiliar de intercalação, ou seja, $O(n)$

Tem-se, portanto:

$$T(n)=O(c)=1, \text{ se } n=1$$

$$T(n) = 2T(n/2) + O(c) + O(n), \text{ se } n>1$$

Sendo que $O(c) + O(n) = O(n)$, com $c<=n$, temos:

$$T(n) = 2T(n/2) + O(n), \text{ se } n>1$$

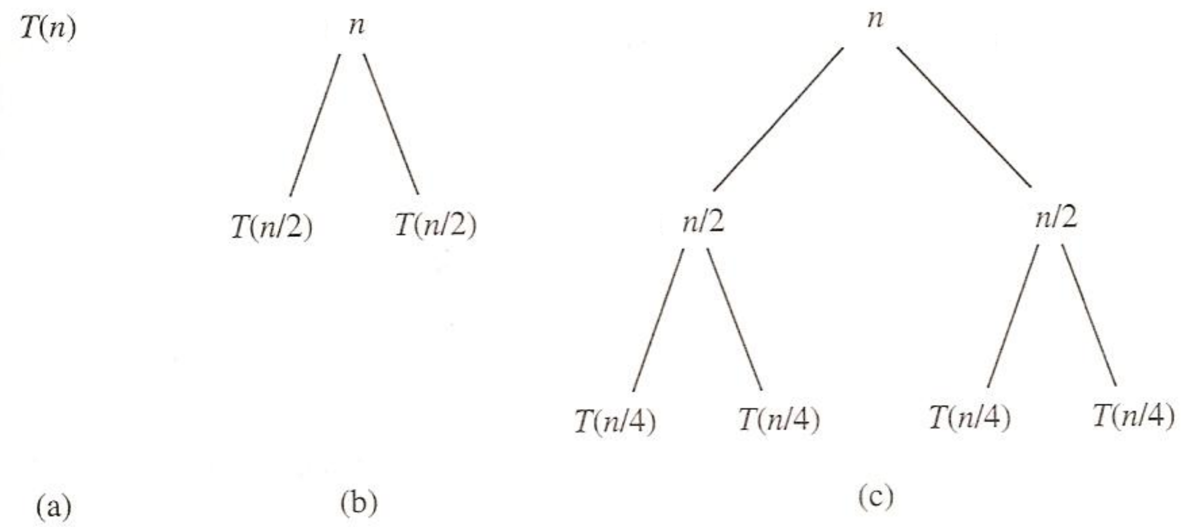
ou simplesmente:

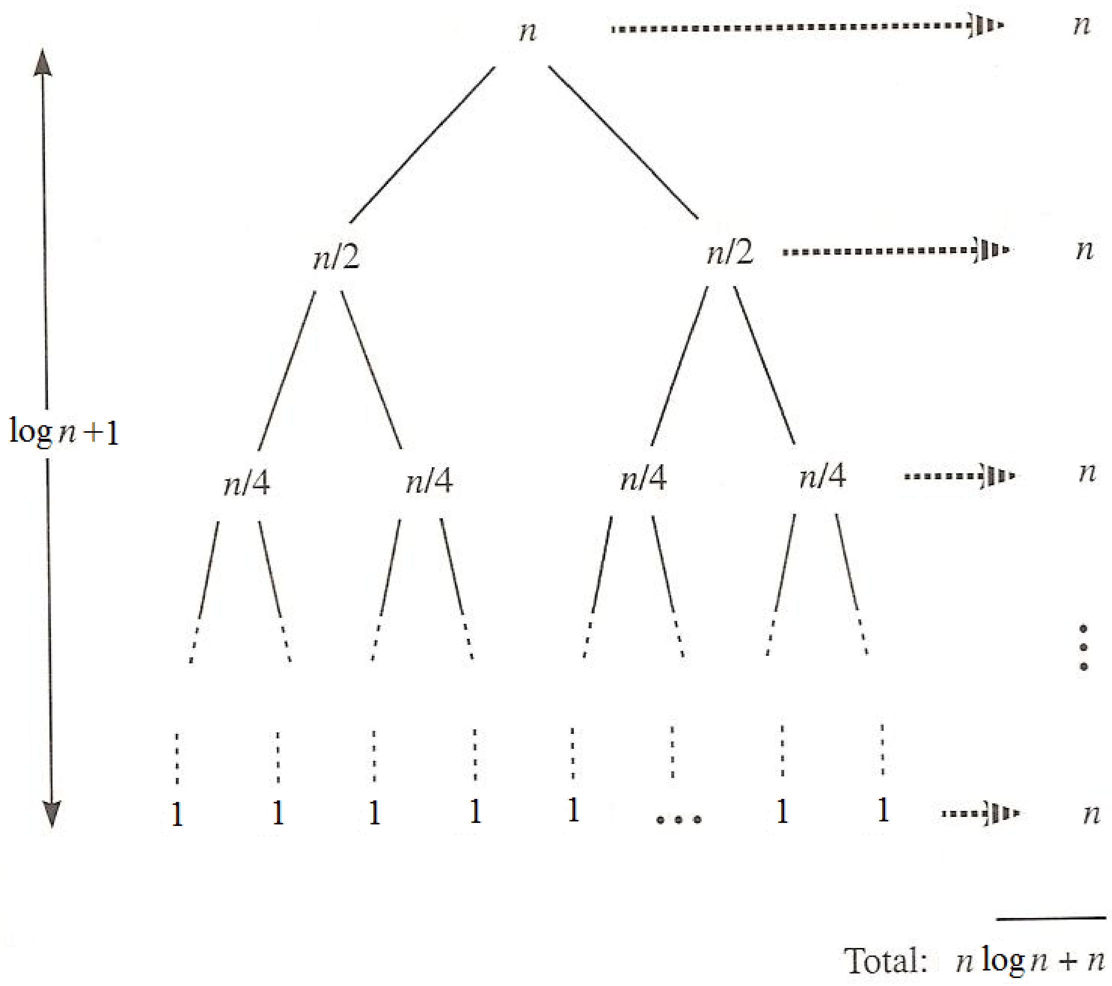
$$T(n) = 2T(n/2) + n, \text{ se } n>1$$

Como resultado, temos:

$$T(n) = 1 \quad \text{se } n=1$$
$$T(n) = 2T(n/2) + n \quad \text{se } n>1$$

Montando a árvore de recorrência





(d)