Aula 17: 16/OUT/2018

Aulas passadas

- problema ordenação: bublesort, seleção ou inserção (aulas 12 e 13)
- recursão: hanoi, fatorial, máximo, ... (aulas 13, 14 e 15)
- recursão: problema do labirinto (aula 16)

Hoje

- intercalação
- mergesort: versão recursiva, versão iterativa
- análise do consumo de tempo: experimental e analítica
- algoritmo por **divisão-e-conquista**

Intercalação

Problema

Dados `v[p:q]` e `v[q:r]` crescentes, rearranjar `v[p:r]` de modo que ele fique em ordem crescente.

Solução

```
#-----------------------------------------------
def intercale(p,q,r,v):
    ''' (int,int,int,list) -> None

    Recebe uma lista v tal que

        - v[p:q] é crescente e
        - v[q:e] é crescente

    e rearranja os elementos de v de tal forma
    que v[p:r] seja crescente.

    Consumo de tempo é O(n) onde n = p-r
    '''
    # crie lista auxiliar
    w = []
    i = p
    j = q
    while i < q and j < r:
        if v[i] < v[j]:
            w.append(v[i])
            i += 1
        else:
            w.append(v[j])
            j += 1

    # copie os elementos que sobraram em v[p:r]
    while i < q:
        w.append(v[i])
        i += 1

    # copie os elementos que sobraram
    while j < r:
        w.append(v[j])
        j += 1

    # copie w[0:r-p] para v[p:r]
    for i in range(r-p):
        v[p+i] = w[i]
```

Mergesort

Ordenação por intercalação.

```
def merge_sort(p,r,v):
    '''(int,int,list) -> None

    Recebe uma lista v[p:r] e rearranja seu elementos
    de maneira que fique crescente.
    '''
    if p < r-1:
        q = (p+r)//2
        merge_sort(p,q,v)
        merge_sort(q,r,v)
        intercale(p,q,r,v)
```

Correção

A função está correta?

A correção da função, que es apoia na correção da função `intercale()` pode ser demonstrada por indução em `n := r-p`.

Consumo de tempo

Desenhar "árvore da recursão" e verificar que o consumo de tempo em cada nível é proporcional a `n` e que há aproximadamente `lg n` níveis.

Divisão e conquista

Algoritmos por divisão-e-conquista, tipicamente, têm três passos em cada nível da recursão:

- **dividir**: o problema é dividido em subproblemas de tamanho menor
- **conquistar**: os subproblema são resolvidos recursivamente e subproblemas "pequenos" são resolvido diretamente.
- **combinar**: as soluções dos subproblemas são combinadas para obter uma solução do problema original